The IBExpert Book: Database Tools for Developers

Author: Holger Klemt

Co-Author: Debra Miles

The IBExpert Book: Database Tools for Developers

ISBN:

First edition: May 2005

Printed and bound in the Federal Republic of Germany for HK–Software, Gerhard–Stalling–Strasse 47a, 26135 Oldenburg, Germany.

www.h–k.de

Author: Holger Klemt
Co–author: Debra J. Miles
Production Manager: Manuel Morbitzer
Cover Design: Bastian Morbitzer

All information and source code in this book is also available online at http://www.ibexpert.info/documentation/.

Dear Reader,

You develop SQL databases, professionally or as a hobby, and need an efficient and powerful tool. With IBExpert you have made the right choice. It enables you in just a short space of time to become acquainted with, and achieve a command of the open source database, Firebird, as well as its commercial relative, Borland InterBase. There are powerful and yet easy–to–learn editors for all vital functions. Development of database objects, database models, stored procedure and trigger programming, performance tuning – all this and much more can be executed simply and quickly using IBExpert.

The Firebird server is an extremely powerful open source database system, which in spite of its very simple installation and administration, offers all essential functions otherwise found only in commercial database systems such as Oracle or Informix. However, following installation the Firebird user has but a few command–line tools at his disposal, there is no powerful GUI tool for data definition and administration included in the kit. This, along with the very limited documentation, is the first hurdle that Firebird users need to overcome.

This is where IBExpert comes to the rescue, whether in the form of the free of charge, functionally limited Personal Edition, the gratis Educational Version or the commercial Full Version including all modules, which is also available as a 45 day Test Version. These resources enable the user to acquire the technical proficiency required for professional applications.

This publication is written for database developers and database administrators, as both will profit from this reference book in their daily work with the Firebird server. All relevant modules are presented in detail and the sundry options (not always immediately obvious) and background information provide both beginners and experts alike with a wealth of information that can only otherwise be laboriously compiled from diverse information sources.

This book would not have been possible without the combined efforts of the HK–Software team. I would like to take this opportunity to thank you all for your support (sorted alphabetically by Christian name):

Alexander Khvastunov – The Main Architect and Programming Expert
Andrea Armbruster – The Commercial and Office Expert
Andrea Schmidt – The Translation Expert
Bastian Morbitzer – The Design Expert
Debra J. Miles – The Translation and Documentation Expert
Manuel Morbitzer – The Web and Programming Expert
Raphael Baier – The Future Generation Expert
Uwe Klemt – The Server and Network Expert

Furthermore I would like to thank all those at the Firebird Foundation who are actively engaged in the continued development of the Firebird server. The Firebird Project can be supported in many ways: subscription membership, active programming or personal contributions to the newsgroups and other media. This also helps the Firebird Project attain the level of awareness and recognition that it deserves.

Following the success of open source operating systems, particularly in the area of server installations, the database market segment is now also on the brink of such a breakthrough. Firebird is here certainly one of the most powerful platforms, and IBExpert the ideal complement for discerning database developers and administrators.

Holger Klemt

Oldenburg, Germany
May 2005

Contents

1 Getting Started

What you need to get started with IBExpert

All you need to get started is a Borland InterBase (version 7 or earlier) or Firebird (version 1.5 or earlier) client installed on your workstation/PC and a connection to an InterBase or Firebird server (either locally or remote).

And of course IBExpert should be installed on the computer, where you intend to work.

IBExpert works ideally on operating systems not older than Win 98; and only 10MB memory is needed to get started.

Please refer to The First Steps for details of how to install Firebird and IBExpert.

1.1 The First Steps

In order to start working and developing with IBExpert, it is necessary to take the following steps:

- Download and install Firebird (Open Source database)
- Download and install IBExpert (Personal, Trial or Customer edition)
- Register a database (the example uses the EMPLOYEE database supplied with Firebird and InterBase)
- Working with a database (based on the EMPLOYEE sample database).

Please refer to the individual subjects for more information.

1.1.1 Download and Install Firebird

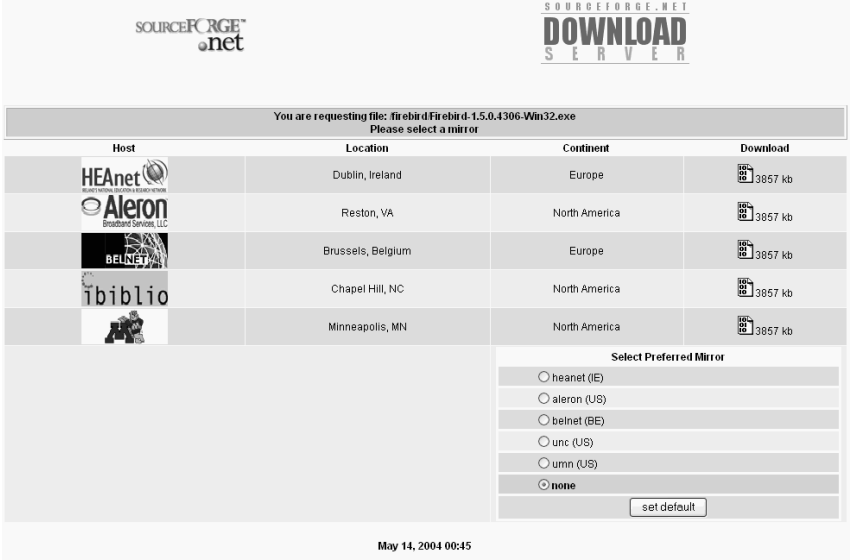
The current Firebird version can be downloaded free of charge from http://firebird.sourceforge.net/, or http://firebirdsql.org subject to Open Source conditions.

Alternatively, use the IBExpert Help menu item Download Firebird to directly access the download website.

Simply click the DOWNLOAD tab and select All Released Packages. Scroll down to the latest file releases and click DOWNLOAD to the right of the version for your platform, for example **firebird_win32** for the Windows version (most current version in March 2005 is Firebird 1.5.2 from 24th December 2004; Firebird 2.00 Alpha 01 was released on March 21 2005, currently for testing purposes only). Please refer to Posix Platforms and Windows Platforms for further information for individual platforms with regard to download and installation.

A new window appears:

Click on the **Firebird-1.5.2.4731-Win32.exe** file. Select a download server:



and click the download symbol. Specify drive and path for the download file and save.

Now double-click the downloaded firebird file to start the installation. Again, please refer to Posix Platforms and Windows Platforms for installation details for the various platforms.

Read and accept the Firebird License Agreement, before proceeding further.

Please note that the Firebird server, along with any databases you create or connect to, must reside on a hard drive that is physically connected to the host machine. It is not possible to locate components of the server or database on a mapped drive, a file system share or a network file system.

It is important to note that the Firebird server must be installed on the target computer. The new Firebird 1.5. Embedded Server version provides a useful enhancement: the client library is embedded in the server, this combination performing the work of both client and server, for a single attached application.

Should problems be encountered during installation, please refer to the Firebird Information file.

Copy of Firebird License Agreement

INTERBASE PUBLIC LICENSE
Version 1.0

1. Definitions. 1.0.1. "Commercial Use" means distribution or otherwise making the Covered Code available to a third party.
1.1. ''Contributor'' means each entity that creates or contributes to the creation of Modifications.
1.2. ''Contributor Version'' means the combination of the Original Code, prior to Modifications used by a Contributor, and the Modifications made by that particular Contributor.
1.3. ''Covered Code'' means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.
1.4. ''Electronic Distribution Mechanism'' means a mechanism generally accepted in the software development community for the electronic transfer of data.
1.5. ''Executable'' means Covered Code in any form other than Source Code.
1.6. ''Initial Developer'' means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.
1.7. ''Larger Work'' means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.
1.8. ''License'' means this document.
1.8.1. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.
1.9. ''Modifications'' means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:
A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.
B. Any new file that contains any part of the Original Code or previous Modifications.
1.10. ''Original Code'' means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.
1.10.1. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.
1.11. ''Source Code'' means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a com-

pressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.
1.12. "You'' (or "Your") means an individual or a legal entity exercising rights under, and complying with, all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You'' includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control'' means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. Source Code License. *2.1. The Initial Developer Grant.* The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

(b) under Patents Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

(c) the licenses granted in this Section 2.1(a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices.
2.2. Contributor Grant. Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) the licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations. *3.1. Application of License.* The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be dis-

tributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.

Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications. You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters (a) Third Party Claims.

If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL'' which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs.

If Contributor's Modifications include an application programming interface and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the LEGAL file.

(c) Representations.

Contributor represents that, except as disclosed pursuant to Section 3.4(a) above, Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices. You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients' rights or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do

so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.
3.6. Distribution of Executable Versions. You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.
3.7. Larger Works. You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation. If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License. This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License. *6.1. New Versions.* Inprise Corporation (''Inprise'') may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.
6.2. Effect of New Versions. Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Inprise. No one other than Inprise has the right to modify the terms applicable to Covered Code created under this License.
6.3. Derivative Works. If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must (a) rename Your license so that the phrases ''Mozilla'', ''MOZILLAPL'', ''MOZPL'', ''Netscape'', "MPL", ''NPL", "Inprise", "ISC", "InterBase", "IB''

or any confusingly similar phrase do not appear in your license (except to note that your license differs from this License) and (b) otherwise make it clear that Your version of the license contains terms which differ from the Mozilla Public License and Netscape Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)
6.4 Origin of the InterBase Public License. The InterBase Public License V 1.0 is based on the Mozilla Public License V 1.1 with the following changes:
1. The license is published by Inprise Corporation. Only Inprise Corporation can modify the terms applicable to Covered Code.
2. The license can be modified and used for code which is not already governed by this license. Modified versions of the license must be renamed to avoid confusion with Netscape's or Inprise Corporation's public license and must include a description of changes from the InterBase Public License.
3. The name of the license in Exhibit A is the "InterBase Public License".
4. The reference to an alternative license in Exhibit A has been removed.
5. Amendments I, II, III, V, and VI have been deleted.
6. Exhibit A, Netscape Public License has been deleted
7. A new amendment (II) has been added, describing the required and restricted rights to use the trademarks of Inprise Corporation.

7. DISCLAIMER OF WARRANTY. COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS'' BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION. 8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.
8.2. If You initiate litigation by asserting a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that:
(a) such Participant's Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either: (i) agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of Modifications made by such Participant, or (ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant. If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the

60 day notice period specified above.

(b) any software, hardware, or device, other than such Participant's Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant's Contributor Version directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. LIMITATION OF LIABILITY.UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS. The Covered Code is a ''commercial item,'' as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of ''commercial computer software'' and ''commercial computer software documentation,'' as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. MISCELLANEOUS. This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS. As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

13. MULTIPLE-LICENSED CODE. Initial Developer may designate portions of the Covered Code as "Multiple-Licensed". "Multiple-Licensed" means that the Initial Developer permits you to utilize portions of the Covered Code under Your choice of the IPL or the alternative licenses, if any, specified by the Initial Developer in the file described in Exhibit A.

*EXHIBIT A - InterBase Public License.*The contents of this file are subject to the InterBase Public License Version 1.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.Inprise.com/IPL.html
Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.
The Original Code was created by Inprise Corporation and its predecessors.
Portions created by Inprise Corporation are Copyright (C) Inprise Corporation. All Rights Reserved.
Contributor(s): _____.

AMENDMENTS I. Inprise and logo. This License does not grant any rights to use the trademarks "Inprise'', "InterBase," "Java" or "JavaScript" even if such marks are included in the Original Code or Modifications.
II. Trademark Usage.
II.1. Advertising Materials. All advertising materials mentioning features or use of the covered Code must display the following acknowledgement: "This product includes software developed by Inprise Corporation. "
II.2. Endorsements. The names "Inprise," "InterBase," "ISC," and "IB" must not be used to endorse or promote Contributor Versions or Larger Works without the prior written permission of Inprise.
II.3. Product Names. Contributor Versions and Larger Works may not be called "Inprise" or "InterBase" nor may the words "Inprise" or "InterBase" appear in their names without the prior written permission of Inprise Corporation.

Copy of Firebird Information file

Firebird Database Server 1.5 Final Release

This document is a guide to installing this package of Firebird 1.5 on the Win32 platform. These notes refer to the installation package itself, rather than Firebird 1.5 in general.

** IMPORTANT NOTE **

If you used one of the installable binaries from Sourceforge to install RC2 it is recommended that you uninstall directly from the installation directory with `unins000.exe`.

Do not use the Control Panel as the path to the uninstaller. There appears to be a bug that prevents the server service from being shut down if the uninstaller is run from there.

This only applies to the uninstaller with RC2 from Firebird on Sourceforge. Later installable binaries can be uninstalled via any available means.

** END **

Contents

Before installation

It is recommended that you `UNINSTALL` all previous versions of Firebird 1.0, Firebird 1.5 or InterBase before installing this package.

New features of the installer in Firebird 1.5

- This installer now combines the super server and classic server binaries into a single installation package. You can choose to install one or the other, but not both. To switch server type you need to uninstall and re-install.
- The rules for library installation have changed considerably. They are explained in detail in `.\docs\README.Win32LibraryInstallation.txt` which will be available to you after installation.
  As a result of these new rules the installer checks for an existing install of Firebird or InterBase.

- If Firebird 1.5 is already installed it will attempt to install over it. If the server is running it will halt the install.
- If another version of Firebird or InterBase is already installed it will warn the user. If the user continues the installer will install Firebird and set up registry entries but it will not configure Firebird to run, either as a service or as an application. This must be done manually.
- The installer has a new command-line option `/force` which allows those with a 'devil may care' attitude to override the above.

- If an amended firebird.conf exists in the installation directory it is saved as: `firebird.conf.saved.n` where `n` is a number. The installer always installs the default `firebird.conf` file. This is to guarantee consistency to the installation process. Otherwise the installer would have to parse the existing (and possibly broken) configuration file.

Deprecated Features related to installation

- Firebird 1.0 reserved a new registry key for Firebird use. It was: `HKLM\SOFTWARE\FirebirdSQL` This is now deprecated and will be deleted by the installer. If you have applications which rely on this key you should add it back manually.
  However, it is preferable if you rebuild your application to read the new key.
- Earlier Firebird 1.5 release candidates installed `fbclient.dll` in the `<system>` directory. This practice is now deprecated. An option to install it into the `<system>` directory is available at install time. However, it is preferable if you rebuild your applications to conform to the new usage of `fbclient`.

Deinstallation

- It is preferred that this package be uninstalled correctly using the uninstallation application supplied. This can be called from the Control Panel.
  Alternatively it can be uninstalled by running `unins000.exe` directly from the installation directory.
- If Firebird is running as an application (instead of as a service) it is recommended that you manually stop the server before running the uninstaller. This is because the uninstaller cannot stop a running application. If a server is running during the uninstall the uninstall will complete with errors. You will have to delete the remnants by hand.

Other Notes

Firebird requires WinSock2. All Win32 platforms should have this, except for Win95. A test for the Winsock2 library is made during install. If it is not found the install will fail. You can visit this link:

http://support.microsoft.com/default.aspx?scid=kb;EN-US;q177719

to find out how to go about upgrading.

Installation from a batch file

The setup program can be run from a batch file. The following parameters may be passed:

`/SP-` Disables the 'This will install... Do you wish to continue?' prompt at the beginning of Setup.

`/SILENT, /VERYSILENT` Instructs Setup to be silent or very silent. When Setup is silent the wizard and the background window are not displayed but the installation progress window is. When a setup is very silent this installation progress window is not displayed. Everything else is normal so for example error messages during installation are displayed and the startup prompt is (if you haven't disabled it with the '`/SP-`' command line option explained above).

If a restart is necessary and the '`/NORESTART`' command isn't used (see below) and Setup is silent, it will display a Reboot now? messagebox. If it's very silent it will reboot without asking.

`/NORESTART` Instructs Setup not to reboot even if it's necessary.

/DIR="x:\dirname" Overrides the default directory name displayed on the Select Destination Directory wizard page. A fully qualified pathname must be specified. If the [Setup] section directive DisableDirPage was set to yes, this command line parameter is ignored.

/GROUP="folder name" Overrides the default folder name displayed on the Select Start Menu Folder wizard page. If the [Setup] section directive DisableProgramGroupPage was set to yes, this command line parameter is ignored.

/NOICONS Instructs Setup to initially disable the Don't create any icons check box on the Select Start Menu Folder wizard page.

/COMPONENTS="comma separated list of component names" Choose from -

- SuperServerComponent,
- ClassicServerComponent,
- ServerComponent,
- DevAdminComponent and
- ClientComponent

Overrides the default components settings. Using this command line parameter causes Setup to automatically select a custom type. A full install requires combining components. For example:

```
/COMPONENTS="SuperServerComponent, ServerComponent,
DevAdminComponent, ClientComponent"
```

would be required for a full install.

/FORCE Tells the installer to ignore its analysis of the existing environment. It will attempt to install and configure Firebird 1.5 as if no previous version of Firebird or InterBase was installed.

This can be useful if you have a seriously broken installation that you cannot uninstall. Or it could be another way to aggravate your users by breaking a perfectly good working install of InterBase. It's your choice.

/NOCPL Don't install the Control Panel Applet. This is useful for two reasons:

- Installing/Uninstalling the CPL applet will often require a system restart.
- You may wish to use an alternative CPL applet.

/NOGDS32 Don't install a copy of the client library into the system directory, even if installation analysis concludes it is OK to do so.

/COPYFBCLIENT Copy the fbclient.dll to the system directory. This is recommended for client installs if you are sure that you will only ever be accessing a single server version. If your client applications are likely to take advantage of accessing different server versions this is not recommended. See doc/README.Win32LibraryInstallation.txt for more information.

Windows platforms

On Windows server platforms - Windows NT, 2000 and XP, the Firebird service is started upon completion of the installation. It starts automatically every time the server is booted up.

The non-server Windows platforms, Windows 95, 98 and ME, do not support services. The installation starts the Firebird server as an application, protected by another application known as the Guardian. Should the server application terminate abnormally, the Guardian will attempt to restart it.

Excerpts of this article have been taken from the IBPhoenix "Firebird Quick Start Guide". Many thanks to Paul Beach (www.ibphoenix.com)!

Posix platforms

As there may be significant variations from release to release of any Posix operating system, especially the open source one, it is important to read the release notes pertaining to the Firebird version to be installed. These can be downloaded from the Download page at http://firebird.sourceforge.net. Further help can be found at http://firebird.sourceforge.net/index.php?op=doc&id=install.

Please consult the appropriate platform documentation, if you have a Linux distribution supporting rpm installs, for instructions about using the RedHat Package Manager. Most distributions offer the choice of performing the install from a command shell or through a GUI interface.

For Linux distributions that cannot process rpm programs, use the "`.tar.gz`" kit. Again instructions are included in the release notes (see above link).

Shell scripts have been provided, but in some cases, the release notes may advise modification of the scripts as well as some manual adjustments.

Excerpts of this article have been taken from the IBPhoenix "Firebird Quick Start Guide". Many thanks to Paul Beach (www.ibphoenix.com)!

Performing a client–only install

Each remote client machine needs the client library that matches the release version of the Firebird server: `libgds.so` on Posix clients; `gds32.dll` on Windows clients.

Firebird versions from 1.5 onward require an additional client library, `libfb.so` or `fb32.dll`, which contains the full library. In these newer distributions, the "`gds`"-named files are distributed to maintain compatibility with third-party products which require these files. Internally, the libraries jump to the correct access points in the renamed libraries.

Also needed for the client-only install:

Windows

If you want to run Windows clients to a Linux or other Posix Firebird server, you need to download the full Windows installation kit corresponding to the version of Firebird server installed on the Linux or other server machine.

Simply run the installation program, as if you were going to install the server, selecting the `CLIENT ONLY` option in the Install menu.

Linux and some other Posix clients

Some Posix flavors, even within the Linux constellation, have somewhat idiosyncratic requirements for file system locations. For these reasons, not all *x distributions for Firebird even contain a client-only install option.

For the majority, the following procedure is suggested for Firebird versions lower than 1.5. Log in as root for this.

- Search for `libgds.so.0` in `/opt/interbase/lib` on the machine where the Firebird server is installed, and copy it to `/usr/lib` on the client.
- Create the symlink `libgds.so` or it, using the following command:
  `ln -s /usr/lib/libgds.so.0 /usr/lib/libgds.so`
- Copy the `interbase.msg` file to `/opt/interbase`.
- In the system-wide default shell profile, or using **setenv()** from a shell, create the `INTERBASE` environment variable and point it to `/opt/interbase`, to enable the API routines to locate the messages.

Excerpts of this article have been taken from the IBPhoenix "Firebird Quick Start Guide". Many thanks to Paul Beach (www.ibphoenix.com)!

Performing a minimum Firebird 1.5 client install

by Stefan Heymann (April 11th 2004)

This article describes how to run Firebird 1.5 based applications with the absolute minimum client installation required.

What you need

Your application needs access to the Firebird client library, `fbclient.dll`. The easiest way to do this is to put `fbclient.dll` in the same directory as your application's .exe file.

`fbclient.dll` needs access to two other DLLs: `msvcp60.dll` and `msvcrt.dll`. Both are delivered together with the Windows installation of Firebird, so if you have a Firebird server installed on your development machine, you'll find these DLLs in the bin directory of your Firebird installation.

`msvcrt.dll` (Microsoft Visual C/C++ RunTime) is a part of Windows and resides in the Windows\System directory on Win9x machines and in Windows\System32 on NT-based machines (NT4, W2K, XP, 2003). On Windows 95 and Windows 98 machines, it's too old for the `msvcp60.dll` that `fbclient.dll` uses. So you'll have to replace the `msvcrt.dll` by the one that comes with Firebird (or even a newer one).

msvcp60.dll can stay in your application directory.

Your application directory now looks like this:

```
<YourApp>.exe and other application files
fbclient.dll
msvcp60.dll
```

That's it. Easy!

What you have to write to the Registry

Nothing - there's nothing you'll have to do to the registry.

What you have to do to the Windows\System directory

Only on Windows 95 and Windows 98 "First Edition" machines: you will need to replace msvcrt.dll with the newer version that comes with Firebird 1.5 (if there isn't already a new version installed).

Some version numbers of msvcrt.dll:

Windows 98 FE	5.00.7128	does NOT work
Windows 98 SE	6.00.8397.0	works
Firebird 1.5.0	6.00.8797.0	works
Windows XP SP1	7.0.2600.1106	works

What you have to do to your code (Delphi, IBObjects)

A "normal" InterBase access library uses gds32.dll as the client library. Firebird's client library is named fbclient.dll. If you use IBObjects (http://www.ibobjects.com/), you can set another client library name.

- Include IB_Constants.pas as the first unit in your USES clause .
- Put the following line in the INITIALIZATION part of your Unit:
  IB_Constants.IB_GDS32 := 'fbclient.dll';
- This line must be executed before the first database connect is performed.

Classic Server versus SuperServer

Many thanks to Paul Beach of IBPhoenix for this article.

- The InterBase SuperServer Architecture
- The InterBase Classic Architecture
- Comparing Classic and SuperServer
- Why Two Implementations?

InterBase SuperServer architecture

SuperServer is a multi-client, multi-threaded implementation of the InterBase server process. This implementation replaces the "Classic" implementation used for previous versions of InterBase.

SuperServer serves many clients at the same time using threads instead of separate server processes for each client. Multiple threads share access to a single server process. The benefits of SuperServer architecture include:

Having a single server process eliminates bottlenecks resulting from arbitration for shared database pages and reduces the overhead required for multiple process start-ups and database queries.

SuperServer improves message interaction performance because a shared library call is always faster than an interprocess communication request to a server process.

SuperServer improves database integrity because only one server process has write access to the database, rather than one process for each client. All database engine functionality is encapsulated into a unified, protected subsystem that is isolated from user application error.

SuperServer allows for the collection of database statistics and user information that InterBase's tools can use for performance monitoring and administrative tasks.

SuperServer is more cost-effective than the Classic architecture. All operating systems have limits on the number of OS processes that can run concurrently. SuperServer allows for a fixed number of database threads to be multiplexed over a potentially large number of concurrent database connections. Since these threads are not hard-wired to any specific database connection, SuperServer can support a larger number of users with minimum resources use.

InterBase Classic architecture

Classic architecture, the design in InterBase 4.0 and earlier, was process-based. For every client connection, a separate server process was started to execute the database engine, and each server process had a dedicated database cache. The server processes contended for access to the database, so a Lock Manager subsystem was required to arbitrate and synchronize concurrent page access among the processes.

Invoking the Classic Server

The InterBase Classic server runs on demand as multiple processes. When a client attempts to connect to an InterBase database, one instance of the `gds_inet_server` executable runs and remains dedicated to that client connection for the duration of the connection.

The initiator of `gds_inet_server` is `inetd`, the UNIX service turnkey process. It has a configuration file, `/etc/inetd.conf`, which associates services with the executable that is to receive the connection. When `inetd` receives a connection request for a given service, it looks up the appropriate program in `/etc/inetd.conf`, executes it, and transfers the network connection to the service program.

When the client chooses to disconnect, gds_inet_server closes its connection to the database and any other files, and then exits. When there are no clients connected to any database, there should be no invocations of gds_inet_server running.

Lock management

Lock management is taken care of by another process, gds_lock_mgr. This program is started when the second client attaches to a given database. The job of the lock manager is to serve (metaphorically) as a traffic cop. It grants locks on database resources to clients. It also requests that clients relinquish locks on a resource when that resource is in demand by other clients. The gds_lock_mgr remains running even after the last client disconnects. The next time a client connects, it can avoid the slight overhead of starting the lock manager process.

Use of Posix signals

The gds_lock_mgr process communicates with each client process by using a shared memory area, and a signaling mechanism using the POSIX signals SIGUSR1 and SIGUSR2. Signals are caught in signal handling routines in libgdslib.a, and for this reason user applications should not perform signal handling or any modification to the signal mask. Applications which need to use POSIX signals must compile with an alternate InterBase library, libgds.a. This library functions identically to libgdslib.a, but it handles signals sent by the lock manager in a child process called gds_pipe. All client applications compiled with libgds.a automatically run with this child process. No changes to application code are needed, only a different linking option.

Resource use

Each instance of gds_inet_server keeps a cache of database pages in its memory space, which is likely to result in some duplication of cached data across the system. While the resource use per client is greater than in SuperServer, Classic uses less overall resources when the number of concurrent connections is low.

Local access method

The Classic architecture permits application processes to perform I/O on database files directly, whereas the SuperServer architecture requires applications to request the ibserver I/O operations by proxy, using a network method. The local access method is faster than the network access method, but is only usable by applications which run on the same host as the database.

Monitoring

The database information call for active connections always reports exactly one connection on a Classic server, no matter how many clients are connected to databases on that server. The reason for this is that every client connection has its own gds_inet_server process on the server, and each instance of that program knows only about its own connection. Only in SuperServer does the server process have the ability to report all client connections on the server.

Security

In order for InterBase Classic to work with a mixture of local and remote clients running as different user ID's, the server executables `gds_inet_server` and `gds_lock_mgr` must run as root.

The processes must run with a real uid of root to set their effective uid to that of the client uid. The lock manager must have the superuser privilege to send signals to the processes. In some IT environments, the presence of executables with setuid bits turned on raises concerns about security. Nevertheless, do not change the runtime configuration of InterBase server. The setuid root configuration of the Classic software is important to its function.

Because applications can run as any uid, database files must be writable by all uids that access the databases. To simplify maintenance, database files are created writable by the whole world.

With care, you can restrict these file permissions, so that the database files are safe from accidental or deliberate damage. Make sure you understand file permissions completely before attempting this, because all local and remote clients need write access to the database, even if they intend only to read data.

Classic versus SuperServer

Please refer to:

- Invoking SuperServer
- Lock Management
- Resource Use
- Threaded Server and UDFs
- Security

Invoking SuperServer

SuperServer runs as a single process, an invocation of the ibserver executable. ibserver is started once by the system administrator or by a system boot script. This process runs always, waiting for connection requests. Even when no client is connected to a database on the server, ibserver continues to run quietly.

The SuperServer process is not dependant on `inetd`; it waits for connection requests to the `gds_db` service itself.

The SuperServer process is a multi-threaded application. Different threads within the process are dedicated to different tasks. For instance, one thread waits on the `gds_db` service port for incoming connection requests. Other threads are analogous to individual `gds_inet_server` processes in the Classic model, serving client queries. Another thread serves as the lock manager, replacing the `gds_lock_mgr` process from the Classic model.

Lock management

The lock manager in SuperServer is implemented as a thread in the ibserver executable. Therefore InterBase does not use the `gds_lock_mgr` process. Likewise, POSIX signals are not used by the lock manager thread in SuperServer; interthread communication mechanisms are used.

Resource use

The SuperServer implementation has less overhead and uses fewer system resources per client connection than the Classic model. SuperServer has one cache space for all client attachments, allowing more efficient use of cache memory. For these and other reasons, SuperServer has demonstrated an ability to efficiently serve a higher number of concurrent clients.

Threaded server & UDFs

User-Defined Functions (UDFs) are libraries of functions that you can add to extend the set of functions that the InterBase server supports. The functions in your UDF library execute within the process context of the InterBase server. Due to the threaded implementation of SuperServer, there are issues with UDFs that require that you write UDF functions more carefully than when writing UDFs for a Classic server.

You must design UDFs for SuperServer as thread-safe functions. You cannot use global variables in your UDF library, because if two clients run the UDF simultaneously, they conflict in their use of the global variables.

Do not use thread-local global variables to simulate global variables. SuperServer implements a sort of thread pooling mechanism, to share threads among all the client connections. It is likely that if a given client executes a UDF twice, that each execution is not executed in the context of the same thread. Therefore, you cannot depend on thread-local variables keeping values from one execution of the UDF to the next for a given client.

UDFs that allocate memory dynamically run the risk of creating a memory leak. Because SuperServer is supposed to stay up and running indefinitely, not just for the duration of the client connection, memory leaks can be more damaging in SuperServer than in Classic. If your UDFs return dynamically allocated objects, then you must use `malloc()` to allocate the memory for these objects (on Win32, you must use `ib_util_malloc()` or the `malloc()` that is part of the Microsoft Visual C++ runtime library). Do not use new or `globalalloc()` or the Borland `malloc()`.

Finally, such functions must be declared in databases with the `FREE_IT` option of the `DECLARE EXTERNAL FUNCTION` statement.

By contrast, in Classic, there is a separate process for each client connection, so the UDFs are guaranteed not to conflict. Global variables are safe to use. Also, memory leaks are not as dangerous, because any leaked memory is released when the client disconnects. InterBase recommends that you design UDFs for SuperServer, the more restrictive model, even if you use a version of InterBase implemented with the Classic model. Eventually InterBase will be implemented with SuperServer on the platform you

use. If you design UDFs with this assumption, you can upgrade to a later version of InterBase without the risk that your UDFs must be redesigned to work with SuperServer.

Security

SuperServer can be configured to run as a non-root uid, for enhanced security. In SuperServer, you can restrict the permissions on database files to allow only the InterBase server uid to access the database.

Why two implementations?

The Classic implementation predates the SuperServer implementation, and the SuperServer implementation is the future of InterBase. Classic configuration is used on operating systems that currently don't have the technology for threaded applications, which is required for SuperServer. InterBase also distributes the Classic version on platforms that have threading technology, but which benefit from the low-profile implementation.

SuperServer has a greater ability to meet the demands of a growing multi-user system, while retaining good performance and efficiency. SuperServer is implemented in InterBase product on all platforms where it is technically practical. It is the intention that SuperServer is the future direction of InterBase on all platforms.

Changing server to solve undefined crashes

September 2004. Many thanks to Gerhard Behnke at dpa (Deutsche Presse Agentur) for this contribution.

We managed to solve our problem with undefined Firebird crashes in the following way:

W2003/Superserver

It is essential to check Firebird's memory requirements using the Task Manager. If the requirements are approaching 2 GB, there is a danger of Firebird crashing, e.g. if more than 2 GB is required when submitting a long and detailed query.

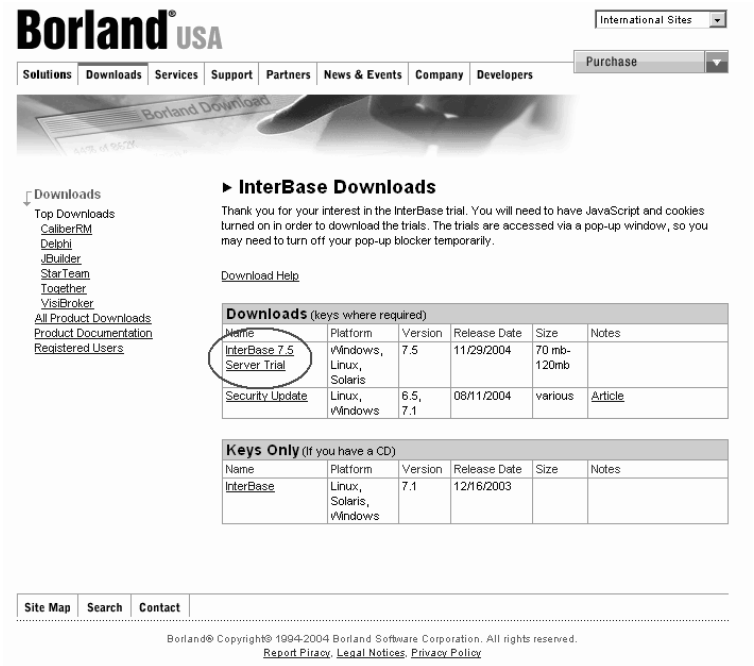
Solution:

- Equip your server with at least 3 GB, and ensure the 3GB switch is set in the Boot.ini. In order to handle this 3 GB address space, it is necessary to use the appropriate Firebird version (when the normal Firebird version is only linked with a different link flag). I think we may be the only company to currently be in possession of such a Firebird version (Paul Reeves performed the linking for us).
- The best solution is however to change to the Firebird Classic Server, together with sufficient RAM and more that one CPU. This certainly puts life back into the database!

1.1.2 Download and Install InterBase®

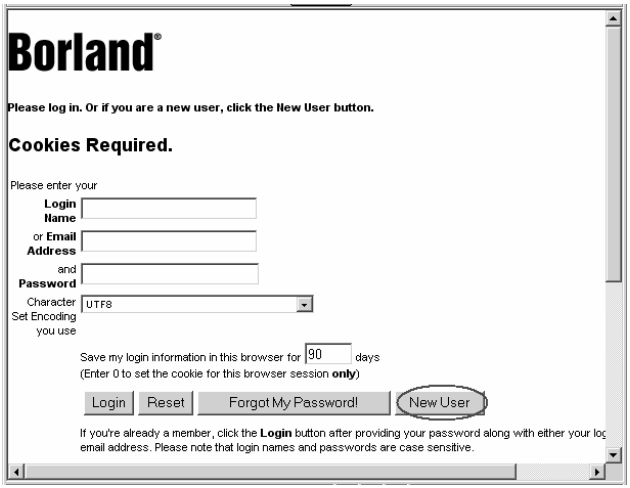
This guide will lead you through the process of downloading and installing the free trial version of InterBase. For those having purchased InterBase®, the installation routine is the same (just skip the download instructions).

The current InterBase® trial version (at the time of writing this it was version 7.5. Please refer to InterBase 7.5 trial version for further information) can be downloaded free of charge from
http://www.borland.com/products/downloads/download_interbase.html.



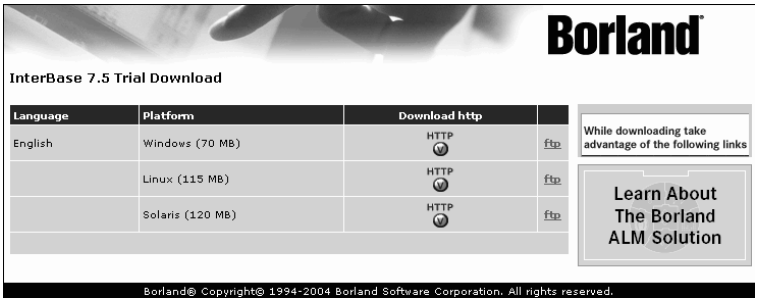
The InterBase 7.5 Release Notes are included in the download file. The full InterBase documentation can be purchased in printed form if wished.

To proceed, click the Trial Download link, and enter your login name/Email and password, if you are already a registered Borland user. If not, the click the New User button:



If not already registered, you will have to register as a user, before proceeding further.

Following login/registration, the download dialog appears:

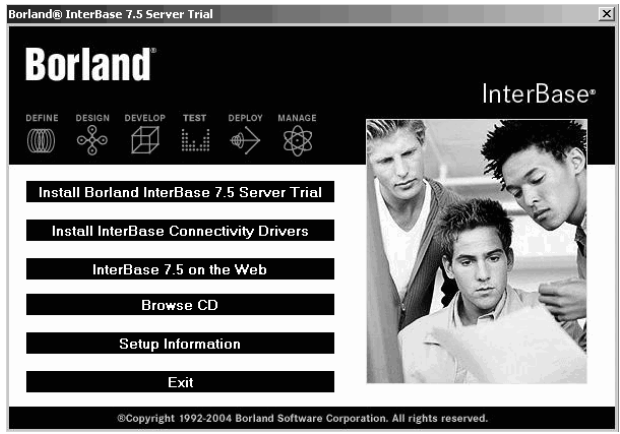


Simply click the button (HTTP or FTP) for the platform required and specify the drive and path on the computer where the InterBase trial version is to be installed.

To complete product registration, open the Borland Product Registration e-mail and save the attached text file to your hard drive in the InterBase home directory before running the InterBase install.

If the e-mail has no attachment, save it as a text or .eml file into the InterBase home directory. For example:
Netscape: Choose File / Save As / File and save it as (`reg999.txt`) into `<interbase_home>`.
Outlook: Choose File / Save As and save it as (`reg999.eml`) into `<interbase_home>`.

Extract the ZIP file (for example in Windows to `C:\Program Files\Interbase\`) and start the relevant `install_[platform].exe` file.



For those installing InterBase for the first time, we recommend reading the *Setup Information* first. As Interbase has introduced a new installer with version 7, the *Setup Information* is also of interest to those upgrading from older versions.

The Install InterBase 7.5 button guides you through the installation: click the *Install Borland InterBase 7.5 Server Trial* button and follow the prompts to accept the license agreement. In the *Choose Install Set* panel, select *Server and Client* or *Client Only*, as appropriate.

Choose an install location and click *Next* to proceed to the review panel. In most cases, the defaults chosen for you by the Install Wizard will be appropriate. If you see anything that you need to change, click the *Previous* button to return to the earlier panels. To install different product components, return to the *Install Set* panel and select *Custom*. This choice lets you select which components to install.

When the *Review* panel displays your desired configuration, click *Install*. The installer completes the product installation and then displays the Registration Wizard.

You should register now because you cannot start the InterBase server until registration is complete. If you have an active internet connection and you are able to specify the drive and path of the registration text file, this should run automatically.

InterBase® 7.5 Trial Version

The trial version of InterBase 7.5 (released November 29, 2004), available for Windows, Linux and Solaris, offers you a chance to evaluate InterBase and see just how easy it is to develop and deploy complex business applications.

New users and current users alike can use this 90 day trial version to evaluate the performance gains resulting from greater SMP scalability and simultaneous memory allocations, improved trigger and stored procedure cache management, enhanced Inter-Client Type 4 JDBC driver, index optimization, along with all the usual advantages of an InterBase® database.

Trial version features:

- All of the functionality of InterBase 7.5 is provided.
- The server can address up to 4 CPUs
- 20 users (80 logical connections) to the server are available.
- The InterBase 7.5 Trial version is available for Windows, Linux and Solaris.
- A limited documentation set (Release Notes in PDF format) is included in the trial version.

Limitations of the trial version:

The 7.5 trial version licensing expires 90 days after the installation and registration of the trial, on all platforms. After that date, the 7.5 trial version will cease to function.

The trial version is self-contained in licenses as stated above. It will not make use of any existing `ib_license.dat` (InterBase licenses) from previous versions of InterBase you may have.

InterBase® 7.5 minimum system requirements

Borland InterBase 7 minimum system requirements:

- 32 MB RAM
- 20 MB hard disk space for install
- CD-ROM drive

Windows®

- Microsoft® Windows® 2000 (SP2), Windows Server™ 2003, Windows NT® 4.0 (SP6a or higher), or Windows XP

Linux®

- Red Hat® Enterprise Linux® and SUSE® LINUX Enterprise Server 9 recommended

Solaris™

- Solaris™ 7, 8, or 9

JDBC™

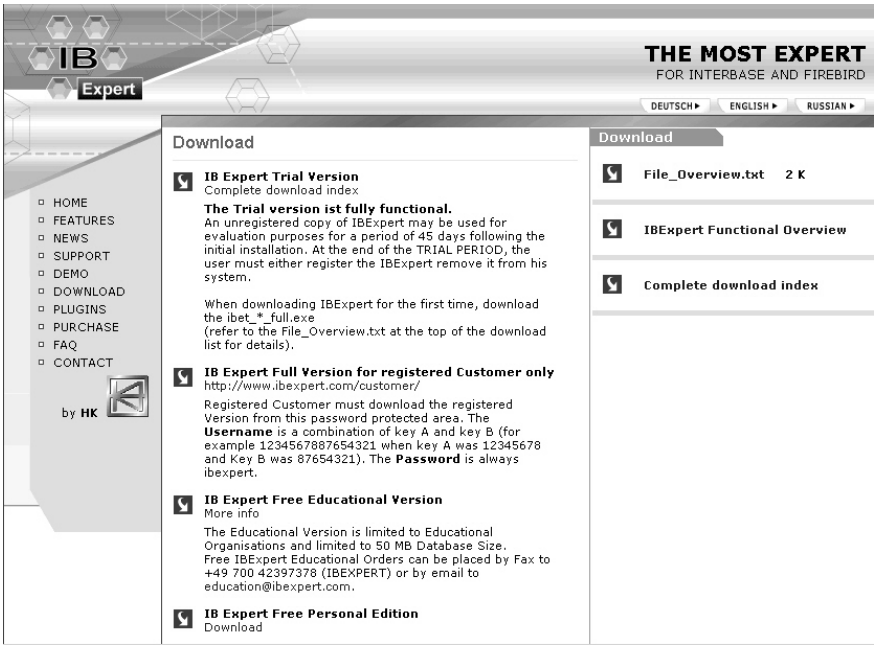
- Development JDK™ 1.2 , 1.3 , 1.4, or J2SE™ 5.0
- Deployment JDK 1.2 , 1.3, 1.4, or J2SE 5.0

Product registration required.

1.1.3 Download and Install IBExpert

IBExpert can be downloaded from http://www.ibexpert.com/. There are a number of versions - please refer to IBExpert License for further information.

The DOWNLOAD page on the IBExpert website offers a number of download options:



Registered customers should click on the http://www.ibexpert.com/customer/ link.

Enter your user name and the password supplied with the registration confirmation. The **Username** is a combination of key A and key B (for example 1234567887654321 when key A is 12345678 and key B is 87654321). The **Password** is always ibexpert.

For those wishing to download the free Personal Edition (for more information please refer to IBExpert Personal Edition), click on the download link at the bottom of the page. For those wishing to download the Trial Version, click on *Complete download index*.

In the Download Area, you will find the following EXE files:

ibet_ = Trial version: these files are fully functional versions in the last stable build. They run for 45 days without any restrictions.

ibep_ = Personal Edition: this edition is totally free of charge. The files are fully functional and temporally unlimited. This version does however have some functional limitations.

In the Password protected Customer Area, you will find the following files:

ibec_ = Customer versions: these files include the unlimited use of the full version. When started for the first time, it is necessary to enter the computer-specific key, provided with the registration form. Customers with site or VAR licenses need to copy the license file into the IBExpert directory before starting IBExpert for the first time, in order to avoid this key request.

All IBExpert versions include one of the above prefixes, followed by the version number (date in reverse order), and the suffix **_full.exe** (this single version replaced the earlier options *full* and *exe* in September 2004*). This also includes all command-line tools.

**Since IBExpert version 2004.9.12.1 all files (IBExpert.stg, IBExpert.tb, IBExpert.scm etc.) are stored in the user home directory (for example, C:\Documents and settings\<User_Name>\Application Data\ HK-Software\IBExpert\). Existing files will be moved to this directory automatically when the new version is started for the first time.*

Double click on the .EXE file to start the installation. IBExpert is then automatically started upon completion.

Should you encounter any problems whilst attempting to download IBExpert, please send an e-mail (in either the English or German language) to register@ibexpert.com, with a detailed error description.

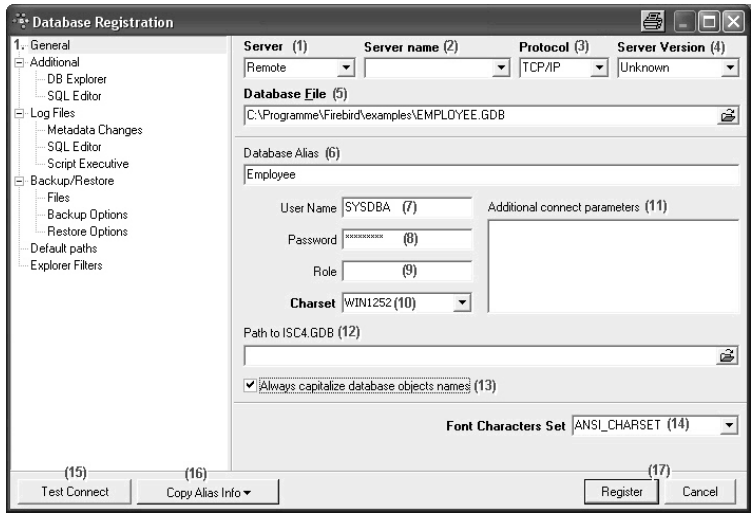
1.1.4 Registering a database (using the EMPLOYEE example)

In order to administrate a database using IBExpert, it is first necessary to register the database. For detailed information regarding database registration, please refer to Register Database.

Here we will briefly show how to register a database, based on the sample EMPLOYEE database supplied with both Firebird and InterBase.

First open the Register Database dialog, using the IBExpert menu item Database / Register Database, right-clicking in the DB Explorer (left-hand panel) and selecting the Register Database menu item, or using the key shortcut [Shift + Alt + R].

The Register Database dialog appears:



(1) Server: first the server storing the database needs to be specified. This can be local (localhost) or remote (see Create Database).

By specifying a local server, fields **(2)** and **(3)** are automatically blended out, as they are in this case irrelevant.

(2) Server name: must be known when accessing remotely. The standard port for InterBase and Firebird is 3050. However this is sometimes altered for obvious reasons of security, or when other databases are already using this port. If a different port is to be used for the InterBase/Firebird connection, the port number needs to be included as part of the server name (parameter is server/port). For example, if port number 3055 is to be used, the server name is SERVER/3055. This is sometimes the case when a Firewall or a proxy server is used, or when another program uses the standard port. For using an alias path for a remote connection, please refer to the article remote database connect using an alias.

(3) Protocol: a pull-down list of three options: TCP/IP, NetBEUI or SPX. TCP/IP is the worldwide standard (please refer to Register Database for more information).

(4) Server versions: this enables a server version to be specified as standard/default from the pull-down list of options. This is necessary for various internal lists. For example, possible key words can be limited this way.

(5) Database File: by clicking on the folder icon to the right of this field, the path can easily be found and specified and the database name and physical path entered. For example for Firebird:

```
C:\Programs\Firebird\Firebird_1_5\examples\EMPLOYEE.FDB
```

for InterBase:

```
C:\Programs\Interbase\examples\EMPLOYEE.GDB
```

If no database alias has been specified, the database name must always be specified with the drive and path. Please note that the database file for a Windows server must be on a physical drive on the server, because InterBase/Firebird does not support databases on mapped drive letters.

(6) Database Alias: descriptive name for the database (does not have to conform to any norms, but is rather a logical name). The actual database name and server path and drive information are hidden behind this simple alias name - aiding security, as users only need to be informed of the alias name and not the real location of the database. For example:

```
Employee
```

(7) User Name: the database owner (i.e. the creator of the database) or SYSDBA.

(8) Password: if this field is left empty, the password needs to be entered each time the database is opened. Please refer to Database Login for further information. The default password for SYSDBA is *masterkey*. Although this may be used to create and register a database, it is recommended - for security reasons - that this password be changed at the earliest opportunity.

(9) Role: an alternative to **(7)** and **(8)**;can initially be left empty.

(10) Charset (abbreviation for Character Set): Here the default character set can be specified. This is useful when the database is designed to be used for foreign languages, as this character set is applicable for all areas of the database unless overridden by the domain or field definition. If not specified, the parameter defaults to `NONE`, i.e. values are stored exactly as typed. For more information regarding this subject, please refer to Charset/Default Character Set. If a character set was not defined when creating the database, it should not be used here.

(11) Additional connect parameters: input field for additional specifications. For example, system objects such as system tables and system-generated domains and triggers can be specified here. They will then automatically be loaded into the DB Explorer when opening the database alias.

(12) Path to ISC4.GDB: This can be found in the InterBase or Firebird main directory. This database holds a list of all registered users with their encrypted passwords, who are allowed to access this SERVER.

When creating new users in earlier InterBase versions (<6), IBExpert needs to be told where the ISC4.GDB can be found. Since InterBase version 6 or Firebird 1 there is a services API. So those working with newer versions may ignore this field!

(13) Always capitalize database objects' names (checkbox): this is important as in SQL Dialect 3 entries can be written in upper or lower case (conforming to the SQL 92 standard). InterBase however accepts such words as written in lower case, but does not recognize them when written in upper case. It is therefore recommended this always be activated.

(14) Font character set: this is only for the IBExpert interface display. It depends on the Windows language. If an ANSI-compatible language is being used, then the ANSI_CHARSET should be specified.

(15) Test connect: the Comdiag dialog appears with a message stating that everything works fine, or an error message - please refer to the IBExpert Services menu item, Communication Diagnostics for more details.

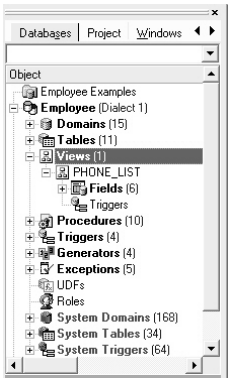
(16) Copy Alias Info: alias information from other existing registered databases can be used here as a basis for the current database. Simply click on the button and select the registered database which is to be used as the alias.

(17) Register or Cancel: after working through these options, the database can be registered or cancelled.

Details of further options (listed in the left-hand panel in the Register Database dialog) may be found under Register Database (individual subjects are listed in the upper gray panel in the online documentation). These are not compulsory, and may be altered at a later date, if wished, using the Database / Database Registration Info menu item.

1.1.5 Working with a database

A registered database can be connected simply by double-clicking on the database name in the DB Explorer. Alternatively use the IBExpert menu item Database / Connect to Database, click the Connect Database icon in the toolbar, or use the key shortcut [Shift + Ctrl + C]. The database and its objects appear in a tree form in the DB Explorer:



For information with regard to the details displayed in the DB Explorer, please refer to Register Database / Additional and Options / Environment Options / Tools for alternatives regarding the DB Explorer.

The individual database objects may be opened by double-clicking on the object name. For further information about the individual objects, please refer to Database Objects.

For further information regarding IBExpert navigation, please refer to IBExpert Screen. Options and templates may be specified and adapted using the IBExpert Options menu. Other important IBExpert features can be found in the IBExpert Tools menu and IBExpert Services menu.

The IBExpert online documentation provides not only a comprehensive documentation for using IBExpert, but also offers many tips for those new to database development. The online documentation can be viewed under http://www.ibexpert.info/documentation or alternatively individual subjects may be viewed context-sensitively, using the [F1] key from any IBExpert dialog or the DB Explorer. The documentation includes an index, search function, and even a "What's New" function, whereby you can select, for example, all new articles written after a certain date. Or you can download the complete documentation files onto your hard drive.

And if you can't find an answer to your problem there, please mail us at documentation@ibexpert.com.

1.2 What is IBExpert?

Have you taken a look at the IBExpert Guided Tour yet? For a brief overview demonstrating many of the main IBExpert features, please refer to http://www.ibexpert.com/DEMO page, IBExpert Guided Tour (published December 2003. Therefore some of the more recent IBExpert features are not included in this tour).

IBExpert is a professional Integrated Development Environment (IDE) for the development and administration of InterBase and Firebird databases. Whether you enjoy the control of hand-coding DML or DDL statements or working in a visual editing environment, IBExpert makes it easy to get started and provides you with vital tools to speed and enhance your work.

IBExpert includes many coding tools and features: visual editors for all Database Objects, an SQL Editor and Script Executive, a Debugger for stored procedures and triggers, a Query Builder, a powerful Database Designer and much, much more...

IBExpert's visual editing features allow even the total beginner to quickly create a database and add database objects without writing a single line of code. You can view, navigate and work on all your database objects in the IBExpert DB Explorer.

Features list:

SQL

- Supports InterBase 4-7.x and Firebird 1.x
- Hyperlinks in all Editors
- SQL Editor History
- SQL Monitor
- Visual Query Builder
- Export of results in several formats
- Background execution in own thread
- Direct CSV file data import using SQL statement
- Insert Into with automatic creation of target table
- Insert Into also into another database
- Automatic display of parameters with history

Editors

- Code Completion
- Customizable Keyboard Templates
- Editors for all Database Objects
- Stored Procedure and Trigger Debugger including Trace Into, conditional Breakpoints, and much more
- Blob Editor with text, RTF, image and Hex display options
- Autoinc Assistant for Trigger, Generator and Stored Procedure creation
- Alteration of field sequence in tables as wished
- Parser for Stored Procedures and Triggers with warnings and tips (e.g. unused variables, etc.)

Security

- Special Editors for users, groups and rights
- Autogrant for automatic assignment of rights for new objects

Data Definition

- Entity Relation Database Designer including layers, subject areas, autoroute, reverse engineering and much more
- Assistant for Dependencies Analysis
- Database Comparer for comparison of two database structures, including creation of update script

Performance

- Performance Analysis
- Plan Analysis
- Global Stored Procedure/Trigger operations index analysis
- Metadata cache for speedier opening of database connections via slow connections

Scripts

- Own Editor for SQL script execution
- Parser for Scripter with tree display of those objects created by the script
- Metadata Extract for the creation of an empty database copy with identical structure
- Data extract by creation of SQL scripts
- Inclusion of blob data in SQL scripts
- Creation of complete scripts including both metadata and data for database restoration

Reports

- User-defined reports
- Ready-made reports for database documentation
- All reports can be saved as PDF, Word, and other formats
- Reports can be stored as a file or in the database
- HTML report database documentation

Wizards

- Backup and Restore with ability to store options
- Database Statistics summary
- Full-text search in metadata

Tools

- Data Analysis - an OLAP and data warehouse tool for analyzing data
- Integrated Communication Diagnostics
- SIUD Assistant for creation of Select, Insert, Update and Delete procedures
- Assistant for the creation of procedures and views from any SQL statements
- Test Data Generator
- Database monitoring for InterBase 7.x
- Integrated Bug Tracking System
- IBExpert Direct for information about the newest versions

Specials

- User-defined project views for customized structure of database objects
- Stored Procedure and Trigger Version Control System VCS
- Log Assistant for constructing Data Change protocols
- Customizable templates for all automatically created objects
- User Interface offers MDI or SDI application options
- User-definable toolbars, colors and fonts
- Open PlugIn interface with Delphi source code sample
- Integration of external programs into the Tools menu
- Assistant for global updating of index statistics

- Assistant for compilation of all stored procedures and triggers
- Assistant for deactivation and reactivation of all stored procedures and triggers
- Multilingual

Educational Version

- Includes all the functionality of the full version; the database size is, however, limited to 50MB

1.3 IBExpert License

The IBExpert License Agreement can be found and read under IBExpert Help / About. The various licensing options are listed briefly here. For more details and email contact for queries, please refer to http://www.ibexpert.com.

Single, Additional, Site or VAR License?

If you purchase the Site License, we will create a License File for all computers used in your company. The Single License is only for one computer.

The IBExpert VAR (Value Added Reseller) Version is for software companies, who wish to bundle IBExpert with their software. Here we offer a new model to integrate IBExpert. We create a license file for your company and you can install IBExpert everywhere, where your software is installed.

Updates

All products include all new versions for 12 months. This includes both major and minor updates. At the end of this period you can purchase Extension Products for downloading all new versions for the next 24 months.

Upgrades

If you want to upgrade from any Single, Additional or Starter Pack combination to a Site or VAR license, you will receive access to all new versions for 24 months.

Additional License for the computer in the home office?

We have had a lot of queries regarding the licensing model of IBExpert. Here is our offer to users, who work on two computers (one in the office and the other at home). If you do not work on both computers at the same time, you can order a free License Key for your home office computer, providing you have purchased a license for the computer in the office.

Simply send an e-mail to register@ibexpert.com quoting the computer name of your home computer or laptop, your company name and the current registered license ID of your work computer.

New! Free Version for Educational Purposes!

As we've had a lot of queries from students, who want to use IBExpert as part of a course at a university or school, we have decided to create a special version for this.

The only limitation is that a database is restricted to a maximum of 50 MB, when working with this version. There are no further constraints, such as a time limit, so you can use it for studying SQL Language (DDL, DML), Transactions, Stored Procedures, Triggers, Entity Relationship Modeling, Reporting, Optimizing and much more.

In addition to the free database InterBase or Firebird, you get a unique opportunity to learn professional database programming at no costs.

How to get the free IBExpert Educational Version?

We need an e-mail or fax order, where you describe the institution where IBExpert is to be used (University, School, ...). We also need a brief description of the course in which you plan to use it. You gain access to our Educational Area as soon as we receive your order.

The email with the Download Info will only be sent to an e-mail in a University/School domain. Unfortunately we will not be able to send a license if we do not receive this information.

Free IBExpert Educational Orders can be placed by fax to +49 700 42397378 (IBEXPERT) or by email to education@ibexpert.com.

How to get the free IBExpert Personal Edition?

You can download it from http://www.ibexpert.com/download.

Further information can be found at http://www.ibexpert.com/.

1.3.1 IBExpert Personal Edition

The IBExpert Personal Edition is a free version, offering new users the chance to get acquainted with IBExpert at their own pace. It is however somewhat limited in its functionality, and does not include the following features:

- Data Analysis
- Database Designer
- SP/Triggers Debugger
- Visual Query Builder
- Report Manager
- Test Data Generator
- Blob Editor
- Grant Manager
- SP/Triggers/Views Analyzer
- Database Comparer
- Log Manager
- Table Data Comparer
- some other features such as autogranting privileges, recomputing selectivity of all indices etc.

To view and test these features it is necessary to download the trial version.

1.4 How to register IBExpert

If you are reading this, you have probably already installed the IBExpert trial version. We hope you will take the time to test IBExpert thoroughly during the next forty-five days; you will soon see how quick and easy it is to develop and work with InterBase or Firebird using IBExpert, even if you have no previous experience of database development.

Following the forty-five day trial period, you are required to either remove the program from your hard drive or to register IBExpert.

How to register:

All prices can be found in our Shop at https://secure.element5.com/shareit/product.html?productid=153856.

Don´t forget to fill out all fields. The data is transmitted using a secure connection. Payment can also be made by invoice and remittance to our bank account, but in this case, the keys will not be sent until after receipt of payment. For details contact support@ibexpert.com.

License Key:

Your license key will be generated and sent to you by e-mail on the working day following confirmation of payment. Please remember, the license is generated for the computer name(s) entered in our shop. For the VAR and site licenses we do not need computer names.

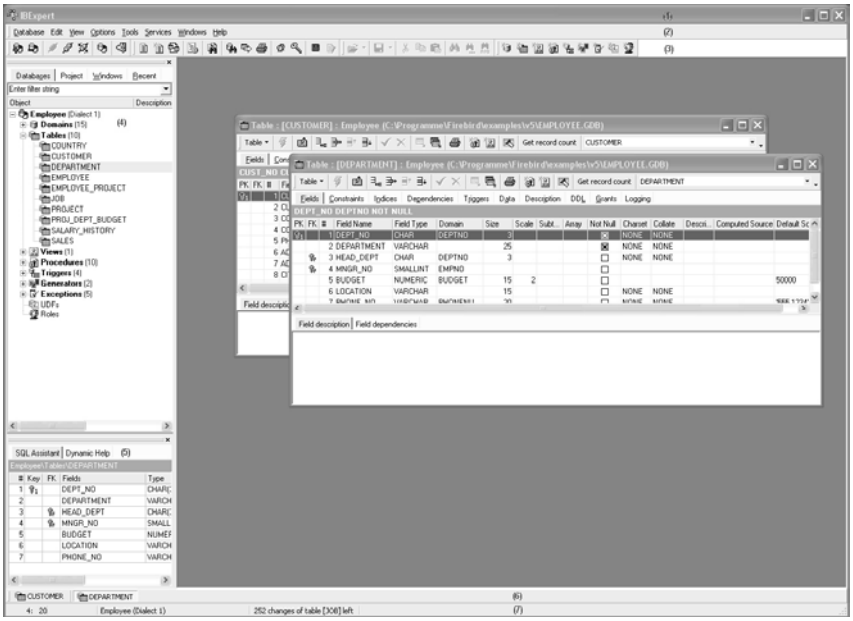
Computer Name:

To determine your computer name in Windows, go to the Windows "My Computer" and select the right-click menu item "System Properties". On the second page you will find the computer name ("Full Computer Name").

If you have to change the computer name for this license, please send us the new computer name, not forgetting to write the License ID on the form, that you received with the first registration.

1.5 IBExpert Screen

When IBExpert is started, the standard IBExpert screen appears as follows:



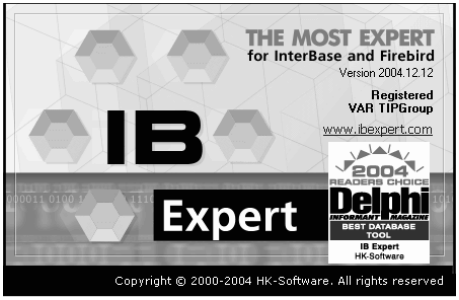
The standard IBExpert settings display a large working window, with the Menu and toolbar at the top of the screen, a windows bar and status bar at the bottom, and the Database Explorer on the left, divided from the SQL Assistant (lower left) by a splitter.

The View menu can be used to blend the DB Explorer, status bar, windows bar and toolbars in or out.

Further visual options can be specified by the user in the IBExpert Options menu.

1.5.1 IBExpert Splash Screen

The IBExpert splash screen appears when IBExpert is started. It displays the IBExpert logo and version number.



The splash screen may be disabled if wished, by checking the "Don't Show Splash Screen" option, found under Options / Environment Options on the initial Preferences page.

1.5.2 (1) Title Bar

The title bar is the blue horizontal bar at the top of the main IBExpert screen, and at the top of all IBExpert editors. It displays the program or editor name on the left, and in the right hand corner there are four small icons (from left to right):

- Print (only on the IBExpert screen with the MDI Interface; with the SDI Interface it appears on the active window/editor)
- Minimize IBExpert / Editor window
- Maximize IBExpert / Editor window
- Exit IBExpert / Exit Editor

1.5.3 (2) Menu

The IBExpert menu bar can be found at the top of the screen:

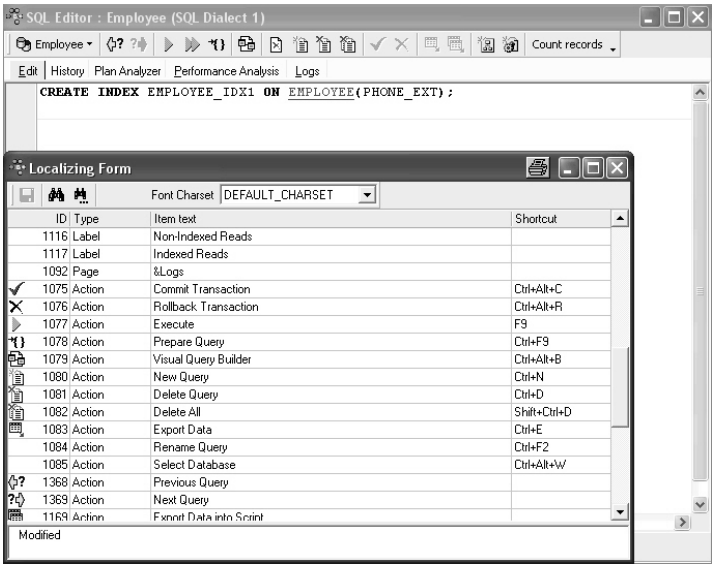
The individual menu headings conceal drop-down lists, opened simply by clicking on one of the words with the mouse or by using [Alt + {underlined letter}], e.g. the Database menu can be started by clicking with the mouse on the word database, or by using the key combination [Alt + D].

The most frequently-used menu items can also be found in the toolbars, represented as icons, or using the right mouse button in either the DB Explorer or the main editors. Alternatively keyboard shortcuts can also be used.

Shortcuts (Localizing Form)

Many menu items can also be executed using so-called keyboard shortcuts (a combination of keys). Where available, these are listed to the right of the menu item name in the menus, and when the cursor is placed over a toolbar icon.

[Ctrl + Shift + Alt + L] works in almost all IBExpert forms and calls the Localizing Form, where you can refer to a complete list of all available shortcuts relevant to the active dialog.



These can be changed by the user as wished.

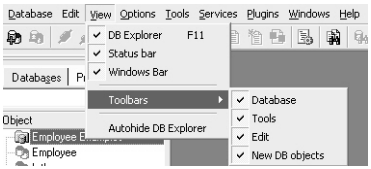
1.5.4 (3) Toolbars

The toolbar is a row of symbols (called icons), representing different menu items. By clicking on an icon with the mouse, a pre-defined menu item is executed. This shortcut is ideal for those operations performed often, as they save the necessity of repeatedly searching through the main menus.

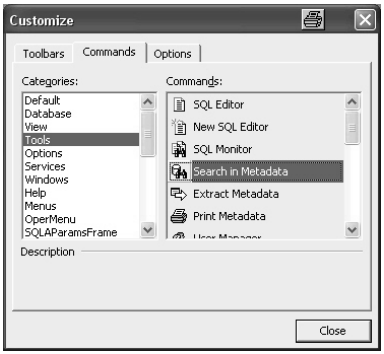
Toolbars can be found in IBExpert in the main window and in the main editors. As with most Windows applications the toolbars are positioned as standard in a horizontal row directly below the main menu in the upper part of the window, or in the upper part of the dialogs. They can however be positioned as wished within the window (main or dialog) using drag 'n' drop.

When the cursor is placed over an icon the respective menu command and keyboard shortcut are displayed.

The user can specify which toolbars he wishes to be displayed in the main IBExpert window using the menu item View / Toolbars.



The individual icons can be specified using the Customize... menu item, opened by holding the mouse over the toolbar and right-clicking.

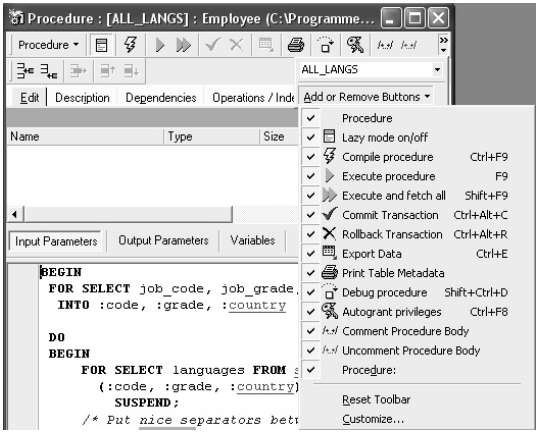


The Customize Tools page displays a list of the toolbar options available. User-defined toolbars can be created here if wished, or reset to the original IBExpert toolbar.

The Command page enables the different menu options listed under Categories to be selected, and the icons (in the right-hand list) added or removed to toolbars using drag 'n' drop.

The Options page allows certain menu and icon options to be checked if wished.

The Editor toolbars can be customized by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list, or using the above method by selecting the last menu item Customize...



Icon

Icons are a principal feature of graphical user interfaces. An icon is a small, square graphical symbol. Each icon represents a menu item, the description of which appears, when the mouse is held over it. Icons can be used as shortcuts by those users who work mainly with a mouse (as opposed to the keyboard).

Icons are usually grouped together in a toolbar, which offers a series of symbols all relating to a certain subject, e.g. new database object, grants etc.

Toolbar Database

This standard toolbar can be viewed in the main IBExpert window. It can be blended in and out using the IBExpert View Menu / Toolbar (check boxes).

The icons (from left to right) can be used to execute the following operations:

- Register Database (Shift + Alt + R)
- Unregister Database (Shift + Alt + U)
- Connect to Database (Shift + Ctrl + C)
- Disconnect from Database (Shift + Ctrl + D)
- Reconnect to Database
- Create Database
- Exit (Alt + F4)

These items can also be found in the main IBExpert Database menu. To alter, customize or reset this toolbar, please refer to Toolbar.

Toolbar Edit

This standard toolbar can be viewed in the main IBExpert window. It can be blended in and out using the IBExpert Menu View / Toolbar (check boxes).

The icons (from left to right) can be used to execute the following operations:

- Load from File (Ctrl + L). The downward arrow produces a pull-down list of the most recent files.
- Save to File (Ctrl + S). The downward arrow produces a pull-down list of the most recent files.
- Cut (Ctrl + X)
- Copy (Ctrl + C)
- Paste (Ctrl + V)
- Find (Ctrl + F)
- Search again (F3)
- Replace (Ctrl + R)

These items can also be found in the main IBExpert Edit menu.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Tools

This standard toolbar can be viewed in the main IBExpert window. It can be blended in and out using the IBExpert Menu View / Toolbar (check boxes).

The icons (from left to right) can be used to execute the following operations:

- SQL Editor (F12)
- New SQL Editor (Shift + F12)
- Query Builder
- Script Executive (Ctrl + F12)
- SQL Monitor (Ctrl + M)
- Search in Metadata (Shift + Alt + F)
- Extract Metadata
- Print Metadata
- User Manager
- Grant Manager
- Report Manager
- Blob Viewer/Editor

These items can also be found in the main IBExpert Tools menu.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar New Database Object

This standard toolbar can be viewed in the main IBExpert window. It can be blended in and out using the IBExpert Menu View / Toolbar (check boxes).

The icons (from left to right) can be used to execute the following operations:

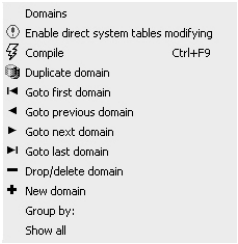
- New Domain
- New Table
- New View
- New Procedure
- New Trigger
- New Generator
- New Exception
- New UDF
- New Role

These items can also be found in the main IBExpert Database menu, or in the IBExpert DB Explorer by clicking the right mouse key to offer a context-sensitive option for the selected database object. Alternatively [Ctrl + N] can be used in the DB Explorer to create new objects (providing an object type has been selected).

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Domain Editor

The standard toolbar for the Domain Editor includes the following icons:

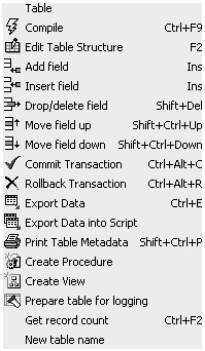


These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Table Editor

The standard toolbar for the Table Editor includes the following icons:

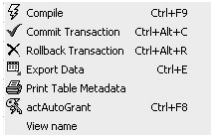


These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar View Editor

The standard toolbar for the View Editor includes the following icons:

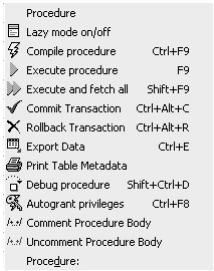


These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Procedure Editor

The standard toolbar for the Procedure Editor includes the following icons:

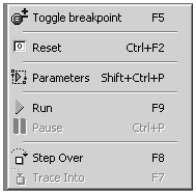


These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Debug Procedure

The toolbar for the Debug Procedure Editor includes the following icons:

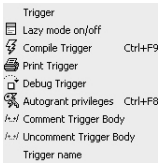


These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Trigger Editor

The standard toolbar for the Trigger Editor includes the following icons:

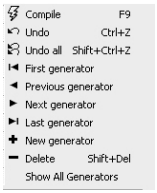


These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Generator Editor

The standard toolbar for the Generator Editor includes the following icons:



These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Exception Editor

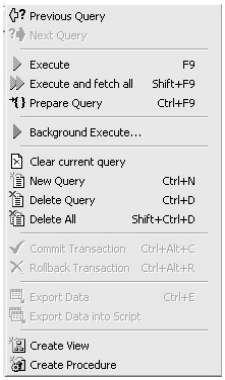
The standard toolbar for the Exception Editor includes the following icons:

These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar SQL Editor

This toolbar can be viewed in the Tools / SQL Editor dialog and includes the following icons:



These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

New to IBExpert version 2.5.0.61 is the added possibility to quickly change the "Transaction Isolation Level" (TIL) for a separate SQL Editor. There is a corresponding button on the SQL Editor toolbar which allows you to choose one of the following isolation levels: Snapshot, Read committed, Read-only table stability, Read-write table stability:

Toolbar Navigation

The navigational toolbar can be found on the Table Editor's Data page, the View Editor's Data page and in the SQL Editor on the Results page and includes the following icons:

The icons (from left to right) can be used to execute the following operations:

- Apply filter
- Show Filter Panel (Ctrl + Alt + F)
- Quick Add Filter Criteria
- Record Number
- Data Analysis (new to IBExpert version 2004.10.25.1)
- Show summary footer (new to IBExpert version 2004.8.5.1)

- Display data as Unicode [F3] (new to IBExpert version 2004.8.26.1)
- First
- Previous
- Next
- Last
- Insert
- Delete
- Edit
- Save Updates
- Cancel Updates
- Refresh

To the right the number of records fetched is displayed.

Toolbar Filter Panel

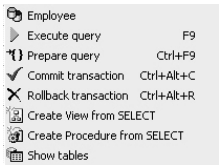
The navigational toolbar can be found on the Table Editor's Data page, the View Editor's Data page and in the SQL Editor on the Results page when the Show Filter Panel is activated, and includes the following icons:

The icons (from left to right) can be used to execute the following operations:

- Apply Filter
- Add New Criteria (Ins)
- Delete Criteria (Ctrl + Del)
- Vertical Layout (Shift + Ctrl + L)
- Count Records

Toolbar SQL Query Builder (Visual Query Builder)

This toolbar can be viewed in the Tools / SQL Query Builder dialog and includes the following icons:



These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Data Analysis (PivotCube Form)

This toolbar can be viewed in the IBExpert Tools / Data Analysis dialog and includes the following icons:

The icons (from left to right) can be used to execute the following operations:

- Load cube from file [Ctrl + L]
- Save cube to file (Ctrl + S]
- Build cube
- Abort cube building
- Toggle toolbars on/off (applies to the cube dimensions, columns and measures toolbars)
- Cube Manager
- Print [Ctrl + P]
- Export data (to Excel (OLE), HTML or metafile) [Ctrl + E]
- Dataset (pull-down list)

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Script Executive

This toolbar can be viewed in the Tools / Script Executive dialog and includes the following icons:

The icons (from left to right) can be used to execute the following operations:

- Pull-down menu detailing the most important operations
- Select database including a pull-down list of available databases
- Load from file (Ctrl + L) including a pull-down list of recent files
- Save to file (Ctrl + S) including a pull-down list of recent files
- Show script explorer (blends the script explorer on the left-hand side in and out)
- Run script (F9)
- Stop script
- Use current connect (toggle)

To alter, customize or reset this toolbar, please refer to Toolbar.

Toolbar Dependencies Viewer

This toolbar can be viewed in the Tools / Dependencies Viewer dialog and includes the following icons:

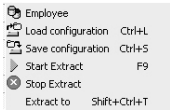
- Refresh
- Clear All
- Print
- Stop
- Don't check domain dependencies (checkbox)

These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Extract Metadata

This toolbar can be viewed in the Tools / Extract Metadata dialog and includes the following icons:



These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Meta Objects

This toolbar can be viewed in the Tools / Extract Metadata dialog under the Meta Objects tab and includes the following icons:

The icons (from left to right) can be used to execute the following operations:

- Domains (Ctrl + Alt + D)
- Tables (Ctrl + Alt + T)
- Views (Ctrl + Alt + V)
- Procedures (Ctrl + Alt + P)
- Triggers (Ctrl + Alt + R)
- Generators (Ctrl + Alt + G)
- Exceptions (Ctrl + Alt + E)
- UDFs (Ctrl + Alt + U)
- Roles (Ctrl + Alt + L)
- Extract all (checkbox)

Toolbar Print Metadata

This toolbar can be viewed in the Tools / Print Metadata dialog and includes the following icons:

The icons (from left to right) can be used to execute the following operations:

- Select database including a pull-down list of available databases.

- Show Domains
- Show Tables
- Show Views
- Show Procedures
- Show Triggers
- Show Exceptions
- Show User-defined functions
- Preview
- Print

To alter, customize or reset this toolbar, please refer to Toolbar.

Toolbar Grant Manager

This toolbar can be viewed in the Tools / Grant Manager dialog and includes the following icons:

These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Grants

This toolbar can be viewed in the Tools / Grant Manager dialog under "Grants on", as well as in the Table Editor under the Grants tab, and includes the following icons:

The icons (from left to right) can be used to execute the following operations:

- Grant All
- Grant All with GRANT OPTION
- Grant to All with GRANT OPTION
- Grant to All
- Grant All to All
- Revoke All
- Revoke from All
- Revoke All from All

Toolbar Localize IB Messages

This toolbar can be viewed in the Tools / Localize IB Messages dialog and includes the following icons:

The icons (from left to right) can be used to execute the following operations:

- Load from File
- Save to File
- Undo
- Goto Message Number
- Find
- Search Again
- Export to Text File
- Import from Text File

Toolbar Localize IBExpert

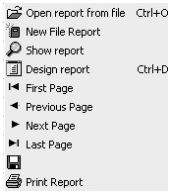
This toolbar can be viewed in the Tools / Localize IBExpert dialog and includes the following icons:

The icons (from left to right) can be used to execute the following operations:

- Save to File
- Find
- Search Again
- Export to Text File
- Import from Text File
- Font Charset (pull-down list)

Toolbar Report Manager

This toolbar can be viewed in the Tools / Report Manager dialog and includes the following icons:



These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Blob Viewer/Editor

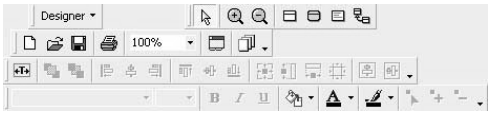
This toolbar can be viewed in the Tools / Blob Viewer/Editor dialog and includes the following icons:

These can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset this toolbar, please refer to Toolbar.

Toolbars Database Designer

These toolbars can be viewed in the Tools / Database Designer dialog. They comprise 4 individual toolbars and include the following icons:



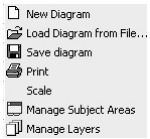
The individual menus are as follows:

I. Menu and Palette:

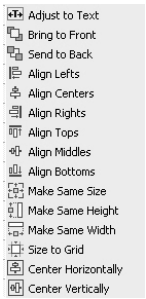
The icons (from left to right) can be used to carry out the following operations:

- Pointer
- Zoom in
- Zoom out
- Table
- New View
- Comment Box
- Reference

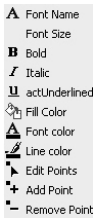
II. Main:



III. Layout:



IV. Font / Colors:



The icons displayed in the Main, Layout and Font Colors toolbars can be blended in and out by clicking the downward arrow to the right of the toolbar, and using the menu item Add or Remove Buttons to check the relevant icons in the menu list.

To customize or reset these toolbars, please refer to Toolbar.

Toolbar Server Properties/Log

This toolbar can be viewed in the Services / Server Properties/Log dialog and includes the following icons:

- Select server (pull-down list of available servers)
- Retrieve
- Preview Log Report
- Print

To customize or reset this toolbar, please refer to Toolbar.

Toolbar Database Statistics

This toolbar can be viewed in the Services / Database Statistics dialog and includes the following icons:

- Select Database (pull-down list of available databases)
- Analyze from File

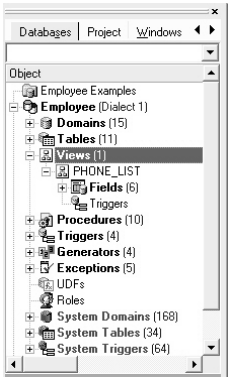
- Retrieve Statistic
- Preview Log Report
- Print
- Export

To customize or reset this toolbar, please refer to Toolbar.

1.5.5 (4) DB Explorer

The IBExpert Database Explorer is a navigator which considerably simplifies the work with InterBase/Firebird databases and the database objects.

The Database Folder displays all registered databases at a glance. A database connection can be made simply by double-clicking on the database name.



Each connected database is displayed in a logical tree form, including a list of all the database objects created in this database. If the database contains objects of some of these types, the name of the respective object branch appears in bold. The blue number in brackets behind the object caption shows the number of objects already created for this database.

When a database, the object captions or the objects themselves are highlighted, the DB Explorer menu can be opened by right-clicking the mouse.

In IBExpert version 2004.9.12.1 a separate node was added for database indices. It is also possible to display system indices (indices for system tables). Use the IBExpert menu item Database Registration Info / DB Explorer / Additional / Show System Indices to enable/disable the display of system indices.

In IBExpert version 2004.12.12.1 support for InterBase 7.5 embedded user authentication was added. There is now a separate node for embedded users in the Database Explorer. It is possible to create, alter and delete embedded users using the DB Explorer context menu.

Using the control panel and right mouse button many basic metadata and data operations can be performed directly from the DB Explorer, such as creating, editing and dropping a database and its objects. In IBExpert version 2004.12.12.1 the option to

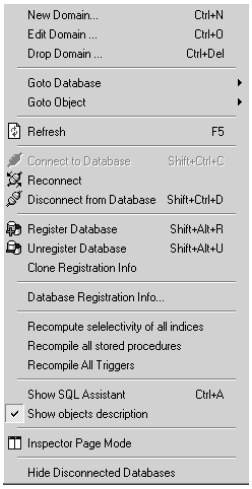
activate/deactivate only selected procedures/triggers was added. Just select the required SP/triggers holding the [Ctrl] or [Shift] keys and choose the *Deactivate/Activate* item in the DB Explorer context menu.

Detailed information regarding the highlighted database object can be viewed in the SQL Assistant (below the DB Explorer).

The object tree branches can be expanded or reduced by double-clicking the object heading or clicking the '+' or '-' sign to the left of these headings (alternatively use the '+' and '-' keys to open a highlighted object heading). The individual objects themselves can be opened with a double-click or by pressing the Enter key.

The object description can be seen to the right of the object name, provided a description was inserted at the time of creation, and providing the DB Explorer is opened wide enough (the width of the DB Explorer can be expanded or reduced by dragging the right-hand splitter (i.e. the divider between the DB Explorer and the Main Window) with the mouse).

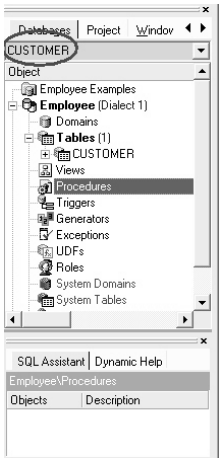
Should you experience any problems with double-click expanding, or your object descriptions are not displayed at all, please check the IBExpert Options menu item Environment Options under the branch, Tools / DB Explorer, to ensure that these options have been checked. It is also possible to specify color display here for system objects, the Database Folder and inactive triggers.



The text input field at the top of the DB Explorer (directly underneath the tabs) can be used to filter object names, e.g. to search for an object, EMP, simply type EMP. If EMP* or EMP% is typed, IBExpert displays all objects beginning with EMP; for an object ending in EMP, type *EMP or %EMP. To display objects which have a substring in their name, it is necessary to type *EMP* or %EMP%.

It is also possible to use ? For example, to display objects whose names start with EMP and are exactly 6 symbols in length. In case type EMP???. Regular expressions are, of course, also allowed.

Please note that this option does not, however, search for individual fields - if this is required, use the IBExpert Tools menu item Search in Metadata.



Certain display default filters can also be defined, under Register Database / Explorer Filters. And under Database Registration Info or Register Database, system tables, system generated domains and triggers and object details (fields, triggers etc. relating to a specific object) can be displayed or blended out as wished, by clicking on the Additional / DB Explorer branches.

The DB Explorer includes the following tabs:

- Database Folder (described above)
- Project View
- Diagrams (only visible when the Database Designer is in use)
- Windows Manager
- Recent List

[F11] blends the DB Explorer in and out. And please also refer to the IBExpert Menu item View / Autohide DB Explorer. This option namely enables the DB Explorer to disappear automatically when any editor is opened - allowing a larger working area. It is blended back into view simply by holding the mouse over the left-hand side of the IBExpert main window.

Objects may be dragged 'n' dropped from the DB Explorer and SQL Assistant into many of the IBExpert Tools and Services code editor windows, for example, the SQL Editor and Query Builder. Since version 2004.2.26.1 this has been greatly improved: when an object node(s) is dragged from the DB Explorer or SQL Assistant, IBExpert will offer various relevant versions of text to be inserted into the code editor.

Since IBExpert version 2004.8.5.1 it is possible to store server info (server type, server name, server version, connection protocol) and client library name for database folders.

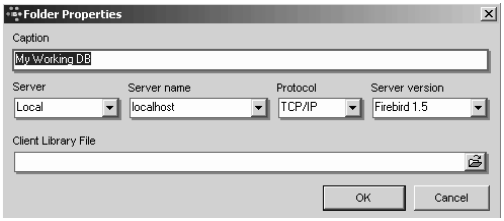
Database Folder

The DB Explorer Database Folder can be used to specify a selection of databases as wished, so that it is not necessary to search through all available databases each time a specific database is required. The database folder allows a hierarchical classification of the Database Registration. This is for example useful for system vendors with many customers and databases, and simplifies, for example, the logging in to customer databases via a router.

When a database is registered, it is automatically displayed here in the folder list. Connected databases are displayed in bold, disconnected in normal type. Please note: it is possible to blend out all unconnected databases using the DB Explorer right-click menu item, Hide Disconnected Databases.

A new database folder can be created in the DB Explorer by highlighting the connected database for which a folder is to be created, right-clicking and selecting New Database Folder ... (or [Ctrl + N]).

It is then possible to rename the database folder, by selecting the folder and using the right-click context-sensitive menu or [Ctrl + O]:

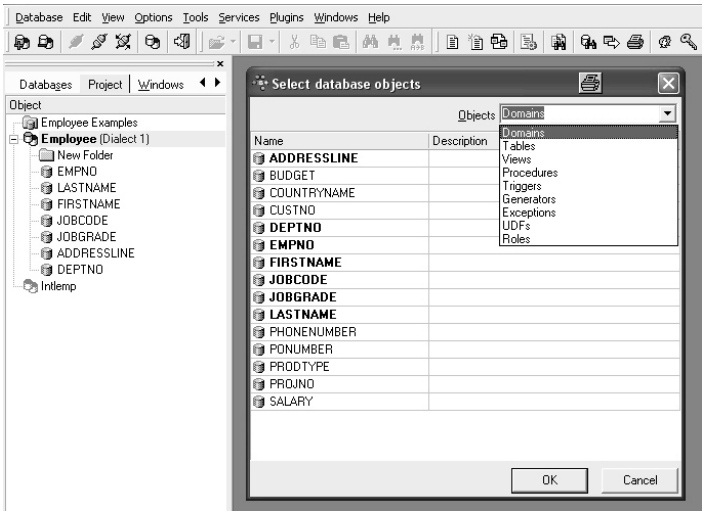


Since IBExpert version 2004.8.5.1 it is also possible to store server information (server type, server name, server version, connection protocol) and client library name for database folders.

A folder can also be deleted (again, using the right-click menu or [Ctrl + Del]). Please be careful when using this delete command, as IBExpert does not ask for confirmation before deleting the folder!

Project View

In the DB Explorer, projects can be defined to streamline the overview of database objects currently being worked with.



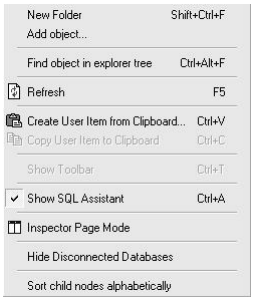
Database objects within a database can be hierarchically classified (user-specified) as wished. For example, for an Accounts project, only those objects necessary for all accounting processes are included, a Sales project would include certain objects used in Accounts and also, in addition, sales-specific objects.

This is ideal for large software projects in an enterprise.

The first time a folder or object is inserted in the project tab, IBExpert asks for confirmation whether it should create certain system tables for the project page.

This only needs to be confirmed once. Following this, folders and objects can be inserted as wished using the right mouse button menu, [Shift + Ctrl + F] or drag 'n' drop in the Inspector Page Mode, to organize databases individually and personally.

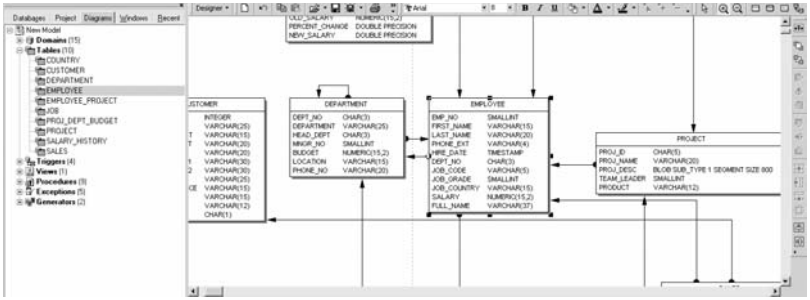
The context-sensitive right-click menu offers a number of further options:



These menu options allow new folders to be created, objects to be added to or deleted from a project (and searched for within the Explorer tree). User items may be created and copied; and the visual display customized (Show SQL Assistant, Inspector Page Mode, Hide Disconnected Databases). Since IBExpert version 2004.2.26.1 there is also the added option to sort items in alphabetical order, using the menu item "Sort child nodes alphabetically".

Diagrams (Database Designer)

The Diagrams page was added in IBExpert version 2004.9.12.1. It provides a Model Navigator to navigate models in the Database Designer quickly and easily.



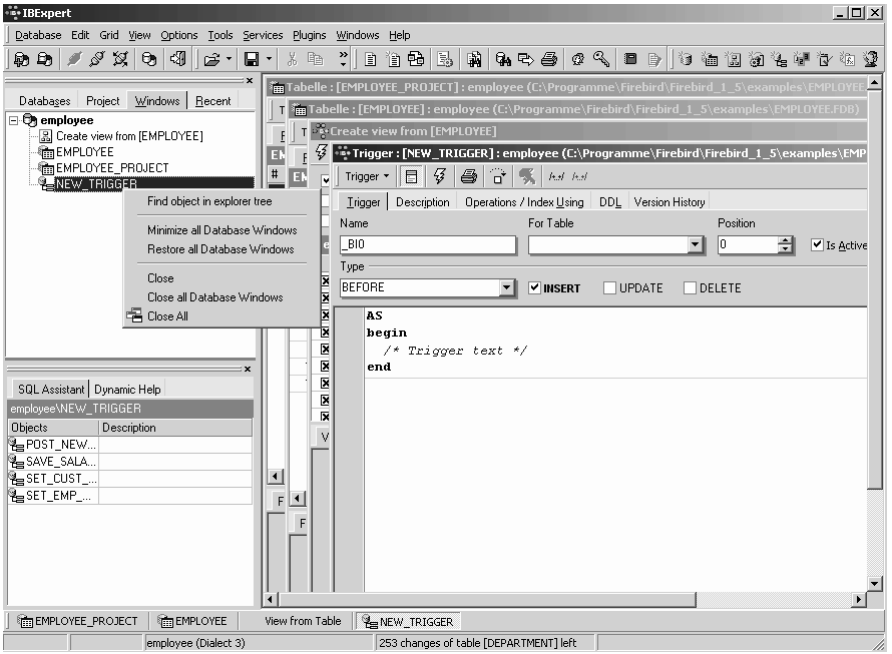
Simply click on an object in the DB Explorer, and it is immediately marked in the main Database Designer window. Double-clicking on a selected object automatically opens the Model Options page in the Database Designer.

Please also refer to the Model Navigator in the SQL Assistant.

Windows Manager

The Windows Manager can be opened using the IBExpert Windows menu item Windows Manager, the key combination [Alt + O], or - of course - by simply clicking on the Windows tab heading directly in the DB Explorer.

In the DB Explorer, the Windows page displays a list of all open windows, and allows the user to change quickly and easily from one window to the next by simply clicking on the object name in the list.

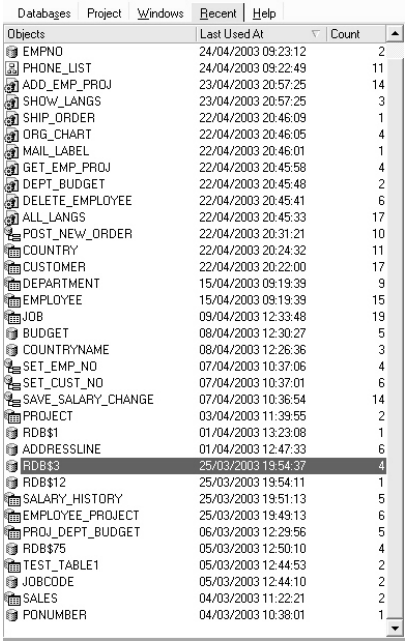


The right mouse button can be used to close individual or all windows, or to find the selected object in the DB Explorer database tree.

The open windows can also be viewed and selected in the windows bar, directly above the status bar at the bottom of the IBExpert Screen.

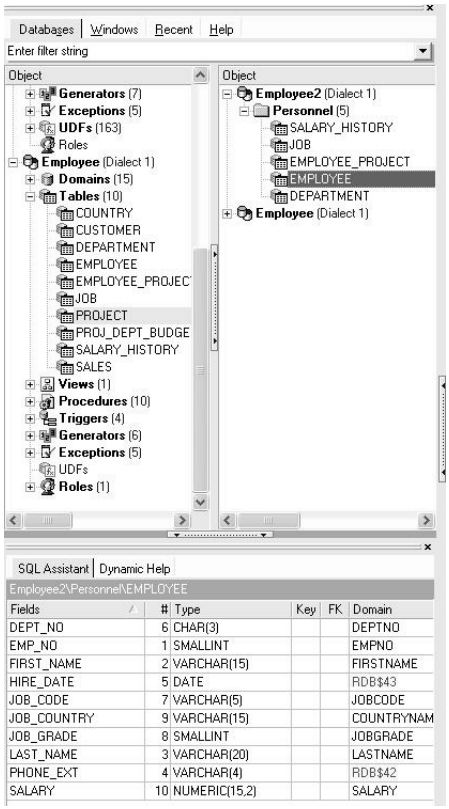
Recent List

By clicking on the Recent tab in the DB Explorer, a list of the most recent objects worked upon appears. This list can be sorted by object name, date or count in ascending or descending order, by simply clicking on the column header. The object can be reopened by double-clicking.



Inspector Page Mode

When either the Database Page or the Project Page in the IBExpert DB Explorer is active (i.e. visible in the foreground), it is possible to compare the two to each other by switching on the Inspector Page Mode. This can be done using the right-click menu and selecting Inspector Page Mode, to produce two adjacent windows:



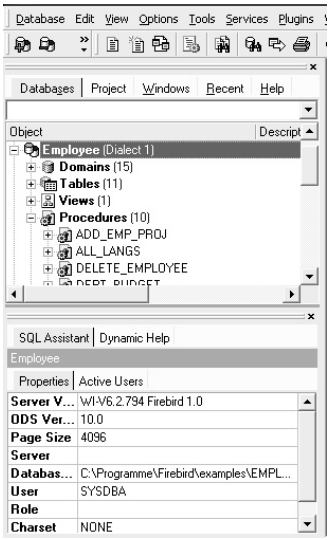
Objects can be dragged 'n' dropped from one window to the other, allowing a quick and easy selection of those objects necessary for a project.

To return to a single window display in the DB Explorer, simply right-click and the select the menu item Inspector Page Mode again.

1.5.6 (5) SQL Assistant

The IBExpert SQL Assistant offers additional detailed information regarding the highlighted database, object or group of objects in the DB Explorer.

The SQL Assistant can be found in the lower left-hand part of the screen, directly below the DB Explorer.



When a database in the DB Explorer is highlighted, the Properties page displays the actual server version of InterBase or Firebird (this can be subsequently corrected in the Database Registration if specified wrongly or previously unknown). The Active Users page shows which users are currently logged on to the database.

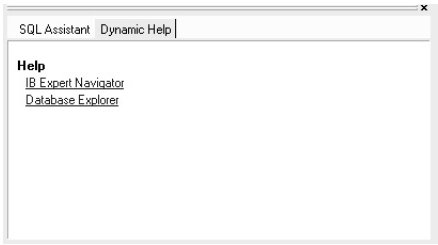
Selecting an object group in the DB Explorer displays a list of the corresponding objects. Selecting a single object displays detailed object information and content in the SQL Assistant.

When a table is selected in the DB Explorer, the fields are not only displayed in the SQL Assistant, but can also be selected and incorporated into any of the SQL Editors using drag 'n' drop. Since version 2004.2.26.1 this has been greatly improved. When an object node(s) is dragged from the DB Explorer or SQL Assistant, IBExpert will offer various relevant versions of text to be inserted into the Code Editor.

The SQL Assistant can be blended in and out as wished using [Ctrl + A] or the DB Explorer right-click menu item Show SQL Assistant.

Dynamic Help

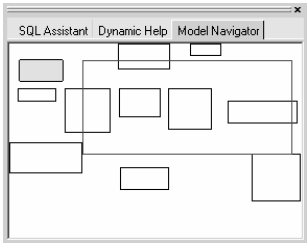
The Dynamic Help page can be found in the SQL Assistant (underneath the DB Explorer) and offers context-sensitive help.



Since IBExpert version 2004.2.26.1, this has been replaced by a new context-sensitive dynamic help system. Pressing [F1] in any of the IBExpert forms now opens a new web-based Help page. It is also possible to download all Help files from http://www.ibexpert.info/documentation/documentation.zip and unzip this in the IBExpert main directory with subdirectories (there must be a new subdirectory called documentation). If a local Help document is available, it will be opened in the browser. Otherwise the browser will open the page from our web server.

Model Navigator (Database Designer)

The Model Navigator page was added in IBExpert version 2004.9.12.1. It provides a visual orientation to aid navigation of models in the Database Designer.



The red rectangle indicates which part of the database model is currently being displayed in the main Database Designer window. It is possible to move this rectangle by drag 'n' dropping with the mouse - much quicker and easier than moving about in the main Database Designer window.

Please also refer to the Diagrams page in the DB Explorer which lists all model objects in the usual DB Explorer tree form.

1.5.7 (6) Windows Bar

The IBExpert windows bar is a horizontal bar and can be found in the lower area of the screen, directly above the status bar:

This displays the number and type of open windows in IBExpert; the symbols indicating the editor type (e.g. Table Editor, Procedure Editor, etc.), followed by the object name or editor type.

1.5.8 (7) Status Bar

The IBExpert status bar is a horizontal bar found in the lower area of the screen, directly below the windows bar:

This displays information concerning the current status of, for example, the connected database, the IBExpert window contents and memory.

changes of table left

Note: each table in an InterBase/Firebird database has its own metadata changes counter. Each table can be altered 255 times (add or remove columns, change field type etc.). When any of these counters reaches the value of 255 it is not possible to alter any tables any further, and a database backup and restore is necessary.

IBExpert indicates in the status bar how many changes my be made in the database before being forced to perform a database backup and restore. This message may be deactivated, if wished, using the IBExpert menu item, Database / Register Database or Database / Database Registration Info, and checking the option "Don't display metadata changes counter info" on the Additional page.

1.5.9 Exit

Exit is the command used to close IBExpert. The program can be closed by using either the menu item Database / Exit, or clicking the black X button in the top right-hand corner of the screen. Alternatively the key combination [Alt + F4] may be used.

IBExpert requires confirmation that you really wish to exit the program - either click on Yes or press the Return/Enter key. Should you wish to eliminate this default setting, uncheck the Confirm Exit box found in the IBExpert Options / Environment Options menu under Confirmations.

Any editors left open at the time of exiting, will automatically be loaded the next time that IBExpert is started, unless the following default setting is switched off: Options / Environment Options / Preferences - uncheck Restore Desktop after Connect.

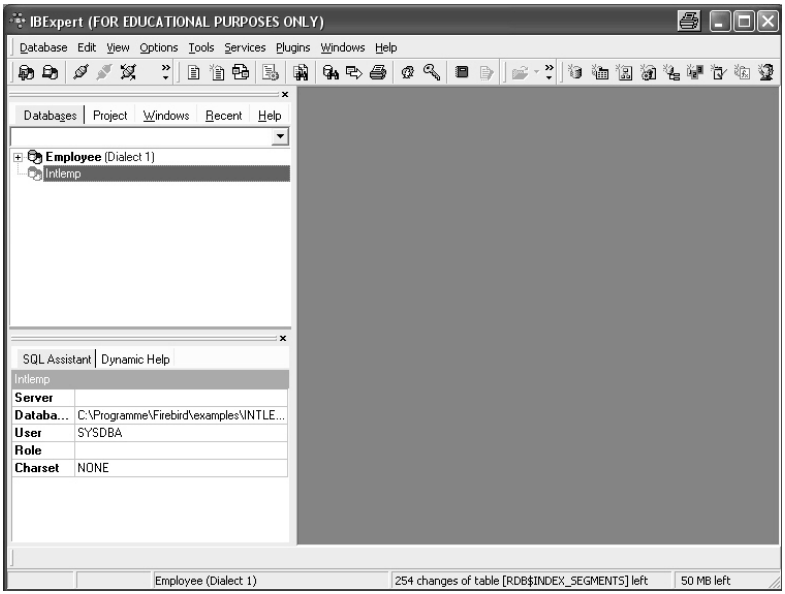
All connected databases are automatically disconnected when IBExpert is shut down.

2 Database

A relational database is a collection of tables related to each other, each storing a specific set of data. A database also contains Indices, business rules and processes, for the database administration. It can be considered to be a collection of pages, each page being of a pre-defined size, which is determined when the database is created.

The data itself may contain any information, be it for business accounts, sales, scientific measurement logging or personal addresses and finances. The information stored in a database may be shared by more than one application.

Available databases can be viewed in IBExpert in the left-hand panel, the DB Explorer. Connected databases are displayed in bold type.



The relational system assumes the following:

- The physical storage model and the logical data storage in files are independent of each other.
- All data is stored in tables.
- Users do not need to know which files are stored how and where. Access occurs via tables, which represent a logical view of data.
- A data set's physical position in the database is irrelevant to the user.
- The relational database administrates all information necessary for internal access optimization internally, using indices.
- The relational database undertakes the data integrity checks independently.

InterBase/Firebird administrates data in database objects. Within the database, the following database objects (database metadata) can be created and maintained:

- Domains
- Tables
- Generators
- Constraints
- Indices
- Views
- Triggers
- Stored Procedures
- Exceptions
- Blob Filters
- User-Defined Functions (UDFs)

2.1 Database Design

A good database design is vital for a client/server application. It is important to think about the design of the tables among each other to optimize data storage, i.e. in which table should each quantity of information be placed, and how this table should be linked to the information in other tables. The normalization process helps here as it avoids double data storage as well as unnecessary wastage of space; data access becomes considerably more efficient, at the same time improving database performance and data integrity. Special business problems in the database can be solved with the aid of database design; for example, they enable typical relationships between master and detail tables.

Relational databases work best when data is broken up into different tables that are joined together on common columns. This design results in narrower, longer tables, where the primary key is used to access the data, and indices are used to speed this process.

Database models are generally designed to solve specific business problems: they allow typical business data relationships to be represented. This is particularly important, for example, when many detail rows need to be joined to one master row. This is most often done by splitting the data into two or more tables and joining them on a shared column. When data is represented in this way, some duplication is unavoidable. There are always columns that must appear in each table in order to actually create the join. However database models allow you to minimize unnecessary duplication.

These models also ensure that if a value is updated in one table, the matching values are updated in related tables, known as referential integrity.

The IBExpert Database Designer is an ideal tool for data modeling and design, whether creating a model of an existing database for analysis, or designing a new database.

2.1.1 Database Normalization

The goal of normalization is to reduce redundant information. In other words, only store one piece of information one time. A table is said to have repeating groups and to be un-normalized if:

- it contains many repetitions of the same piece of information in the same column
- more than one column contains almost the same type of information

- a column consists of complex information that should be broken into several smaller pieces.

Tables without repetitive values are described as normalized. The transition from one design to the other is called normalization.

Five forms of normalization can be differentiated. The first four normalization forms will be described very briefly here, the fifth being an extremely theoretical demand on tables. There is a wide range of specialist literature available on this subject, for those requiring more in-depth information.

Rule Zero

The relational theory requires, as a rule, a unique key in each table, in order to identify information clearly. This is composed from the three following:

- The table, in which the data is stored,
- The field in this table, which needs to be accessed,
- The value of the primary key for this data set.

It is clear that the primary key is important for the identification of a data set. At the same time InterBase/Firebird automatically creates an index via the primary key, so that searches in multi-table queries are much quicker than those without an index.

A table has only one primary key, although the primary key can consist of several columns. So, a simple rule for normalizing databases is - always key your tables!

First Normal Form

The first rule of database design states: eliminate repetitive groups. For each group of related columns, make a separate table and give that table a primary key.

A table is said to be in first normal form if all columns contain atomic (i.e. indivisible) values only. This is another way of saying that there are no repeating groups.

First Normal Form Problems

INSERT anomalies (e.g. certain master data cannot be recorded until an order or sale is placed), UPDATE anomalies (it is too easy to miss certain entries when updating) and DELETE anomalies (whole records disappear from the database, including master data).

Second Normal Form

The second rule of database design is: If a table column is only dependent upon part of a multicolumn key, this column should be removed to a separate table.

For a table in the second normal form, it must already be in the first normal form, and all non-key-column contents must be dependent upon the complete primary key. The second normal form avoids double storage of information.

Tables become narrower, the more the database is normalized, with less duplication of wide column values. Where duplication is unavoidable, it can be made as small as possible by using an ID number.

Second Normal Form Problems

There are no repetitive groups, and all columns are dependent on their table's primary key. However some irregularities can still be found; from the relational viewpoint, certain fields may have no relationship to each other, e.g. a customer telephone number has nothing to do with an order number. It is a customer feature, not an order feature, and leads to storage of redundant data. For this reason, it makes sense to remove this information to a separate table.

Third Normal Form

The third normal form is tantamount to the second normal form, as it is also aimed to avoid update, delete and insert problems. It is mainly concerned with relationships in tables with a single column primary key. The rule can be defined: when column contents have no connection to the table's primary key, they should be removed to a separate table.

A table is in the third normal form, when each column describes data corresponding to the primary key. Most operations are carried out on key fields, ensuring a high performance. Details are maintained in their own tables, secure from UPDATE, DELETE, and INSERT anomalies.

Fourth Normal Form

The majority of applications need go no further than the third normal form. There are however certain situations, in which the data segmentation needs to be refined. For example, each sales team order needs to be assigned to the sales person responsible, for a planned monthly sales per person summary. Where should this information be stored? A simple solution is to expand the relevant table to include the field SalesContact.

The problem becomes clear, when it is considered that often more than one call was necessary to result in one sale. The fourth normal form rule is: isolate independent multiple relationships.

There are one or more calls leading to each order. The order position information has nothing to do with the telephone calls made. Therefore the call information is removed to its own table, to ensure that, here also, the independence of information in each table is warranted.

2.2 Inside InterBase/Firebird

This section offers a more in-depth view of the InterBase/Firebird database and how it functions.

2.2.1 Space management in InterBase

This article was written by Ann Harrison, IBPhoenix.

An InterBase database consists of a set of fixed length pages of different types. Ten page types are currently defined:

- Header Page (HDR)
- Data Page (DPG)
- Blob Page (BLP)
- Transaction Inventory Page (TIP)
- Page Inventory Page (PIP)
- Pointer Page (PTR)
- Index Root Page (IRT)
- B-tree Page (BTR)
- Write-Ahead Log Page (LIP)
- Generator Page (GEN)

Two of these, page inventory and pointer are used for space management. For those not familiar with InterBase's on-disk structure, here is a brief description of each of the page types.

Page Types

All page types include a header that holds generic page information.

```
typedef struct pag {
    SCHAR pag_type;
    SCHAR pag_flags;
    USHORT pag_checksum;
    ULONG pag_generation;
    ULONG pag_seqno; /* WAL seqno of last update */
    ULONG pag_offset; /* WAL offset of last update */
} *PAG;
```

Each specific page type adds more structural information. The first page in every database is its header (HDR) page. Secondary database files also have header pages. Data pages (DPG) contain data; blob pages (BLP) contain blob data for those blobs that don't fit on the data page with their parent record. Any data page contains data for only one table. Any blob page contains data for only one blob. Transaction inventory pages (TIP) contain an array of bits, two per transaction, that indicate the state of the transaction. A transaction id is an index into this array. Every page in the database is represented by one bit in a page inventory page (PIP). The bit indicates whether the page is currently in use. Page inventory pages (PIP) occur at fixed intervals in the database - the interval is determined by the page size. A pointer (PTR) page is the top-level locator for data pages. It contains an array of page numbers for the data pages of the table and a corresponding array of bits that indicate whether the page is full. No pointer page entry is made for blob pages or pages that contain only the second or subsequent pages of data from a fragmented record. Index (IRT) root and b-tree (BTR) pages are what they appear to be. The only odd thing is that each table can have only one index root page. For that reason, you can put more indexes on a table when you use a large page size. The log information pages (LIP) for the write-ahead log are not currently used, though code to use them is included conditionally. Generator pages (GEN) contain arrays of 32 or 64 bit integers, depending on the dialect.

Basic Page Allocation

Page allocation is handled by the routine PAG_allocate in PAG.C. When some routine needs a new page, it calls PAG_allocate. PAG_allocate gets the page control block from the database block to find the first page information page that has free space. If necessary, it reads that pointer page from disk. It then scans the page, looking for the first free bit, and assigns that page number to the new page. The page image is created in the cache manager (CCH), which give it the appropriate page type. The cache manager then returns the buffer pointer to the routine that requested the new page. When the page is marked for write, the page I/O module (PIO) writes it to the appropriate offset in the database file.

Housekeeping Note. To keep the database on disk consistent, the pointer page must be written before any page that is allocated from it to avoid doubly allocated pages. Under ordinary circumstances, the shared cache or page locks keep this from happening. If, however, the machine were to crash in mid-operation, the order of page writes can prevent corruption.

Advanced Page Allocation

If the system does not find space on the first PIP it examines, it reads the next, and so on until it searches the last PIP. If the last unallocated page is the last bit on the last PIP, the routine allocates that page number as the next new PIP, formats it, marks the new PIP as needing to be written and the old PIP as dependent on it. Finally, PAG_allocate calls itself to allocate the page that was requested originally, using the first bit on the new page inventory page. If the database is defined to hold multiple files, when page allocation reaches the end of the first file, it creates a new file, gives it a new header, and resumes allocating pages.

Additional Page Allocation Steps For Data Pages

A data page is recorded as being in use both in the PIP and in a pointer page for that table. Once the new data page has been marked for write, its page number is written into the first free slot one in the current pointer page or the first free slot on any pointer page. The order of writes is: PIP, data page, pointer-page.

Additional Steps For "Interesting" Pages

Information about interesting pages is stored in a System Table called RDB$PAGES. When an index root page, a transaction inventory page, a generator page, or a pointer page is created, a new rows is stored in RDB$PAGES. This operation can cause a new page, a new pointer page, a new page inventory page or even a new file to be allocated.

Releasing pages

The header page is never released. Generator pages and transaction inventory pages are not released either. In theory, they could be, but that would complicate (slightly) some sensitive bookkeeping for (relatively) little gain. Nor are page inventory pages are released. Once a database has grown to some size, the only way to shrink it is to recreate it from a backup. When a page is empty, it is put back in the "free space pool" by clearing its bit on the appropriate page inventory page. B-tree pages are released when the index is deleted, deactivated, or rebalanced. Blob pages are released when the blob is released, because the record that owns it is deleted or because the blob it-

self was modified. Data pages created to hold the trailing part of a fragmented row are released when the row - or at least that version of the row - is removed.

Releasing Data Pages

When the last row on a normal (non-overflow) data page is deleted, the page is returned to free space in a two-part operation. First, the page is removed from its pointer page, which is the page that associates it with its table. If that empties the pointer page, then the pointer page is also marked as released on its page inventory page. Releasing a pointer page requires changing a system table called RDB$PAGES. RDB$PAGES contains one row for each "interesting" page in the database. Pointer pages, index root pages, generator pages, and transaction inventory pages are considered "interesting". Releasing an index root page also requires deleting a row from RDB$PAGES. This process can recurse, just as the allocation process recurses, except that neither files nor page inventory pages are released.

Elementary Allocation On Page

For most of the page types, allocation of space on page is not difficult. Generator pages, transaction inventory pages, page inventory pages, and pointer pages are just arrays. When one page fills, another one is allocated. (Theoretic rather than actual in the case of generator pages, but the principle holds). Routines in the module PAG.C manage header pages - they are essentially simple structures followed by a byte array that holds the filenames for secondary files. Space on generator pages and transaction inventory pages is never reused, so there is no reason to look for space on any page of those types except the last. Space on page inventory pages is reused. When a page is released - no longer needed for whatever purpose it had - its entry is cleared. For that reason, the page number of the lowest PIP with space is carried in the database control block. That number is not considered reliable, but a good starting point.

Finding Space For Data

Each table carries with it a vector of its pointer page numbers, and two high-water marks, one for the first pointer page with data space, and one for the first pointer page with space for a new data page. When storing a record that compresses to less than the page size, DPM looks first for a pointer page with data pages that have free space, then at the header of the pointer page to find the first slot pointing to a page with space.

Now, just a bit more about data pages. Every data page has a header like this:

```
typedef struct dpg {
    struct pag dpg_header;
    SLONG dpg_sequence; /* Sequence number in relation */
    USHORT dpg_relation; /* Relation id */
    USHORT dpg_count; /* Number of record segments on page */
    struct dpg_repeat
    {
    USHORT dpg_offset; /* Offset of record fragment */
    USHORT dpg_length; /* Length of record fragment */
    }  dpg_rpt [1];
} *DPG;
```

The repeating offset/length is an array of pointers to data on the page. These pointers are called line index entries, at least by me. The actual data starts at the bottom of the page and works up. When there is no longer enough space for another line index entry and another minimal sized record, plus whatever space is reserved for future expansion (that's another topic), the page is marked full, both in its header and on the pointer page.

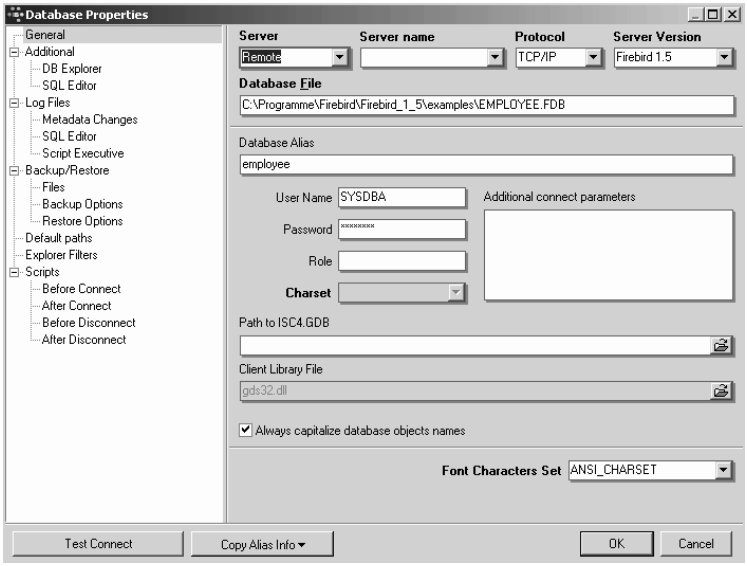
DPM goes through the line index, adding up the space on page. If there's enough for the compressed record, alignment overhead, and a line index entry, it's got a winner. However, the space may not be contiguous. In that case, DPM shuffles all the data down to the bottom of the page. Obviously, it doesn't compress the line index entries, though it does correct the offset for data that has moved. Next step is to create a new line index entry and shoot the data onto the page. Final step is to see if the page's fullness quotient has changed and make appropriate changes if so.

If there is space on page, but not enough for the current compressed record, DPM marches on through the pointer page, checking plausible candidates, then on through other pointer pages until there are no more allocated data pages.

OK, now it's time to allocate a new data page. First, find a free page in the current PIP, or the next PIPs, or create a new PIP. Next, create the page in a buffer. Now, starting with the first pointer page that has space to hold a new data page pointer, or create a new pointer page for the table. That's it. At least that's all I can explain at the moment.

2.3 Database Registration Info

Information appertaining to any of the registered databases can be viewed in IBExpert in the Database Properties dialog, started using the menu item Database / Database Registration Info... or the DB Explorer right-click menu:



The information displayed here is that which was entered, when the database was originally registered (please refer to Register Database for details).

The tree in the left panel shows the various registration options available. Certain items may be amended here. Again please refer to Register Database for further information.

New in version 2.5.0.47! It is possible to automatically connect to a database when starting IBExpert. Use the following menu: Database Registration Info / Additional / check: Open database when IBExpert starts.

New in version 2004.04.01.1 - under Database Registration Info / Additional there are now two additional options:

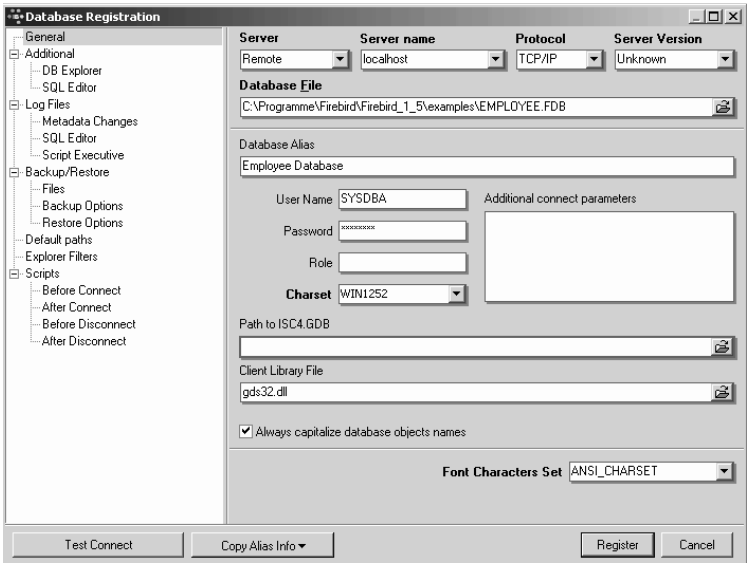
- Disable plan request in SQL Editor
- Disable performance analysis.

New to version 2003.12.18.1: the added possibility to execute SQL scripts before and after connecting to the database and before and after disconnecting from the database. And under Database Registration Info / Additional there is now the additional option - "Always prompt for a user name and password". If this option is activated, IBExpert will display a login prompt dialog each time you try to connect to the database.

2.4 Register Database

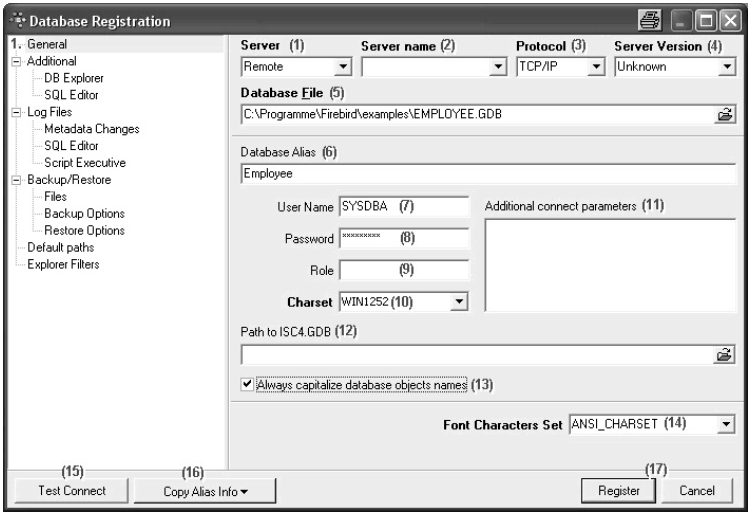
Database registration is necessary, in order for IBExpert to recognize the presence of a database. It is possible to specify certain options, settings and defaults here. The Database Registration Editor can be opened using the IBExpert menu item Database / Register Database, or key combination [Shift + Alt + R]. It is automatically generated when the 'Register Database After Creating' checkbox is flagged in the Create Database dialog.

The Database Registration dialog is split into two sections: on the left-hand side a tree overview of the various registration options is displayed; the right input panel shows the information and setting options available for each tree subject.



2.4.1 General

The following entry fields allow the user to specify certain general properties and defaults for the database to be registered.



(1) Server : firstly the server storing the database needs to be specified. This can be local or remote (see Create Database).

By specifying a local server, fields **(2)** and **(3)** are automatically blended out, as they are in this case irrelevant.

(2) Server name: must be known when accessing remotely. The syntax is as follows:

- Windows `SERVER_NAME:C:\path\database.gdb`
- Linux `SERVER_NAME:/path/database.gdb`

The standard port for InterBase and Firebird is `3050`. However this is sometimes altered for obvious reasons of security, or when other databases are already using this port. If a different port is to be used for the InterBase/Firebird connection, the port number needs to be included as part of the server name. For example, if port number `3055` is to be used, the server name is `SERVER/3055`. For using an alias path for a remote connection, please refer to the article Remote Database Connect using an Alias.

(3) Protocol: a pull-down list of three options: TCP/IP, NetBEUI or SPX. TCP/IP is the worldwide standard (please refer to Register Database for more information).

(4) Server versions: this enables a server version to be specified as standard/default from the pull-down list of options.

(5) Database File: by clicking on the folder icon to the right of this field, the path can easily be found and specified and the database name and physical path entered. The database name must always be specified with the drive and path when registering a database. Please note that the database file for a Windows server must be on a physical drive on the server, because InterBase/Firebird does not support databases on mapped drive letters.

For example for Firebird:

```
C:\Programs\Firebird\Firebird_1_5\examples\EMPLOYEE.FDB
```

for InterBase:

```
C:\Programs\Interbase\examples\EMPLOYEE.GDB
```

(6) Database Alias: descriptive name for the database (does not have to conform to any norms, but is rather a logical name). The actual database name and server path and drive informati0n are hidden behind this simple alias name - aiding security, as users only need to be informed of the alias name and not the real location of the database.

(7) User Name: the database owner (i.e. the creator of the database) or SYSDBA.

(8) Password: if this field is left empty, the password needs to be entered each time the database is opened. Please refer to Database Login for further information. The default password for SYSDBA is *masterkey*. Although this may be used to create and register a database, it is recommended - for security reasons - this password be changed at the earliest opportunity.

(9) Role: an alternative to **(7)** and **(8)**; can initially be left empty.

(10) Charset (abbreviation for Character Set): Here the default character set can be specified. This is useful, when the database is created to be used for foreign languages, as this character set is applicable for all areas of the database unless overridden by the domain or field definition. If not specified, the parameter defaults to NONE, i.e. values are stored exactly as typed. For more information regarding this subject, please refer to Charset/Default Character Set. If a character set was not defined when creating the database, it should not be used here.

(11) Additional connect parameters: input field for additional specifications. For example, system objects such as system tables and system generated domains and triggers can be specified here. They will then automatically be loaded into the DB Explorer when opening the database alias.

(12) Path to ISC4.GDB: This can be found in the InterBase or Firebird main directory. This database holds a list of all registered users with their encrypted passwords, who are allowed to access this SERVER.

When creating new users in earlier InterBase versions (<6), IBExpert needs to be told where the `ISC4.GDB` can be found. Since InterBase version 6 or Firebird 1 there is a services API. So those working with newer versions may ignore this field!

(13) Always capitalize database objects' names (checkbox): this is important as in SQL Dialect 3 as entries can be written in upper or lower case (conforming to the SQL 92 standard). InterBase however accepts such words as written in lower case, but does not recognize them when written in upper case. It is therefore recommended this always be activated.

(14) Font character set: this is only for the IBExpert interface display. It depends on the Windows language. If an ANSI compatible language is being used, then the `ANSI_CHARSET` should be specified.

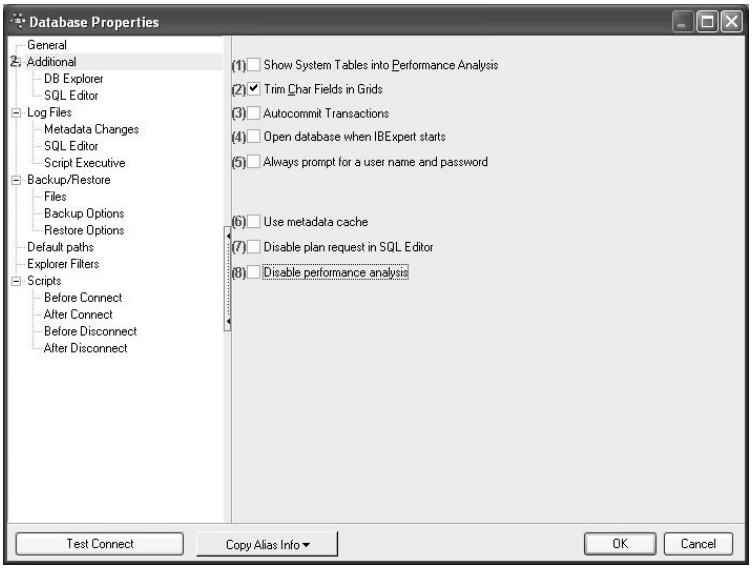
(15) Test connect: the Comdiag dialog appears with a message stating that everything works fine, or an error message - please refer to the IBExpert Services menu item, Communication Diagnostics for more details.

(16) Copy Alias Info: here alias information from other existing registered databases can be used as a basis for the current database. Simply click on the button and select the registered database which is to be used as the alias.

(17) Register or Cancel: after working through all the options listed in the tree view on the left, the database can be registered or cancelled.

2.4.2 Additional

The Database Registration / Additional options are as follows:



(1) Show System tables into Performance Analysis: the developer can choose whether he also wishes to have the database system tables (in addition to the user-defined objects) included in the Performance Analysis found in the SQL Editor, Stored Procedure Editor and Visual Query Builder.

(2) Trim Char Fields in Grids: adapts field length to ideal length in all grids (see Table Editor / Data and SQL Editor / Results as well as the IBExpert Grid menu).

(3) Autocommit Transactions: This allows all transactions to be committed immediately (i.e. IBExpert no longer asks for confirmation of a commit command and there is NO option to rollback). This is an *EXTREMELY* dangerous option! For example, if an irreversible DROP command has been wrongly entered (e.g. instead of typing a `[FIELD_NAME]` the `[DATABASE_NAME]`is mistakenly entered), it is still automatically committed.

(4) Open database when IBExpert starts: New in version 2.5.0.47! Checking this option automatically connects this database when IBExpert is started.

(5) Always prompt for a user name and password: New in version 2003.12.18.1. If this option is activated, IBExpert will display a login prompt dialog each time you try to connect to the database.

(6) Use Metadata cache: e.g. when accessing remotely using a modem line, the InterBase server can only be accessed at a limited speed. IBExpert needs to know which information it needs to fetch, and this may take some time. If the metadata cache is checked, IBExpert does not download the complete database each time, only the information that it really needs.

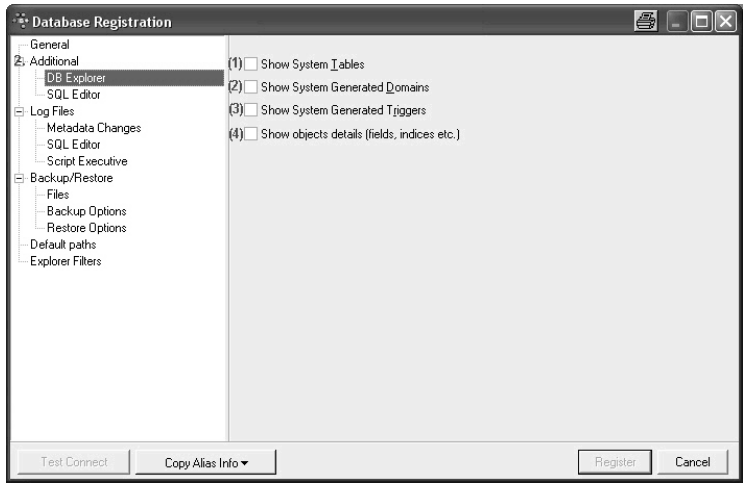
(7) Disable plan request in SQL Editor: New option in version 2004.04.01.1. This deactivates the query plan displayed in the lower panel of the Results page in the SQL Editor.

(8) Disable performance analysis: New option in version 2004.04.01.1. This deactivates the Performance Analysis page in the SQL Editor. This may be desirable, when working remotely on a slow modem connection.

(9) Disable object description in hints These hints appear when you move the mouse cursor over the column captions
in the Data Grid. If descriptions in these hints are not disabled IBExpert executes some SELECTs to get them from the database. If you're working with the database using a slow modem connection this decrease the performance dramatically.

(10) Don't display metadata changes counter info This deactivates the message "253 changes to [TABLE] left", which is displayed in the status bar.

Additional / DB Explorer



(1) Show System Tables: tables generated by InterBase/Firebird are displayed in the IBExpert DB Explorer in red.

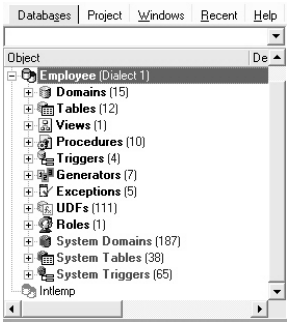
(2) Show System Generated Domains: domains generated by InterBase/Firebird are displayed in the IBExpert DB Explorer in red.

(3) Show System Generated Triggers: triggers generated by InterBase/Firebird are displayed in the IBExpert DB Explorer in red.

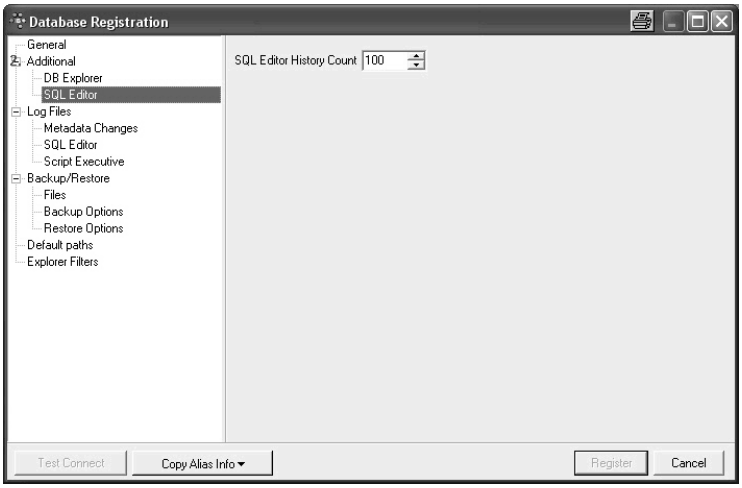
(4) Show System Indices: indices generated by InterBase/Firebird are displayed in the IBExpert DB Explorer in red.

(5) Show objects details (fields, indices etc.)

For database development it is wise to have all these items visible in the DB Explorer.



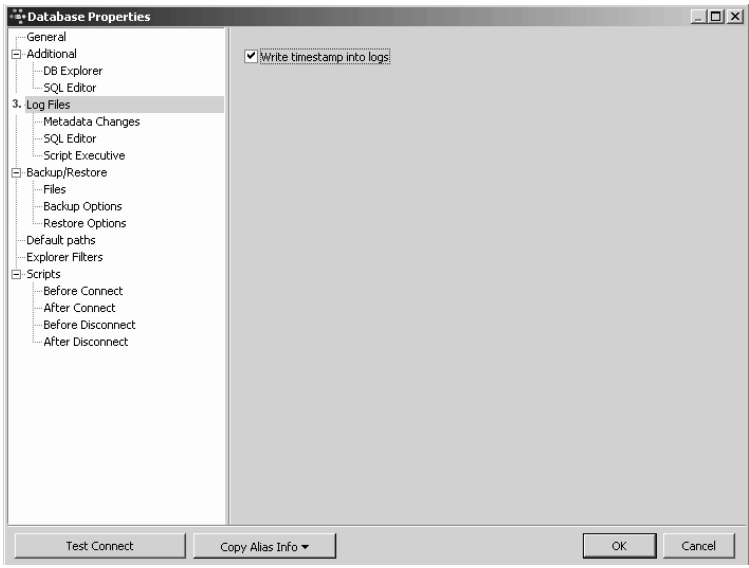
Additional / SQL Editor



The SQL Editor History Count determines the number of SQLs that are saved and displayed in the IBExpert SQL Editor. Here the default value of 100 can be adjusted as wished.

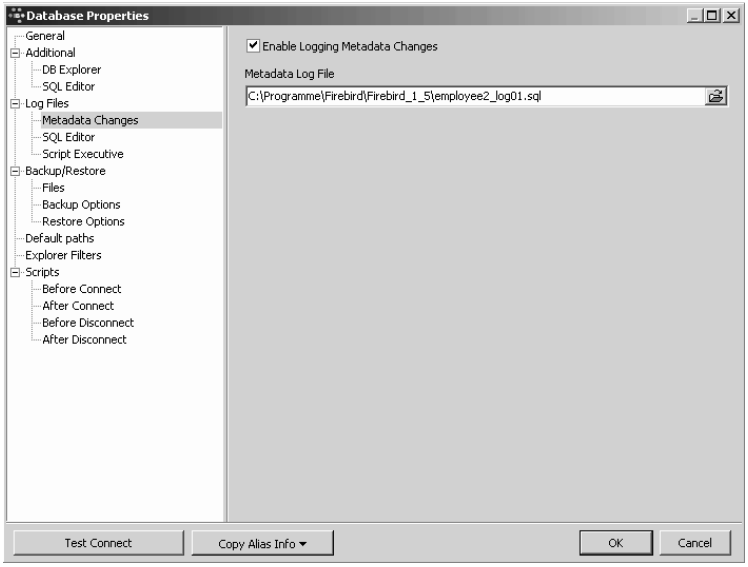
2.4.3 Log Files

If you would like IBExpert to protocol all statements that change metadata and/or are executed from the SQL Editor, use this section to enter path and file names. This is useful for keeping a record of which changes were made to the data structure in IBExpert.



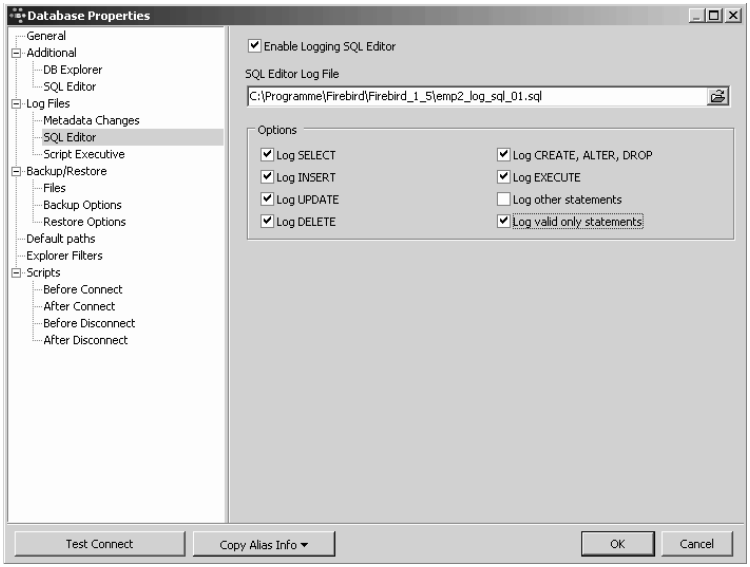
Write Timestamp into logs: the timestamp option is useful for noting date and time on logs.

Log Files - Metadata changes



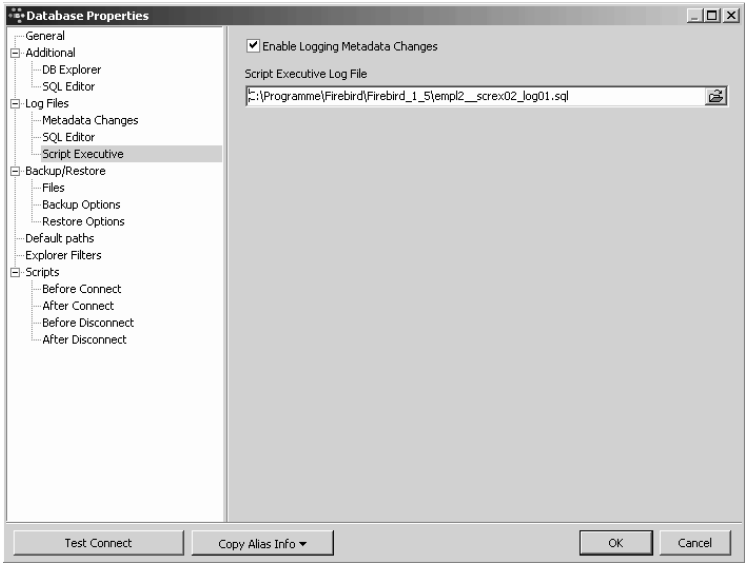
Enable Logging Metadata Changes: allows all changes to metadata to be logged, in order to follow all alterations to the data structure.

Log Files - SQL Editor



Enable Logging SQL Editor: Allows all SQL Editor work to be logged - a useful option, which should be checked. Should the log files become too large, older logs can always be deleted at regular intervals.

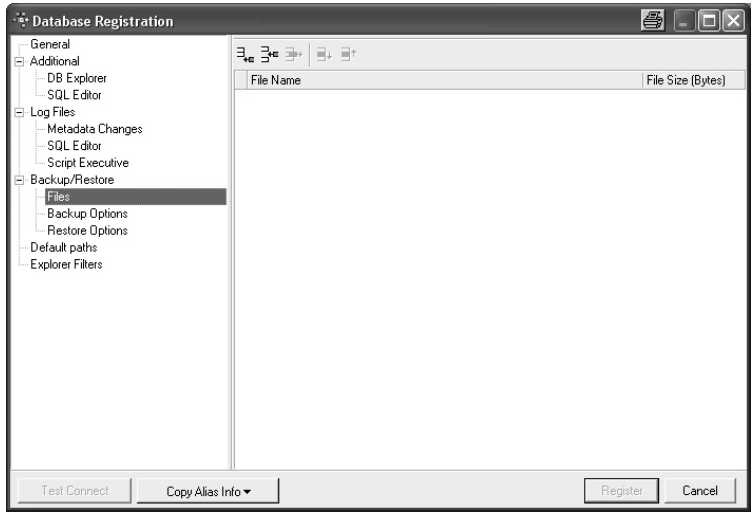
Log Files - Script Executive



Enable Logging Metadata Changes: checkbox to specify whether all alterations to metadata should be logged or not.

2.4.4 Backup/Restore

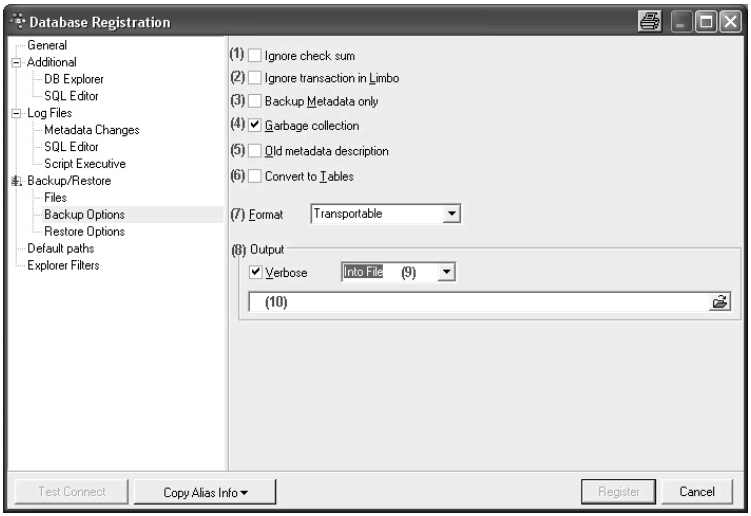
Files



Backup and restore file names and options can be specified for each database alias. This makes it easier to backup a database with a single mouse click from the IBExpert Services menu.

Using the first icon on the left a file name can be specified as the default file for backups. When left empty, the backup file name must be specified for each backup. For versions since Firebird 1.0 or InterBase 6.5 the file size is irrelevant (64B file system). Secondary backup files can also be specified here.

Backup Options



(1) Ignore check sums: ignores any check sum errors and continues to backup the database. This option should be selected if a backup is being performed because database errors are suspected. If this option is not checked, the backup is aborted if a check sum error is found. This is one possibility to force a backup for a corrupt database. Please note that checksums are not maintained in UNIX versions.

(2) Ignore Transactions in Limbo: in limbo transactions are those which are supposed to run across two or more databases and have been started, but neither finally committed nor rolled back at the time of the database backup. This option backs up only the most recent, committed transactions. It allows you to back up a database before recovering corrupted transactions. Generally, you should recover in limbo transactions before performing a backup.

(3) Backup Metadata only: results in an empty copy of the database, as only the database definition (metadata) is saved, not the data itself. This option is similar to using Windows ISQL to extract a database to a file.

(4) Garbage collection: checks every row, removing outdated versions, empty pages and parts of them. Because each page is carefully examined, the backup takes longer. Should a backup need to be executed rapidly, the garbage collection can be switched off here. Only the deleted and NOT the older versions of updated data sets are dumped. The distribution of page occupation can be viewed in the database statistics. The garbage collection in InterBase/Firebird can also be started using the SELECT command.

(5) Old Metadata Description: this enables a backup and restore to older InterBase versions.

(6) Convert to Tables: this concerns so-called external files. Following a backup the external files are also incorporated, and then restored as tables.

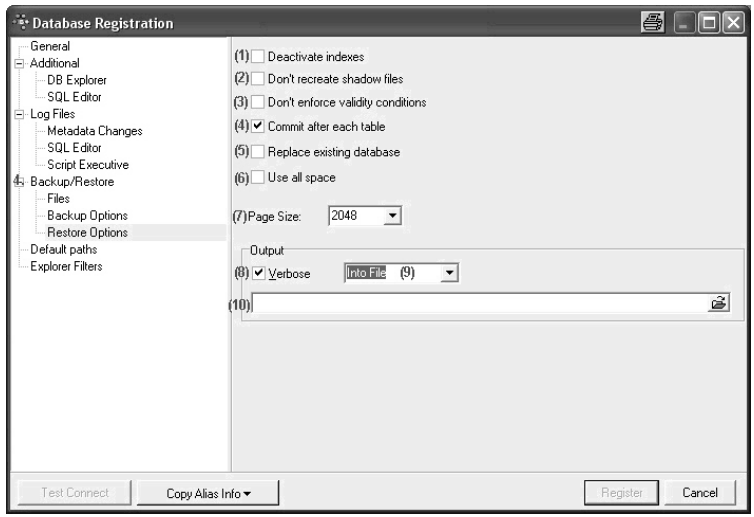
(7) Format: the options 'transportable' or 'non-transportable' are offered here. As a rule always choose "transportable", so that the database can be easily transported to other platforms such as Linux.

(8) Verbose Output: Writes step-by-step status information to the output log. This option is useful if the backup is failing, and the reasons need to be tracked down.

(9) The **output** log options 'on-screen' or 'into file' are offered here.

(10) File name, path and drive can be specified here, if the **'into file'** output option has been chosen.

Restore Options



(1) Deactivate indexes: This option does not restore indices as part of the restore process. It is used to improve restore performance. If this option is not checked, InterBase/Firebird updates indices after all tables have been filled with the restored rows. This option can also be used if duplicate values are suspected in indices that are flagged as unique. After the duplicate values have been found and corrected, the indices can be reactivated.

(2) Don't recreate shadow files: this option deletes the database shadow definition. This option is required if the destination database does not support shadows, if you are migrating from an earlier version of InterBase where shadows were not supported, or if the machine where the shadow resides is not available.

(3) Don't enforce validity conditions: this option does not restore constraints, i.e. it deletes the validity constraints from the database's metadata definition. It is important to save a copy before a restore is performed with this option checked.

This option is necessary if the validity constraints were changed after data had already been entered into the database. When a database is restored, InterBase/Firebird compares each row with the metadata; an error message is received if incompatible data is found. Once the offending data has been corrected, the constraints can be added back.

(4) Commit after each table: this option restores metadata and data for each table in turn as a single transaction, and then commits the transaction. This option is recommended, so that should a problem occur during the restore, at least all correct tables are restored. It is particularly useful, if corrupt data is suspected in the backup, or if the backup is not running to completion. Normally, InterBase/Firebird first restores all metadata and then the data.

(5) Replace existing database: this should. as a rule, be toggled, as it makes no difference if there is no database present as yet. Although leaving this option unchecked provides a measure of protection from accidentally overwriting an existing database file that may still be needed.

(6) Use all space: only relevant if restoring the database to a CD. In this case 100% space of each page is used, and not, as is usual, 80%.

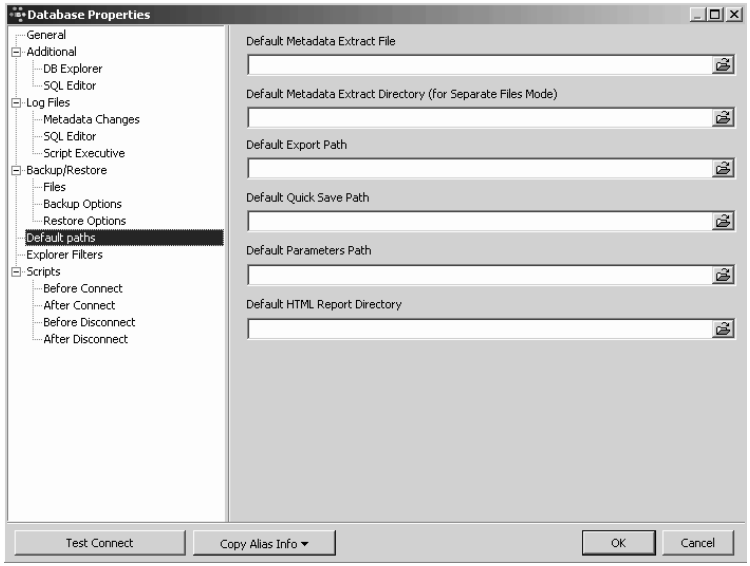
(7) Page size: Changes the default size of each page. There are numerous reasons for wanting to change the database page size (please refer to page size).

(8) Verbose Output: Writes step-by-step status information to the output log. This option is useful if the backup is failing, and you need to track down the reason.

(9) The **output** log options 'on-screen' or 'into file' are offered here.

(10) File name, path and drive can be specified here, if the 'into file' output option has been chosen.

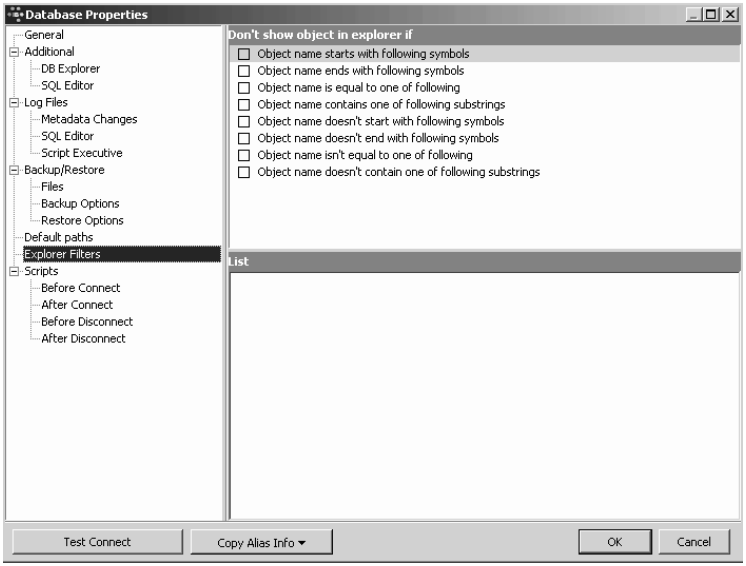
2.4.5 Default paths



Here standard default drives, paths and files may be specified, if wished, for the following:

- Metadata Extract File
- Metadata Extract Directory (for Separate Files Mode)
- Export Path
- Quick Save Path
- Parameters Path
- HTML Report Directory

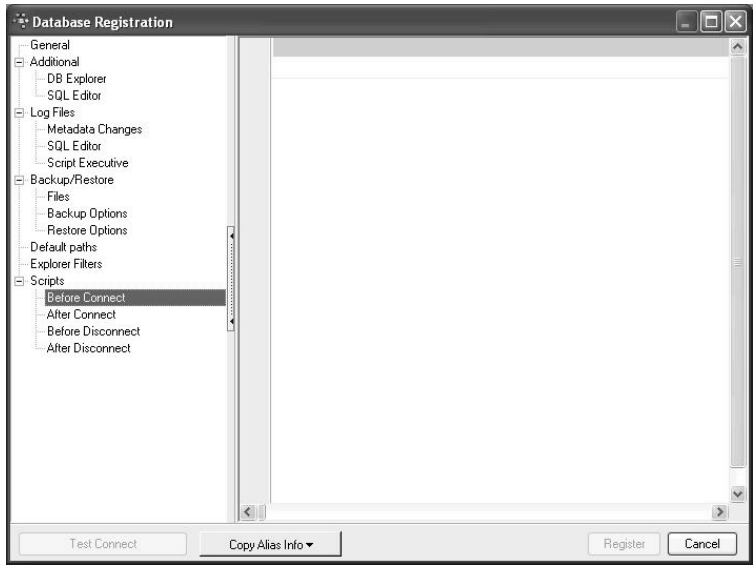
2.4.6 Explorer Filters



This is only of interest for extremely large and complex databases with multiple registrations. It refines the selection of database objects displayed in the IBExpert DB Explorer. The database object names displayed can be filtered according to one or more of the conditions listed.

2.4.7 Scripts

Since IBExpert version 2003.12.18.1 there is the added possibility to execute SQL scripts before and after connecting to the database and before and after disconnecting from the database:



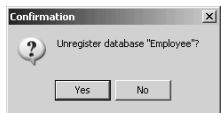
2.5 Unregister Database

It may be desirable to unregister one or more databases in IBExpert, for example when a remote link to a customer database will never be needed again. Unregistering a database does not delete the database; it merely deletes the registration necessary for working with IBExpert.

If you are unsure whether a registered database will ever be needed again, but are tired of having it displayed in the DB Explorer every time work is started, it is possible to blend out unconnected databases using the DB Explorer right-click menu item Hide Disconnected Databases.

A database can be unregistered using the IBExpert menu item Database / Unregister Database, the DB Explorer right-click menu, or the key combination [Shift + Alt + U].

IBExpert asks for confirmation:



before finally unregistering the database.

2.6 Connect to an existing Database

After starting IBExpert, you will see the Database Explorer on the left side. Before a database connection can be made, the database must be registered (please refer to Register Database).

A database connection can be made to a registered database simply by double-clicking on the database alias name, displayed in the DB Explorer. There are also a number of menu options: either using the IBExpert menu item Database / Connect to Database, or the following icon:

in the Database toolbar. Alternatively the DB Explorer right-click menu may be used, or the key combination [Shift + Ctrl + C].

New in version 2.5.0.47! It is possible to automatically connect to a database when starting IBExpert. Use the following menu: Database Registration Info / Additional / check: Open database when IBExpert starts.

Should there be any problems connecting to the database, use the IBExpert Services menu item Communication Diagnostics.

An example connecting to a remote database using the IBExpert Database menu item Database Registration Info:

Server = `Remote`
Server Name = `<network name of the server or its ip address>` e.g. `OUR_SERVER`
Protocol = `TCP/IP`
DB File Name = `<path to the db file on the server PC>` e.g. `"D:\Data\MyDB.fdb"`

Of course Firebird/InterBase should be installed properly on the server PC (where your database is placed) and the Firebird/InterBase client (`fbclient.dll` or `gds32.dll`) on your local PC.

2.6.1 Accessing a Firebird embedded database with Win1252 (or other character set)

This tip comes from Gerhard Knapp.

In order to connect to a Firebird embedded database with WIN1252 (or other character set) using IBExpert:

- Rename `fbembed.dll` to `fbclient.dll` (always recommendable; not just in this case!).
- Define this `fbclient.dll` including drive and path in the IBExpert Database Registration.
- Specify `WIN1252` in IBExpert.
- Copy the subdirectory `"\intl"` from the Program Files directory, where `fbclient.dll` is installed, into the directory `C:\Program Files\HK-Software\IBExpert 2.0` !!

You should then have no further access problems.

Further information:

- When "fbembed.dll" is renamed "fbclient.dll", it is also a fully-fledged client, i.e. if an application needs to access an embedded database on a Firebird server, the fbclient.dll is more than sufficient.

2.6.2 Database login

If a password is not entered at the time of registering the database (see Register Database), it needs to be logged into each time the database is opened.

Specify a username and associated password. If the user is not authorized or the password is not correct, an error message appears.

Optionally, a role may be specified. If the role has previously been GRANTed to the username, all access privileges assigned to that role for the duration of the current session apply for that user.

If the user is an authorized user for that server, and if the password is correct, access is granted to the database.

2.6.3 Remote database connect using an alias

This article was written by Claudio Valderrama (http://www.cvalde.net/ - The InterBase Unofficial Site), February 2002

Many developers wish to avoid the client having to give the engine the full path of the database in the same machine (node) where the engine runs? It is not only inconvenient when the database's location is changed, it is also a low level that the client shouldn't be concerned about. Finally, many developers have concerns with the security. Ideally, the physical location of the engine and the databases shouldn't be disclosed to the client. Only an Alias should be visible.

It's incredible that for years, a built-in solution in the engine (that works whenever the server is a NT machine) has been lying in the heart of the code and nobody made it public, less even documented in some help file. Perhaps because it unfortunately is a Win32 only solution, nothing that can be used on Linux, so the location of a gdb is not truly transparent.

The syntax is very simple. It has the form:

```
\\server\!share_name!\database.gdb
```

or the form

```
server:\!share_name!\database.gdb
```

It's not a true alias, since you still know the name of the database and of course, the server machine should be known. But it helps if you need to move the database around NT servers, without having to change configuration files or recompiling programs. Here, "server" is the NetBEUI name of the NT machine, followed by the pseudo-UNC paths that IB/FB uses. Alternatively, "server" is the TCP/IP name of the NT machine, but followed by backslashes, not the typical slashes the IB's TCP syntax uses. (Really, using slashes or backslashes is not important in a typical full path, since the engine makes the adjustments, but in this case, the syntax to recognize the share demands backslashes.) The difference is that instead of a full path inside the server, a share's name in the server is used, surrounded by exclamation marks. This share points in turn to the full path of the database, so you only have to append the database's name. It has nothing do to with client-side mappings.

How it works: the client library recognizes a UNC-like path and knows it's NetBEUI. Otherwise, it recognizes a TCP-like syntax thanks to the colon. Then it connects to the required server with the right network protocol and passes the remnant of the path, stripping the server's name. A routine inside the engine, named "expand_share_name", will look for the backslash followed by the exclamation mark, then if a matching "!\" occurs, it takes the name inside the two pairs ("\!" and "!\") and will open the registry (RegOpenKeyEx) at

```
SYSTEM\\CurrentControlSet\\Services\\LanmanServer\\Shares
```

to extract the data (RegQueryValueEx) in the value <share_name>, that's supposedly the name of a registered share in the server machine. It proceeds to decode the data and gets the "Path" component inside the multi-string data that's the physical path. It loads this path in its argument and returns to the caller that will continue testing to see finally if the database's name is valid and exists.

For example, given a share's name "myshare", the registry key shown above contains a list of values that denote shares. You can find there the implicit ones such as IAS1$ (very bad, get rid of it since it points to the IIS admin dir), the NETLOGON share and "myshare". Reading the data in the value "myshare", the following can be seen:

```
MaxUses=4294967295.Path=H:\PROY.Permissions=127.Remark=for fb.Type=0..
```

The dots denote the NULL ASCII value, since this is a multi-string. The engine looks for "path" and gets the string that follows, namely H:\PROY, then appends the backlash if missing. Hence, the engine uses information in the server itself to decode the full path. This path will prefix the database name when the function expand_share_name returns to the caller.

An advantage is that you don't need to grant permissions on this share. You can deny anyone any right (even if NT prompts if you are sure) and you can go further: you can stop the service responsible for handling requests of NetBEUI shares. The engine reads the registry directly, so it doesn't query the network layer. It's a true hack, a commodity to avoid the inclusion of hard-coded paths in the client. If you want to change it, just change the share's information, without granting anyone any right on the share.

Since the engine reads that registry location each time a connection string should be analyzed, it will get the changed name in the next attachment request. If you disabled some networks services, so that changing the share is not possible through high level interfaces, you can edit the registry directly and change the path. Beware that the each dot represents a NULL ASCII value in the example shown above, so your path should end with that value. An even nicer feature is that this works:

```
H:\ibdev\fbbuild\interbase\jrd>isql \\atenea\!myshare!\g
Database: \\atenea\!myshare!\g
SQL> ^Z
```

but it's not restricted to NetBEUI. Indeed, as noted before, you can use TCP syntax:

```
H:\ibdev\fbbuild\interbase\jrd>isql localhost:\!myshare!\g
Database: localhost:\!myshare!\g
SQL> ^Z
```

(Remember that there's no restriction to the name of a gdb other than the file name conventions in the platform where the engine resides. In this case, it's simply named "g", although an extension helps the database admin.)

There are a couple of drawbacks: first, this hack is tied to Win32. (Furthermore, I don't have a way to test it on XP, but I've been informed of success with Windows 2000.) Second, when I read that internal function expand_share_name(), I found a possible buffer overrun and closed it. Revisiting the code when I wrote this article, I found a registry key handle that wasn't closed if the function gives up prematurely for lack of RAM. (I solved this second glitch in Firebird at the time I was finishing this article.) Hence, I believe the lack of documentation comes from the untested nature of the facility.

2.7 Reconnect to Database

This menu item is useful should a database connection have accidentally been disconnected (this may happen sometimes with a remote connection).

The reconnection can be simply made either using the Menu Database / Reconnect Database, or the following icon:

in the Database toolbar. Alternatively the DB Explorer right-click menu may be used.

Should there be any problems reconnecting to the database, go to the Database Registration Info and perform a Test Connect.

2.8 Disconnect from a Database

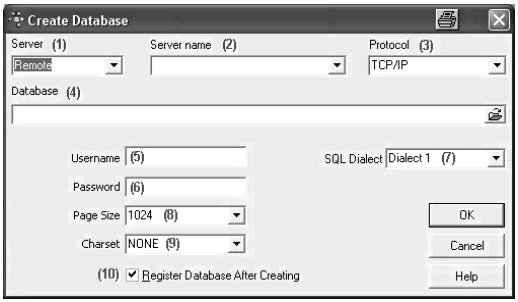
When you have finished working with a database it can be disconnected using the IBExpert menu item Database / Disconnect from Database, or the following icon:

in the Database toolbar. Alternatively the DB Explorer right-click menu may be used, or the key combination [Shift + Ctrl + D].

It is not necessary to disconnect all databases manually when you have finished working with IBExpert. IBExpert does this automatically when it closes down.

2.9 Create Database

A new database can be created by simply using the IBExpert menu item Database / Create Database... or using the respective icon in the Database toolbar. The Create Database dialog appears:



(1) Server: first the server which is to store the database needs to be specified. This can be local or remote.

- Remote - the remote connection needs to be defined by specifying **(2)** Server name and **(3)** Protocol. The pull-down list shows all servers previously connected to/from this workstation/PC.
- Local - LOCALHOST (own Server). To create a new database on the same machine where IBExpert is in use, you do not need to enter a server name.

The DOS `'PING LOCAL HOST'` or `'PING SRVNAME'` command shows the path if unknown (it is not necessary to know which operating system is running or where this server is). By specifying a local server, fields **(2)** and **(3)** are automatically blended out, as they are in this case irrelevant.

(2) Server name: must be known when accessing remotely. The following syntax should be used:

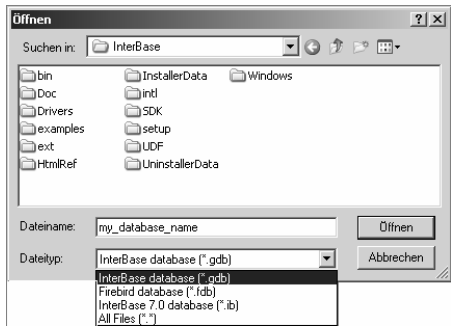
- Windows `SERVER_NAME:C:\path\database.gdb`
- Linux `SERVER_NAME:/path/database.gdb`

The standard port for InterBase and Firebird is `3050`. However this is sometimes altered for obvious reasons of security, or when other databases are already using this port. If a different port is to be used for the InterBase/Firebird connection, the port number needs to be included as part of the server name. For example, if port number `3055` is to be used, the server name is `SERVER/3055`.

(3) Protocol: a pull-down list of three options: TCP/IP, NetBEUI or SPX. As a rule we recommend you always use TCP/IP (worldwide standard).

- SPX used to be used by Novell; now even Novell supports TCP/IP.
- NetBEUI - is not really a network protocol, it simply accesses the line. It is slow as it makes everything available everywhere and anyone can access the information. This is also purely a Windows protocol. Note: in DOS the TRACERT command lists the protocol route. TCP/IP intelligently takes another direction, if one or part of the lines on the quickest route is blocked or down.

(4) Database: by clicking on the folder icon to the right of this field, the path can easily found and specified, the database name entered, and the suffix selected from the pull-down list. The database name must always be specified with the drive and path when creating a database. Please note that the database file for a Windows server must be on a physical drive on the server, because InterBase/Firebird does not support databases on mapped drive letters. The database suffixes do not have to adhere to the forms offered in the list.



(5) User Name: Only those names may be entered when creating a database, which already exist in the server security database ISC4.GDB (which stores server rights; user rights for the database objects are stored in the database itself). The person creating the database becomes the database owner. Only the database owner and the SYSDBA (System Database Administrator) are allowed to perform certain operations upon the database (such as a database shutdown). Therefore if the database owner is defined as the SYSDBA, this is the only person entitled to perform these operations. Note: when a role with the name SYSDBA is created, no other users (not even the SYSDBA) can access the database.

(6) Password: The passwords are encrypted in the ISC4.GDB. If you insist upon using the SYSDBA name as the database owner, at least change the standard password (*masterkey*) to ensure at least some degree of security! The *masterkey* password should be changed as soon as possible after creating the database.

InterBase verifies only the first 8 characters of a password, even if a longer word is entered, i.e. in the case of the *masterkey* password only "*masterke*" is verified. All characters following the 8th are ignored.

(7) SQL Dialect: Here Dialect 1 (up to and including InterBase 5) or 3 (InterBase 6/Firebird) needs to be specified. For more information regarding this subject, please refer to SQL Dialect.

(8) Page size: Specifies the database page size in bytes. For more information regarding this subject, please refer to Page Size.

(9) Charset: Here the default character set can be defined. This character set is useful, when the database created is to be used for foreign languages as it is applicable for all areas of the database unless overridden by the domain or field definition. If not specified, the parameter defaults to NONE, i.e. values are stored exactly as typed. For more information regarding this subject, please refer to Charset/Default Character Set.

(10) Register Database After Creating: This checkbox automatically generates the Database Registration dialog so that the database can be registered. Registration is necessary, so that IBExpert recognizes that a database is present. The Register Database dialog however offers many further options. We recommend clicking this checkbox (the default setting), so that the database is registered immediately after creation. If the database is not registered at the time of creation, it cannot be seen in the DB Explorer of the left of the IBExpert screen. This means that the user must know exactly where the new database can be found (i.e. which server, path, name etc.) when registering at a later date.

Tip: IBExpert recommends creating a User Database - please refer to Environment Options / IBExpert User Database for further information.

For those preferring SQL, the syntax is as follows:

```
CREATE {DATABASE | SCHEMA} 'filespec'
[USER 'username' [PASSWORD 'password']]
[PAGE_SIZE [=] int]
[LENGTH [=] int [PAGE[S]]]
[DEFAULT CHARACTER SET charset]
[secondary_file];
<secondary_file> = FILE 'filespec' [fileinfo] [secondary_file]
<fileinfo> = [LENGTH [=] int [PAGE[S]] | STARTING [AT [PAGE]] int }
[fileinfo]
```

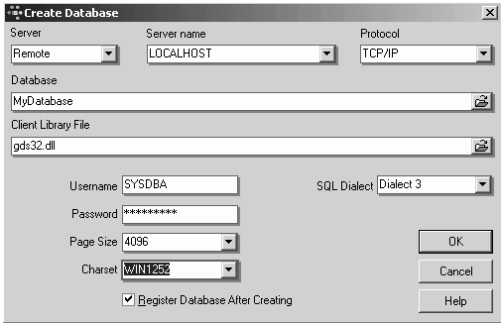
For example:

```
CREATE DATABASE 'C:\DATABASEFILES\employee.gdb'
DEFAULT CHARACTER SET ISO8859_1
FILE 'employee2.gdb' STARTING AT PAGE 10001;
```

2.9.1 Charset / Default Character Set

The default character set is the character set defined when creating the database, and applicable for all areas of the database unless overridden by the domain or field definition. It controls not only the available characters that can be stored and displayed, but also the collation order. If not specified, the parameter defaults to NONE, i.e. values are stored exactly as typed.

InterBase/Firebird supports multiple character sets for use around the world. If no special character set is specified for individual columns, the database default character set is assumed. The default character set is defined in IBExpert in the Create Database dialog:



If a character set is defined as the default character set when creating the database, it is not necessary to define this again for individual columns.

InterBase/Firebird supports more that 20 different character sets directly. The chosen character set is also of importance when importing and exporting data with different character sets. This needs to be taken into consideration when applications are developed with multiple language versions.

The ASCII character set is not synonymous with a non-defined character set. If no character set is defined, Firebird/InterBase chooses the character set NONE. The character set NONE does not translate characters. Umlauts and accents are not sorted correctly. When the ASCII character set is specified, all characters are translated into the ASCII equivalents from the character set under which they were input.

The character set WIN 1252 is recommended for European countries, as it includes all characters and collation orders of the most important European languages.

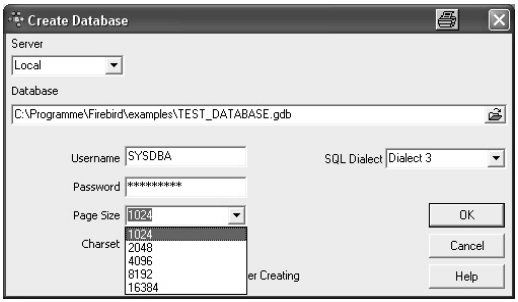
Generally this default character set cannot be altered at a later date (only using the command line tools IBExtract and IBExpert Script). Alternate character sets can however be defined for individual domains and tables, which override the default character set.

2.9.2 Page Size

This is the specification of the database page size in bytes.

Firebird/InterBase databases are saved in blocks. Each of these blocks is called a page. Database administration occurs basically by accessing the hard drive block by block. The more data per access fetched by a single database page, the less often it is necessary to load a new page, at least theoretically. Practically, depending upon the operating system and server hardware, access to larger database pages can even influence the performance negatively, as 1024 bytes can be loaded quicker than 8192 bytes.

When creating a database (IBExpert menu item Database / Create Database) the standard database page size of 1024 bytes is the default value. This is also the smallest unit. Further values permitted are 2048, 4096, 8192 and 16384.



A large page size has certain advantages in the following situations:

1. Many index-based operations (indices work quicker if the index depth is minimized).

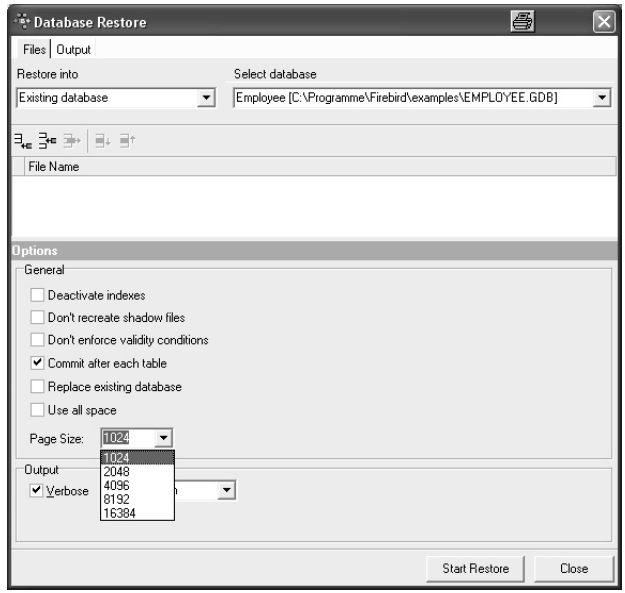
2. Wide records, because with very wide data structures, i.e. with very many and/or very long columns, reading a data set is more efficient. With data sets that do not fit onto one page, several pages have to be read to fetch a single data set. The same applies to writing; ie. fetches across several pages are necessary.

3. Large blob fields, as data is stored and retrieved more efficiently if fewer pages need to be fetched. This is because blob columns, assuming the data contained fits onto one page, are stored in the data pages with other data, and so when one data set is fetched, the data from the standard field and the blob columns are also fetched. With larger blobs the writing and reading processes are also more effective, as, for example, 100 accesses are necessary for a 100k blob column with a 1k page size. However with an 8k page size only 13 accesses are required.

A small page size is sufficient if many transactions return only a small number of rows. Slim table structures with small database pages can be accessed more quickly for reading and writing as less memory is required, and more database pages can be held in the cache.

Creating a database with a page size of 4096 can be viewed as optimal, as this is the Windows block size. Therefore smaller page sizes do not bring any advantages, as Windows will still fetch 4K blocks.

The only way to subsequently alter a database page size, is to perform a database backup followed by a restore (IBExpert menu: Services / Restore Database) where the database page size can be redefined.



2.9.3 Structure of a data page

By Paul Beach
(With thanks to Dave Schnepper and Deej Bredenberg)

A database is considered to be a collection of pages, each page has a pre-defined size [see page size], this size is determined when the database is created [see Create Database] by a database parameter that is passed in the `isc_database_create` call (`gds_dpb_page_size`). Pages are identified by a page number (4 byte unsigned integer), starting at 0 and increasing sequentially from the beginning of the first database file to the end of the last database file.

Page 0 of a database is always the database header page, which contains the information that is needed when you attach to a database. Page 1 is the first PIP page (Page Inventory Page) and the first WAL page is always page 2. By convention, page 3 is the first pointer page for the RDB$PAGES relation, but that location is described on the header page so it could (in theory) change.

Except for the header page there is no specific relationship between a page number and the type of data that could be stored on it.

The types of pages are defined in **ods.h** and are as follows:

```
#define pag_header 1          /* Database header page */
#define pag_pages 2           /* Page inventory page */
#define pag_transactions 3    /* Transaction inventory page */
#define pag_pointer 4         /* Pointer page */
#define pag_data 5            /* Data page */
#define pag_root 6            /* Index root page */
#define pag_index 7           /* Index (B-tree) page */
#define pag_blob 8            /* Blob data page */
#define pag_ids  9            /* Gen-ids */
#define pag_log  10           /* Write ahead log information */
```

Pages are located in the database by seeking within the database file to position `page_number*bytes_per_page`. The structure of a data page, as defined in `ods.h` is as follows:

All pages have a page header, the page header consists of,

```
typedef struct pag {
   SCHAR pag_type;
   SCHAR pag_flags;
   USHORT pag_checksum;
   ULONG pag_generation;
   ULONG pag_seqno;     /* WAL seqno of last update */
   ULONG pag_offset;    /* WAL offset of last update */
} *PAG
```

1	2	Length, bytes	Description
pag_type	Page Type	1	=pag_data
pag_flags	Page Flags	1	e.g. Data page is orphaned (it doesn't appear on any pointer page), Page is full, or a blob or an array exist on the page.
pag_checksum	Page Checksum	2	Always 12345 for known versions
pag_generation	Page Generation	4	how many times has the page been updated.
pag_seqno	Page Sequence Number	4	WAL sequence number of last update, unused.
pag_offset	Page Offset	4	WAL offset of last update, unused.

The remainder of the page (less the 16 bytes above) is used to store page specific data.

A data page holds the actual data for a table, and a data page can only be used by a single table, i.e. it is not possible for data from two different tables to appear on the

same data page. Each data page holds what is basically an Array of records (complete or fragmented). Below the header is 8 bytes of :

- Page Sequence (`dpg_sequence` 4 bytes) sequence number of the data page in a table, used for integrity checking.
- Page's Table/Relation id (`dpg_relation` 2 bytes) this id is also used for integrity checking.
- Number of Records or record fragments that exist on the data page (`dpg_count` 2 bytes).

This is then followed by an Array of descriptors each of the format: offset of record or fragment, length of record or fragment. This descriptor describes the size and location of records or fragments stored on a page. For each record or fragment that is stored on the page there is an equivalent record descriptor at the top of the page. As records get stored the array grows down the page, whilst the records or fragments are inserted backwards from the end of the page. The page is full when they meet in the middle.

```
typedef struct dpg {
    struct pag dpg_header;
    SLONG dpg_sequence;    /* Sequence number in relation */
    USHORT dpg_relation;   /* Relation id */
    USHORT dpg_count;      /* Number of record segments on page */
    struct dpg_repeat
    {
    USHORT dpg_offset;     /* Offset of record fragment */
    USHORT dpg_length;     /* Length of record fragment */
    }  dpg_rpt [1];
} *DPG;
```

Obviously data records can vary in size, so the number of records that may fit on a page can vary. Equally records may get deleted, leaving gaps on a page.

The page free space calculation works by looking at the size of all of the records that exist on a page. If space can be created on the page for a new record, then the records will get compressed i.e. shifted downwards to fill the gaps that would get created during normal insert, update and deletion of data. When the free space is less than the size of the smallest possible fragment - then the page is full.

A record may be uniquely identified by its record number (`rdb$db_key`).

The record header structure is,

1	Length, bytes	Description
rhd_transaction	4	Record header transaction. The transaction id that wrote the record.
rhd_b_page	4	Record header back pointer. Page number of the back version of the record.
rhd_b_line	2	Record header back line. Line number of the back version of the record.
rhd_flags	2	Record header flags. Possible flags are: rhd_deleted - the record has been logically deleted, but hasn't yet been garbage collected. rhd_chain - this record is an old version, a later version points backwards to this one. rhd_fragment - the record is a fragment of a record rhd_incomplete - the initial part of the record is stored here, but the rest of it may be stored in one or multiple fragments rhd_blob - the record stores data from a blob rhd_stream_blob - the record stores data from a stream blob rhd_delta - the prior version of this record must be obtained by applying the differences to the data stored in this array rhd_large - this is a large record object such as a blob or an array rhd_damaged - the record is known to be corrupt rhd_gc_active - the record is being garbage collected as an unrequired record version
rhd_format	1	Record header format. The metadata version of the stored record. When a record is stored or updated, it is marked with the current format number for that table. A format is a description of the number and physical order of fields in a table and the datatype of each field

When a field is added or dropped, or the data type of a field is changed, a new format is generated for that table. A history of all of the formats for a table is stored in RDB$FORMATS. This allows the database to reconstruct records that were stored at any time based on the format that existed for the table at that time. Metadata changes, such as the above do not directly affect the records when the metadata change itself takes place, only when the records are actually next visited.

Record header data (rhd_data size n as needed) is the actual record data and is compressed by RLE (Run Length Encoding). When a run takes place the compression algorithm will use 1 extra byte per 128 bytes, to represent the run length followed by one or more bytes of data. A positive run length indicates that the next sequence of bytes should be read literally, whilst a negative run length indicates that the following byte is to be repeated ABS(n) times.

```
typedef struct rhd {
    SLONG rhd_transaction;    /* transaction id */
    SLONG rhd_b_page;         /* back pointer */
    USHORT rhd_b_line;        /* back line */
    USHORT rhd_flags;         /* flags, etc */
    UCHAR rhd_format;         /* format version */
    UCHAR rhd_data [1];
} *RHD;
```

This paper was written by Paul Beach in September 2001, and is copyright Paul Beach and IBPhoenix Inc.

2.9.4 SQL Dialect

Structured Query Language is a language for relational databases, which serves to define, manipulate, find and fetch data in a database.

There are currently two SQL dialects used with InterBase and Firebird:

Dialect 1 = database performance is fully compatible to InterBase 5.x (e.g. numeric up to 15 digits)
Dialect 3 = all new functions in InterBase 6 and upwards with SQL 92 features are available (e.g. numeric up to 18 digits)

For those that work with the BDE, this can only work with dialect 1 up to and including Delphi 6 (i.e. dialect 3 from Delphi 7 onwards).

Differences between dialects 1 and 3 include the numeric (15 or 18), and also for example date:

Dialect 1 = Date includes the date and time
Dialect 3 = Date = date, time = time, timestamp = date and time.

For new projects it is recommended that dialect 3 be specified.

Occasionally the question arises\"What about SQL Dialect 2?". Dialect 2 is similar to dialect 1, generates however warnings for all objects that are incompatible to Dialect 3 (i.e. only suitable for the client end); therefore, in principle, not really of importance.

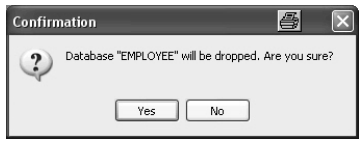
The SQL dialect to be used in a database is specified when creating the database (IBExpert menu: Database / Create Database). It can subsequently be altered using the IBExpert menu Services / Database Properties (although watch out for possible dialect incongruencies, for example, the different date and time types).

2.10 Drop Database/Delete Database

Databases can be dropped in IBExpert using the menu item Database / Drop Database.

When an InterBase/Firebird database is dropped, all the metadata and data for this database are also deleted, along with all its secondary, shadow and log files ...permanently!

IBExpert asks for confirmation:



before finally dropping the database. Once dropped, it cannot be retrieved, so be extremely careful when using this command.

For those users preferring direct SQL input, the syntax is:

```
DROP DATABASE;
```

A database may only be dropped by its creator or the SYSDBA.

2.11 Recreate Database

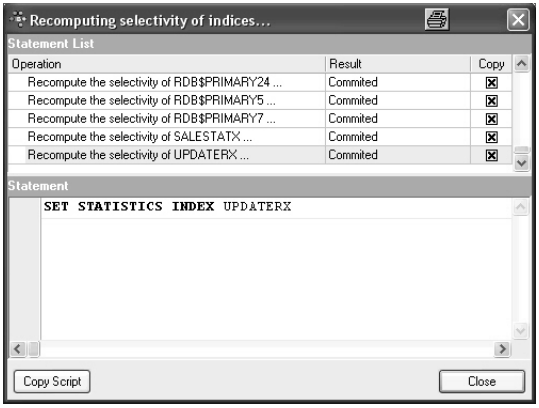
This new IBExpert menu item, Recreate Database was introduced in IBExpert version 2004.9.12.1. This drops the database, along with all its contents, and creates it again without the metadata and data content (after confirmation, of course) using the parameters of the database just dropped. The parameters are: server name, protocol, user name, password, page size, SQL dialect and default character set.

2.12 Recompute selectivity of all indices

Indices statistics are used by the InterBase/Firebird Optimizer, to determine which index is the most efficient. All statistics are recalculated only when a database is restored after backing up, or when this is explicitly requested by the developer.

When an index is initially created, its statistical value is 0. Therefore it is extremely important, particularly with new databases where the first data sets are being entered, to regularly explicitly recompute the selectivity, so that the optimizer can recognize the most efficient indices. This is not so important with databases, where little data manipulation occurs, as the selectivity will change very little.

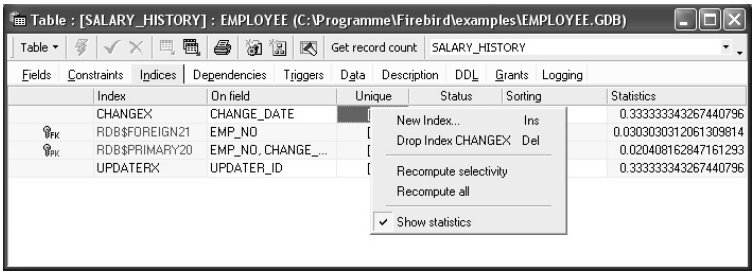
To recompute the selectivity of all indices use the IBExpert menu item Recompute Selectivity of all Indices. This recomputes the selectivity for all indices in the database and can be found in the IBExpert Database menu or using the right mouse button in the DB Explorer.



Individual indices can be recomputed directly in the SQL Editor using the command:

```
SET STATISTICS INDEX <index_name>;
```

Single or multiple indices can also be recomputed directly in the Table Editor / Indices page, using the right-click menu.



The same Recomputing Selectivity dialog as above is then displayed.

The new statistical values can be viewed for individual tables in the Table Editor on the Indices page (providing the Statistics are blended in using the right-click menu item Show Statistics).

2.13 Recompile all Stored Procedures and Triggers

Stored procedures and triggers use indices internally. The Recompile command ensures that the most up-to-date indices are used. Using this command it is also possible to recognize when one procedure or trigger calls another.

This is also useful, for example, when backing up an older InterBase version (e.g. v5) and restoring in a newer version, such as InterBase 6 or Firebird 1.5, as InterBase/Firebird simply copies the data and metadata into the new version when restoring. So that if a variable name that is a keyword in the stored procedure is wrong, it is not recognized, as the compiler does not recognize variable names as such. When however procedures and triggers are recompiled, any such problems are discovered.

The menu items, Recompile all Stored Procedures and Recompile all Triggers can be found in the IBExpert Database menu or using the right-click menu in the DB Explorer.

2.14 Database Security

Please refer to the following subjects, for further information regarding database security:

- Server Security `ISC4.GDB / SECURITY.FDB`,
- User Manager,
- Grant Manager.

2.15 Database Corruption

The following articles provide important information regarding the causes leading to database corruption, as well as ways to recover a corrupt database. We would like to thank the authors for allowing us to publish their articles here.

2.15.1 How to corrupt a database

Although Firebird is extremely stable and secure, there are a few things that you should *NOT* do, as these could result in corrupting the database!

The following tips have been taken from the *Firebird Quick Start Guide, © IBPhoenix Publications 2002,2003.* Many thanks to Paul Beach (www.ibphoenix.com)!

Modifying metadata tables

Firebird stores and maintains all of the metadata for its own and your user-defined objects in a Firebird database! More precisely, it stores them in relations (tables) right in the database itself. The identifiers for the system tables, their columns and several other types of system objects begin with the characters 'RDB$'.

Because these are ordinary database objects, they can be queried and manipulated just like your user-defined objects. However, just because you *can* does not say you *should*. The Firebird engine implements a high-level subset of SQL (DDL - please refer to Data Definition Language for further information) for the purpose of defining and operating on metadata objects, typically through CREATE, ALTER and DROP statements.

It cannot be recommended too strongly that you use DDL - not direct SQL operations on the system tables - whenever you need to alter or remove metadata. Defer the 'hot fix'" stuff until your skills in SQL and your knowledge of the Firebird engine become very advanced. A wrecked database is neither pretty to behold nor cheap to repair.

Source: *Firebird Quick Start Guide, © IBPhoenix Publications 2002,2003*

Disabling forced writes

Firebird is installed with forced writes (synchronous writes) enabled by default. Changed and new data are written to disk immediately upon posting.

It is possible to configure a database to use asynchronous data writes - whereby modified or new data are held in the memory cache for periodic flushing to disk by the operating system's I/O subsystem. The common term for this configuration is forced writes off (or disabled). It is sometimes resorted to in order to improve performance during large batch operations.

The big warning here is - *do not disable forced writes on a Windows server*. It has been observed that the Windows server platforms do not flush the write cache until the Firebird service is shut down. Apart from power interruptions, there is just too much that can go wrong on a Windows server. If it should hang, the I/O system goes out of reach and your users' work will be lost in the process of rebooting.

- Windows 9x and ME do not support deferred data writes

Disabling Forced Writes on a Linux server

Linux servers are safer for running an operation with forced writes disabled temporarily. Do not leave it disabled once your large batch task is completed, unless you have a very robust fall-back power system.

Source: *Firebird Quick Start Guide, © IBPhoenix Publications 2002,2003*

Restoring a backup to a running database

One of the restore options in the GBAK utility (`gbak -r[estore]`) allows you to restore a gbak file over the top of an existing database. It is possible for this style of restore to proceed without warning while users are logged in to the database. Database corruption is almost certain to be the result.

- Be aware that you will need to design your Admin tools and procedures to prevent any possibility for any user (including SYSDBA) to restore to your active database if any users are logged in.

If is practicable to do so, it is recommended to restore to spare disk space using the `gbak -c[reate]` option and *test the restored database* using isql [or IBExpert]. If the restored database is good, shut down the server. Make a file system copy of the old database and then copy the restored database file (or files) over their existing counterparts.

Source: *Firebird Quick Start Guide, © IBPhoenix Publications 2002,2003*

Allowing users to log in during a restore

If you do not block access to users while performing a restore using `gbak -r[estore]` then users may be able to log in and attempt to do operations on data. Corrupted structures will result.

Source: *Firebird Quick Start Guide, © IBPhoenix Publications 2002,2003*

2.15.2 Recovering corrupt databases

The following is an excerpt from the successful Russian book, "The InterBase World" first published in September 2002, with a second edition following in April 2003. The authors of the book are Alexey Kovyazin, developer of IBSurgeon (www.ibsurgeon.com) and well-known Russian InterBase specialist, and Serg Vostrikov, CEO of the Devrace company (www.devrace.com).

Here the authors would like to offer you a draft copy of one chapter of this book devoted to recovery of InterBase/Firebird databases.

They would like to pass on their thanks to all who helped create this guide: Craig Stuntz, Alexander Nevsky, Konstantin Sipachev, Tatjana Sipacheva and all the other kind and knowledgeable members of the InterBase and Firebird community.

Main causes of database corruption

Unfortunately there is always a probability that any information stored will be corrupted and some of this information will be lost. Databases are not an exception to this

rule. In this chapter we will consider the principal causes that lead to InterBase/Firebird database corruption, some methods of repairing databases and extracting information from them. We will also make recommendations and offer precautions that will minimize the probability of information loss.

First of all, if we speak about database repair we should perhaps first define 'database corruption'. A database is usually described as damaged if, when trying to extract or modify some information, errors appear and/or the information to be extracted turns out to be lost, incomplete or incorrect. There are cases when database corruption is hidden and can only be found by testing with special facilities. However there are also real database corruptions, when it is impossible to connect to the database, when adjusted programs send strange errors to the clients (without any data manipulation having occurred), or when it is impossible to restore the database from a backup copy.

Principal causes of database corruption are:

- Abnormal termination of the server computer, especially an electrical power interruption. For the IT-industry it can be a real blow and that is why we hope there is no need to remind you once again about the necessity of having a source of uninterrupted power supply on your server.
- Defects and faults on the server computer, especially the HDD (hard disk drive), disk controllers, the computer's main memory and the cache memory of Raid controllers.
- An incorrect connection string to a multi-client database with one or more users (in versions prior to 6.x). When connecting via TCP/IP, the path to the database must be pointed to a server name:
  `drive:/path/databasename /`
  For servers on UNIX platforms: `servername: /path/databasename/`
  Using a NetBEUI protocol: `\\servername\drive:\path\databasename`.
  Even when connecting to a database from the computer, on which the database is located and where the server is running, the same specification should be used, renaming the servername as `localhost`. It is not possible to use mapped drives in the connection specification. If you break one of these rules, the server thinks that it is working with different databases and database corruption is guaranteed.
- File copy or other file access to the database when the server is running. The execution of the command 'shutdown', or disconnecting the users in the usual way is not a guarantee that the server is doing nothing with the database. If the sweep interval is not set to 0, garbage collection may be being executed. Generally the garbage collection is executed immediately after the last user disconnects from the database. Usually it takes several seconds, but if many DELETE or UPDATE operations were committed before it, the process may take longer.
- Using unstable InterBase server versions 5.1-5.5. The Borland Company officially admitted that there were several errors in these servers and these were removed in the stable upgrade 5.6 only after the release of certified InterBase 6 was in free-running mode for all clients of servers 5.1-5.5 on its site.
- Exceeding size restriction of a database file. At the time of writing this, for most existing UNIX platform servers the limit is 2 GB, for Windows NT/2000 - 4 GB, but it is recommended to assume 2 GB. When the database size is approaching its limit, an additional file must be created.
- Exhaustion of free disk space when working with the database.

- For Borland InterBase servers using versions under 6.0.1.6 - exceeding the restriction of the maximum number of generators, according to Borland InterBase R & D defined as follows (please refer to table 1 below).

Critical number of generators in early InterBase versions				
Version	**Page size=1024**	**Page size=2048**	**Page size=4096**	**Page size=8192**
Pre 6	248	504	1016	2040
6.0.x	124	257	508	1020

- For all Borland InterBase servers - exceeding the permissible number of transactions without executing a backup/restore. The number of transactions that have been made in the database since the last backup and restore can be determined by invoking the utility `GSTAT` with the key `-h` parameter `NEXT TRANSACTION ID`. According to Ann W. Harrison, the critical number of transactions depends on the page size, and has the following values (please refer to table 2 below):

Critical number of transactions in Borland InterBase servers	
Database page size	**Critical number of transactions**
1024 byte	131 596 287
2048 byte	265 814 016
4096 byte	534 249 472
8192 byte	1 071 120 384

The constraints of Borland InterBase servers enumerated above are not applicable to Firebird servers except for the earliest versions 0.x, the existence of which has already become history. If you use the final version Firebird 1.0 or above, or InterBase 6.5-7.x, you should not worry about points 5, 6, 8 and 9 and should instead concentrate your efforts on other causes. Now we will consider the most frequent of these in detail.

Power supply failure

When shutting off the power on the server, all data processing activities are interrupted in the most unexpected and (according to Murphy's law) dangerous places. As a result the information in the database may be distorted or lost. The simplest case is when all uncommitted data from a client's applications are lost as a result of an emergency server shutdown. After a power-cut restart the server. This analyzes the data, makes a note of incomplete transactions related to none of the clients, and cancels all modifications made within the bounds of these «dead» transactions. Actually such behavior is normal and assumed from the start by InterBase developers.

However power supply interruption is not always followed just by such insignificant losses. If the server was executing a database extension at the moment of power supply interruption, there is a large probability of orphan pages present in the database file (pages that are physically allocated and registered on the page inventory page (PIP), upon which it is however impossible to write data).

Only `GFIX`, the repair and modification tool (we will consider it further on), is able to combat orphan pages in the database file. Actually orphan pages lead to unnecessary use of disk space and, as such, are not the cause of data loss or corruption. Power loss leads to more serious damages. For example, after shutting off the power and restarting, a great amount of data, including committed data, may be lost (after adding or modification of which the command «commit transaction» was executed). This happens because confirmed data is not written immediately to the database file on disk. The file cache of the operating system (OS) is used for this purpose. The server process gives

the data write command to the OS. Then the OS assures the server that all the data has been saved to disk although in reality the data is initially stored in the file cache. The OS doesn't hurry to save this data to disk, because it assumes that there is a lot of main memory left, and therefore delays the slow operation of writing to disk until the main memory is full. Please refer to the next subject - Forced Writes - cuts both ways - for further information.

Forced writes – cuts both ways

In order to influence this situation, tuning of the data write mode is provided in InterBase 6 and Firebird. This parameter is called `FORCED WRITES` (FW) and has 2 modes - `ON` (synchronous) and `OFF` (asynchronous). FW modes define how InterBase/Firebird communicates with the disk. If FW is turned on, the setting of synchronous writes to disk is switched on, and confirmed data is written to disk immediately following the `COMMIT` command, the server waits for writing completion and only then continues processing. If FW is switched off InterBase doesn't hurry to write data to disk after a transaction is committed, and delegates this task to a parallel thread, while the main thread continues data processing, not waiting until all writes are written to disk.

Synchronous writes mode is one of the most careful options and it minimizes any possible data loss. However it may cause some loss of performance. Asynchronous writes mode increases the probability of loss of a great quantity of data. In order to achieve maximum performance FW Off mode is usually set. But as a result of power interruption a much higher quantity of data is lost using the asynchronous writes mode than when using the synchronous mode. When setting the write mode you should decide whether a few percentage points of performance are more significant than a few hours of work should power be interrupted unexpectedly.

Very often users are careless with InterBase. Small organizations save on any trifle, often on the computer server, where the DBMS server and different server programs (not only server) are installed and running as well. If they hang-up people don't think for long, and simply press RESET (it happens several times a day). Although InterBase is very stable with regard to such activities compared with other DBMS, and allows work with the database to start immediately after an emergency reboot, such a procedure is not recommended. The number of orphan pages increases and data lose connections among themselves as a result of faulty reboots. It may still function and continue for a long time, but sooner or later it will come to an end. When damaged pages appear among PIP or generator pages, or if the database header page is corrupted, the database may never open again and become a big chunk of separate data from which it is impossible to extract a single byte of useful information.

Corruption of the hard disk

Hard disk corruptions lead to the loss of important database system pages and/or the corruption of links among the remaining pages. Such corruptions are one of the most difficult cases, because they almost always require low-level interference to restore the database.

Database design mistakes

It is necessary to learn of some mistakes made by database developers that can lead to an impossible database recovery from a backup copy (`*.gbk` files created by the

GBAK program). First of all a careless use of Constraints at database level. A typical example is the constraint NOT NULL. Let's suppose that we have a table filled with a number of records. Now using the ALTER TABLE command we'll add one more column to this table and specify that it mustn't contain the non-defined value NULL. Something like this:

```
ALTER TABLE sometable Field/INTEGER NOT NULL
```

In this case there will be no server error as should be expected. This metadata modification will be committed and we won't receive any error or warning message, which creates an illusion of normality.

However, if we Backup the database and try to restore it from the backup copy, we'll receive an error message at the phase of restoring (because NULLs are inserted into the column that has NOT NULL constraint, and the process of restoring will be interrupted. (An important note provided by Craig Stuntz: with version InterBase 7.1 constraints are ignored by default during a restore (this can be controlled by a command-line switch) and nearly any non-corrupt backup can be restored. It's always a good idea to do a test restore after performing a backup, but this problem should pretty much disappear in version 7.1.). This backup copy can't be restored. If the restore was directed to a file having the same name as the existing database (during restoration of the existing database the working file was being rewritten), we'll lose all information.

It has to do with the fact that NOT NULL constraints are implemented by system Triggers which check only incoming data. During restoration, data from the backup copy is inserted into the empty, newly created tables - here we can find inadmissible NULLs in the column with the constraint NOT NULL.

Some developers consider such InterBase behavior to be incorrect, but others will be unable to add a field with NOT NULL restriction to the database table.

The question about required value by default and filling with this value at the moment of creation was widely discussed by Firebird architects, but it wasn't accepted because of the fact that the programmer is obviously going to fill it according to an algorithm, which is rather complicated and maybe iterative. But there is no guarantee, whether he'll be able to distinguish the records ignored by previous iteration from unfilled records or not.

A similar problem can be caused by a garbage collection fault, caused by the specification of an incorrect path to the database (the cause of corruption 3) at the time of connection, and file access to database files when the server is working with it (the cause of corruption 4), and records wholly filled with NULLs can appear in some tables. It's very difficult to detect these records, because they don't correspond to integrity control restrictions, and operator Select just doesn't see them, although they get into the backup copy. If it is impossible to restore for this reason, the GFIX utility should be used (see below), to find and delete these records using non-indexed fields as search conditions. After this try to make a backup copy again and restore the database from it. In conclusion we can say that there are a great number of causes of database corruption and you should always be prepared for the worst - that your database could become damaged for one reason or another. You should therefore be prepared at all times to restore and rescue valuable information.

Precautions and methods of repair

And now we shall consider precautions that guarantee Firebird/InterBase database security, as well as methods of repairing damaged databases.

Regular backups

In order to prevent database corruption, backup copies should be created regularly (if you want to know more about backup then please refer to Backup and Restore for further information). It's the most trusted method to prevent and combat database corruption. Only a backup gives 100% guarantee of database security. As described above, it is possible get a useless copy as the result of restoring a backup file (i.e. a copy that can't be restored); that's why restoring a base from the copy should not be performed by writing over the script, and a backup must be carried out according to definite rules. Firstly, a backup should be executed as often as possible, secondly it must be serial and thirdly, backup copies must be checked for their restoring capability.

Usually, a backup means that it's necessary to make a backup copy rather often, for example, once every twenty-four hours. The shorter the period is between database backups, the less data will be lost as a result of a fault. The sequence of backups means that the number of backups should increase and should be stored for at least a week. If possible, backups should be written to special devices such as a streamer, but if this is not possible - copy them to another computer. The history of backup copies will help to discover hidden corruptions and cope with an error that perhaps arose some time ago but has only just showed up unexpectedly. It is necessary to check whether it is possible to restore the saved backup without errors or not. This can be checked in only one way - through the test restore process. It should be mentioned that the restore process takes 3 times longer than the backup, and it’s difficult to execute restore validation every day for large databases, because it may interrupt the users' work for a few hours (a night break may not be enough).

It would be better if big organizations didn't save at the wrong end and assigned one computer just for these purposes.

In this case, if the server must work with a serious load 24 hours 7 days a week, we can use the SHADOW mechanism for taking snapshots of the database, and performing further backup operations from the immediate copy. When creating a backup copy and then restoring the database from this backup, all data in the database is recreated. This process (backup/restore or b/r) contributes to the correction of most non-fatal errors in the database connected with hard disk corruptions, detecting problems with integrity in the database, cleaning the database of garbage (old versions and fragments of records, incomplete transactions) which decreases the database size considerably.

Regular backup/restore is a guarantee of Firebird/InterBase database security. If the database is working, then it is recommended to execute backup/restore on a weekly basis. To tell the truth, there are some examples of Firebird/InterBase databases that are intensively used for some years without a single backup/restore.

Nevertheless, to be on the safe side it's desirable to perform this procedure regularly, especially as it can be easily automated (please refer to Backup and Restore).

If it's impossible to perform a regular backup/restore for certain reasons, then the GFIX tool can be used for checking and restoring the database. GFIX allows you to check and remove many errors without performing a backup/restore.

2

Using GFIX

The command-line utility GFIX is used for checking and restoring databases. Furthermore GFIX can also execute various database control activities: changing the database dialect, setting and canceling the mode 'read-only', setting cache size for a specific database and also some important functions.

GFIX is committed in command-line mode and has the following syntax:

```
Gfix [ options] db_name
```

Options is a set of options for executing GFIX, db_name is the name of the database for which the operations are to be performed, defined by a set of options. The following table displays the GFIX options related to database repair:

GFIX tool options for database restoration

Option	Description
-**f**[ull]	This option is used in combination with -v and means it's time to check all fragments of records
-**i**[gnore]	Option makes GFIX ignore checksum errors at the time of validation or database cleaning
-**m**[end]	Marks damaged records as not available, as a result of which they will be deleted during the following backup/restore. This option is used when preparing a corrupted database for backup/restore.
-**n**[o_update]	Option is used in combination with -v for read-only database validation without correcting corruptions
-**pas**[sword]	Option allows the password to be set when connecting to the database. (Note that there is an error in the InterBase documentation: -pa[ssword], the shortcut "-pa" will not work – you need to use "-pas")
-**user**	Option allows the user's name to be set when connecting to the database
-**v**[alidate]	Option for presetting the database validation when errors are discovered
-**m**[ode]	Option for setting the write mode for the database – for read-only or read/write. This parameter can accept 2 values – read write or read only.
-**w**[rite] {**sync** | **async**}	Option that switches on and off the mode synchronous/ asynchronous forced writes to database. sync – to turn synchronous writes on (FW ON); async –to turn asynchronous writes on (FW OFF);

Here are some typical GFIX examples:

```
gfix -w sync -user SYSDBA -pass masterkey firstbase.gdb
```

In this example we set for our test database, firstbase.gdb, the synchronous writes mode (FW ON). (Of course, this is more useful before corruption occurs). And below is the first command that you should use to check the database after corruption has occurred:

```
gfix -v -full -user SYSDBA -pass masterkey firstbase.gdb
```

In this example we start checking our test database (option -v) and specify that fragments of records must be checked as well (option -full). Of course, it is more conven-

ient to set various options for the checking and restoring process using IBExpert or another GUI interface, but we'll review the functions of database recovery using command-line tools. These tools are included in InterBase and Firebird and you can be sure that their behavior will be the same on all OS running InterBase. It is vital that they always be close to the server. Besides the existing tools, allowing you to execute database administration from a client's computer, you can use the Services API, which isn't supported by the InterBase server Classic architecture. That means you need to use a third party product (such as IBExpert or other administration tool) with the SuperServer architecture.

Repairing a corrupt database

Let's assume there are some errors in our database. Firstly, we have to check the existence of these errors; secondly, we have to try to correct these errors. We recommend the following procedure:

You should stop the InterBase server if it's still working and make a copy of the file or the database files. All the restore activities should only be performed with a database copy, because it may lead to an unsatisfactory result, and you'll have to restart the restore procedure (from a starting point). After creating a copy we'll perform the complete database validation (checking fragments of records).

We should execute the following command for this (or use the IBExpert Services menu item Database Validation):

```
gfix -v -full corruptbase.gdb -user SYSDBA -password
```

In this case `corruptbase.gdb` - is a copy of the damaged database. This command will check the database for any structural corruption and produce a list of unsolved problems. If such errors are detected, we'll have to delete the damaged data and get ready for a backup/restore using the following command (or using the IBExpert Services Menu item Backup Database):

```
gfix -mend -user SYSDBA -password your_masterkey corruptbase.gdb
```

After committing this command you should check if there are any errors left in the database. Run `GFIX` using the options -v -full, and when the process is over, perform a database backup:

```
gbak -b -v -ig -user SYSDBA -password corruptbase.gdb corruptbase.gbk
```

This command performs a database backup (option `-b`) and we'll get detailed information about the backup process execution (option `–v`). Errors with regard to checksums will be ignored (option `-ig`).

Please refer to `GBAK` and Backup Database for further information.

If some errors are found during the backup, you should start it in another configuration:

```
gbak -b -v -ig -g -user SYSDBA -password
corruptbase.gdb corruptbase.gbk
```

Where option -g will switch off garbage collection during the backup. This often helps to solve backup problems.

Also it may be possible to make a backup of a database if it is set in the read-only mode beforehand. This mode prevents writing any modifications to the database and sometimes helps to complete the backup of a damaged database. For setting a database to read-only mode, you should use the following command (or the IBExpert Services menu item Database Properties):

```
gfix –m read_only –user SYSDBA –password masterkey
Disk:\Path\file.gdb
```

Following this, you should try to perform the database backup again using the parameters given above (or the IBExpert Service menu item Backup Database).

If the backup was completed successfully, you should restore the database from the backup copy, using the following command (or the IBExpert Services menu item Restore Database):

```
gbak –c –user SYSDBA –password masterkey Disk:\Path\backup.gbk
Disk:\Path\newbase,gdb
```

When you are restoring the database, you may come across some problems, especially when creating the indices.

In this case the `-inactive` and `-one_at_a_time` options should be added to the restore command. These options deactivate indices when creating from the database backup and commit data confirmation for each table. Alternatively use the IBExpert Services menu item Restore Database.

Extract data from a corrupt database

It is unfortunately possible that even the operations previously mentioned in this section do not lead to a successful database recovery.

It means that the database is seriously damaged or it cannot be restored as a single entity, or a huge effort must be made to recover it. For example, it is possible to execute a modification of system metadata, use non-documented functions and so on. It is very hard, time-consuming and ungrateful work with doubtful chances of success. If at all possible, try to evade it and use other methods. If a damaged database opens and allows you to perform reading and modification operations with some data, you should take advantage of this possibility and save the data by copying it to a new database, and say good-bye to the old one for good.

So, before transferring the data from the old database, it’s necessary to create a new destination database. If the database hasn’t been altered for a long time, you can use the old backup, from which metadata can be extracted for creating the new database. Based on these metadata it is necessary to create a data destination and start copying the data. The main task is to extract the data from the damaged database. Then we’ll have to allocate the data in a new base, but that’s not very difficult, even if we have to restore the database structure from memory.

When extracting data from tables, you should use the following algorithm of operations:

- At first you should try to execute `SELECT * from table N`. If it ran normally you could save the data you've got in the external source. It's better to store data in a script (using IBExpert Tools menu item Extract Metadata for example), as long as the table doesn't contain blob fields. If there are blob fields in the table, then this data should be saved to another database by a client program that will play the role of mediator.
- If you failed to retrieve all data, you should delete all the indices and try again. In fact, indices can be deleted from all the tables from the beginning of the restore, because they won't be needed any more. Of course, if you don't have a metadata structure which is the same as that of the corrupted database, it's necessary to input a protocol of all operations that you are doing with the damaged database source.
- If you cannot read all the data from the table after deleting the indices, try to execute a range query by primary key, i.e. select a definite range of data. For example:
  `SELECT * FROM table N WHERE field_PK >=0 and field_PK <=10000`
  `Field_PK` here is a primary key.

InterBase has page data organization and that's why a range query of values may be rather effective.

Nevertheless it works because we can expel data from the query from damaged pages and fortunately read the other ones. You may recall our thesis that there is no defined order of storing records in SQL.

Really, nobody can guarantee that an unordered query will, during restarts, return the records in the same order, but nevertheless the physical records are stored within the database in a defined internal order. It's obvious that the server will not mix the records purely to abide to SQL standards. Try to use this internal order when extracting data from a damaged database. Vitaliy Barmin, an experienced Russian InterBase developer reported that in this way he managed to restore up to 98% of information from an unrecoverable database (there were a great number of damaged pages). Thus, data from a damaged database must be moved to a new database or into external sources such SQL scripts. When you copy the data, pay attention to generator values in the damaged database (they must be saved for restarting proper work in the new database. If you don't have a complete copy of the metadata, you should extract the texts of stored procedures, triggers, constraints and the definition of indices.

Restoring hopeless databases

In general, restoring a database can be very troublesome and difficult and that's why it's better to make a backup copy of the database and then restore the damaged data and whatever has happened, you shouldn't despair because a solution can be found even in the most difficult situations. And now we'll consider two cases.

The first case (a classic problem): A backup that can't be restored because of having NULL values in a column with NOT NULL constraints (the restore process was run over the working file). The working file was erased and the restore process was interrupted because of an error. And as a result of thoughtless actions the result was a great

amount of useless data (that can't be restored) instead of a backup copy. But a solution was found. The programmer managed to recollect which table and which column contained the constraint NOT NULL. The backup file was loaded to a hexadecimal editor. And a combination of bytes, corresponding to the definition of this column, was found by searching. After innumerous experiments it turned out that the constraint NOT NULL adds 1 somewhere near the column name. In the HEX-editor this 1 was corrected to 0 and the backup copy was restored. Following this, the programmer memorized once and for all how to execute the backup process and restore successfully!

The second case: The situation was catastrophic. The database corrupted on the extension phase because of lack of disk space.

When increasing the database size, the server creates a series of critically important pages (for example, Transaction Inventory Page and Page Inventory Page, additional pages for RDB$Pages relations) and writes them down at the end of database.

As a result, the database could not be opened, neither by administration facilities nor using the utility GBAK. And when we tried to connect to the database, an error message (`Unexpected end of file`) appeared. When we ran the utility GFIX strange things happened: The program was working in an endless cycle. When GFIX was working, the server was writing errors to log (file InterBase log) at high speed (around 100 Kb per second). As a result, the log file filled all the free disk space very quickly. We even had to write a program that erased this log by timer. This process lasted for a long time - GFIX was working for more than 16 hours without any results.

The log was full of the following errors: `Page XXX doubly allocated`. When starting InterBase sources (in file `val.c`) there is a short description of this error. It says that this error appears when the same data page is used twice.

It's obvious that this error is a result of corruption of critically important pages.

As a result, after several days of unsuccessful experiments, all attempts to restore the data in the standard way were abandoned. Which is why we had to use a low-level analysis of the data stored in the damaged database.

Alexander Kozelskiy, head of Information Technologies at East View Publications Inc, had the idea of how to extract information from similar unrecoverable databases.

The method of restoring, arrived at as a result of our research, was based on the fact that a database has page organization and data from every table is collected by data pages. Each data page contains an identifier of the table for which it stores data. It was especially important to restore data from several critical tables. There was data from similar tables, received from an old backup copy that worked perfectly and could be used as a model. This database sample was loaded into an editor of hexadecimal sources and then we searched for the patterns of the data that interested us. This data was copied into a buffer in hexadecimal format and then the remains of the damaged database were loaded into the editor. A sequence of bytes corresponding to the sample was found in the damaged database, and the page was analyzed (on which this sequence was found).
At first we needed to define the start page, which wasn't difficult because the size of the database file is divisible by the data page size. The number of current bytes divided by page size - 8192 bytes, approximates the result to integer (and we obtained the

number of the current page). Then the number of current page was multiplied by page size and we got the number of bytes corresponding to the beginning of the current page. Having analyzed the header, we defined the type of page (for pages with data the type is 5 - please refer to the file `ods.h` from the set of InterBase sources as well as the identifier of the necessary table.

Then a program was written, that analyzed the whole database, collected all the pages for the necessary table into one single piece and moved it to file.

Thus, once we had the data we initially needed, we began analyzing the contents of the selected pages. InterBase uses data compression widely in order to save space. For example, a string such as VARCHAR containing an ABC string, stores a sequence of following values: string length (2 bytes), in our case it is 0003, and then the symbols themselves followed by a checksum. We had to write an analyzer of the string as well as other database types that converted data from hexadecimal format into an ordinary view. We managed to extract up to 80% of the information from several critical tables using a 'manual' method of analyzing the database contents. Later, on the basis of this experience, Oleg Kulkov and Alexey Kovyazin, one of the authors of this book, developed the utility InterBase Surgeon which performs direct access to the database, bypassing the InterBase engine and enables you to read directly and interpret the data within the InterBase database in a proper way.

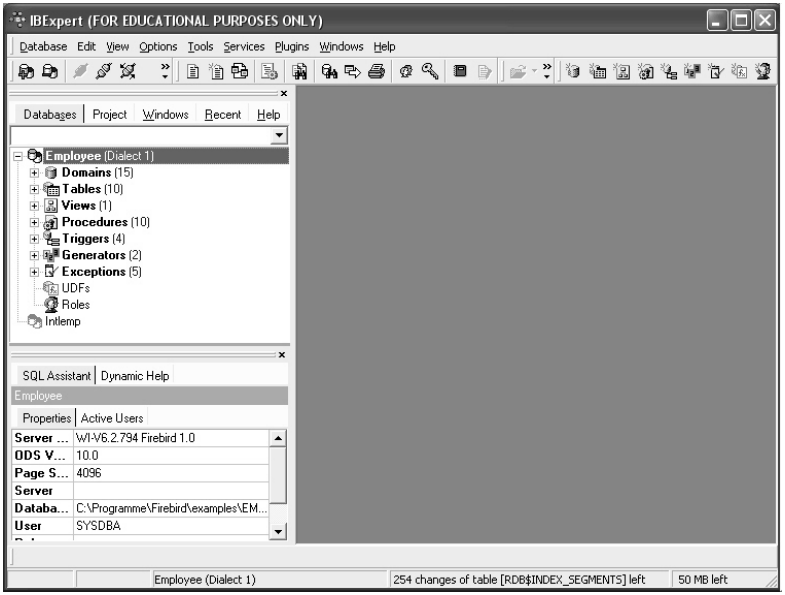
Using InterBase Surgeon, we have managed to detect the causes of corruption and restore up to 90% of absolutely unrecoverable databases, which can't be opened by InterBase and restored by standard methods. This program can be downloaded from the official site www.ib-aid.com.

3 Database Objects

InterBase/Firebird administrates the database data in database objects. These are the fundamental building blocks of the database and include the following:

- Domains
- Tables
- Generators
- Constraints
- Indices
- Views
- Triggers
- Stored Procedures
- Exceptions
- Blob Filters
- User-defined functions (UDFs)

The database objects can be viewed, created, edited and deleted using the IBExpert DB Explorer.



Alterations to database objects (online operation) are limited to 255 alterations per object (see Status bar for more details). At this stage a backup and restore is necessary, in order to perform further alterations. This limitation is due to the fact that InterBase stores each data structure every time a record is inserted.

The IBExpert object editors all contain detailed dialogs for inserting, altering and dropping individual objects. The majority of editors display a number of tabs, comprising multiple input and display pages. Certain typical windows recur in several object editors:

- **Dependencies**: all objects, which depend on other objects or where other objects are depending on this object, can be viewed on the object editor's Dependencies page.
- **DDL**: the SQL code, resulting from the user input, is displayed.
- **Performance Analysis**: for stored procedures and the SQL Editor, the result set can be started with [F9]. The performance result is displayed on a new page.
- **Description**: shows the description field from the InterBase/Firebird database.
- **Grants**: this page allows user rights to be granted for the active object directly in the object editor dialog, without having to leave and start the Grant Manager each time a new object is created. It is even possible to switch to other objects (i.e. views, triggers, procedures and roles), without having to leave the editor.

These pages are explained in more detail in the Table Editor (except Performance Analysis - details under SQL Editor / Performance Analysis).

3.1 Domain

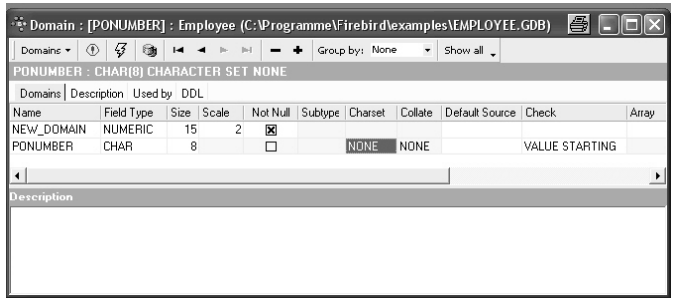
A domain is a user-defined data type global to the database. It is used for defining the format and range of columns, upon which actual column definitions in tables may be based.

This is useful if columns in one or several database tables have the same properties, as it is much simpler to describe such a column type and its behavior as a domain. The columns can then simply be defined by specifying the domain name in the column definition. The column properties (e.g. field length, type, Not Null, constraints, arrays etc.) only need to be defined once in the domain.

Certain attributes specified in the domain can be overwritten in the table field definition, i.e. a column can be based upon a domain; however small changes may still possibly be made for this column.

In addition to the data type, a number of conditions and checks can be defined.

A domain is a database object and is part of the database's metadata, and can be created, modified and dropped as all other InterBase/Firebird objects in the IBExpert DB Explorer.



When developing a normalized database, the question arises in how far domains are necessary (multiple fields, multiple data etc.).

However, it does make life easier, should column alterations be necessary; e.g. zip code alteration from 4 to 5 digits (as was the case in Germany after the reunion), change of currency (e.g. from DM or Lire to Euro). In such cases, only the domain needs to be altered, and not each relevant column in each table individually throughout the database.

It should also be noted, that if user-defined domains are not explicitly defined and used for table column definitions, InterBase/Firebird generates a new domain for every single table column created!

3.1.1 Domain Integrity

Domain integrity ensures that a column is kept within its allowable limits. This is achieved by keys and constraints.

3.1.2 New Domain / Domain Editor

A new domain can be created for a connected database, either by using the menu item Database / New Domain, or using the DB Explorer right-click menu (or key combination [Ctrl + N]), when the domain heading of the relevant connected database is highlighted), or the New Domain icon on the New Database Object toolbar.

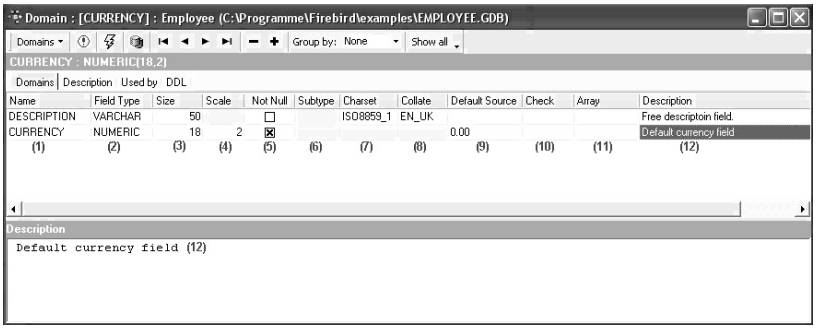
A New Domain dialog appears, with its own toolbar, and a pull-down menu (domain button). The toolbar offers the following options:

- Enable direct modifying of system tables
- Compile
- Duplicate the selected domain
- Navigational buttons
- Group by either Type or Charset
- Display all domains

For those users preferring to use the old IBExpert Modal Editor, check the *Use old-style Modal Editor* option in the IBExpert Options menu: Objects Editor Options / Domains Editor.

A domain can also be created or selected and edited, when creating a new field is created, or an existing field edited in a table, using the Table Editor. (Please refer to Insert Field for further information).

Initially a domain name is specified (1) in the first column on the first page "Domains":



(Illustration displays the new default Domain Editor.)

(2) Field Type: Here the data type can be specified.

(3) Size: Specifies the field size.

(4) Scale: Here the number of decimal places can be specified for all numerical fields.

(5) Not Null: This check box can be marked by double-clicking or using the space bar. Not Null forces data to be entered in this field (i.e. the field may not be left empty).

(6) Subtype: A subtype should be specified for blob fields.

(7) Charset: A character set may be specified for individual domains. This overrides the database default character set. Although this is seldom used, it may be necessary should, for example, Asian, Russian or Arabic addresses need to be input and collated in a database with a European default character set.

(8) Collate: Determines collation for a character set specified for a domain.

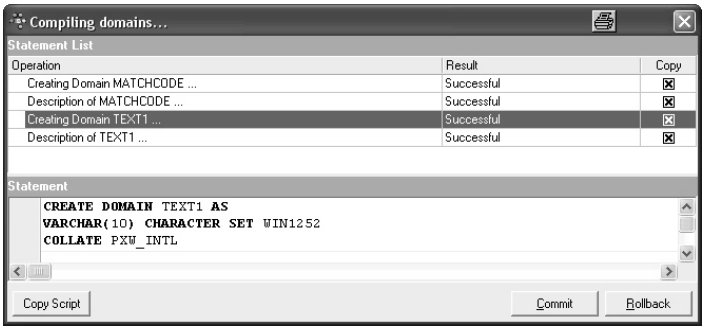
(9) Default Source: Here a default data entry (text or numeric, depending upon the specified data type) can be specified, e.g. the text NOT KNOWN can be specified as a default source, if an address field cannot be input by the user, because the information is unavailable.

(10) Check: Each data set is examined for validity according to an expression defined in brackets. Certain conditions can be specified (see Check Constraint) causing an automatic database examination during data input, to ensure data consistency in the tables and among each other.

(11) Array: Although arrays contradict all the rules of database normalization, there are certain situations (for example storing measurement data), when they are necessary.

(12) Description: Useful for database documentation. The Description page should be used to describe the domain; the Description field for describing the field.

Several domains can be created simultaneously in the New Domain Editor. After creating the new domain(s), including all necessary parameters, don't forget to compile (using [Ctrl + F9] or the respective icon):



and finally committing, or should amendments be necessary, rolling back.

Tip: by clicking on the column headers (i.e. PK, FK, Field Name etc.), the fields can be sorted into ascending or descending order based upon that column.

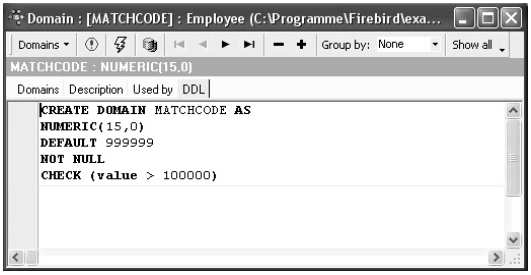
Double-clicking on the right edge of the column header adjusts the column width to the ideal width.

In addition to the Domains page, there are also Description, Used By and DDL pages:

Description: this displays the description for the highlighted domain (i.e. the domain, where the cursor is currently standing).

Used By: this displays those database objects which use or depend upon this domain.

DDL: the DDL page displays the SQL statement created by IBExpert to create all specifications made by the user on the Domains page.



Domains can also be created and edited directly from the New Field Editor (please refer to Insert Field).

Domains can, of course, also be created using DDL directly in the SQL Editor, using the following syntax:

```
CREATE DOMAIN domain_name [AS] <data_type>
[DEFAULT {expression | NULL | USER}]
[NOT NULL] [CHECK (<domain_such_expression>)]
[COLLATE collation];
```

For example:


```
CREATE DOMAIN MATCHCODE
   AS INTEGER
      DEFAULT  999999
      NOT NULL
      CHECK (VALUE > 100000);
```

3.1.3 Alter Domain

A domain can be altered in the Domain Editor, opened by double-clicking on the domain name in the DB Explorer. Alternatively use the DB Explorer's right mouse-click menu item Edit Domain or key combination [Ctrl + O].

CHECK instructions and default values may be added, altered or deleted. However it is not possible to alter the basic data type (for example, from Numeric to Varchar). Neither is it possible to drop a NOT NULL constraint. To alter these the domain has to be dropped and recreated (see Drop Domain).

Please note that if you want to change the CHECK constraint for a domain that already has a constraint defined, the existing constraint must first be dropped and then the new one added. ADD CHECK does not replace the current constraint with the new one. It is also important to realize that altering a CHECK constraint does not cause existing database rows to be revalidated; CHECK constraints are only validated when an INSERT or UPDATE is performed. One way of overcoming this limitation is to perform an UPDATE query using a dummy operation. If existing rows violate the new CHECK constraint, the query fails. These rows can then be extracted by performing a SELECT.

Any changes made apply immediately to all columns using the domain definition, unless, of course, the column's (field) definition overrides these.

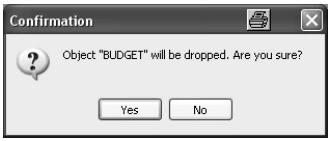
The SQL syntax for this command is:

```
ALTER DOMAIN <domain_name>
SET DEFAULT <default_value> | NULL | USER
   DROP DEFAULT
ADD CHECK <domain_search_condition>
   DROP CONSTRAINT;
```

3.1.4 Drop Domain/Delete Domain

A domain may only be dropped if it is not currently being used by any of the database tables. The Domain Editor's Used By page shows which database objects use this domain. The dependent objects may also be directly dropped here, if wished, using the right-click menu on the selected object, and choosing the menu item Drop Object or [Ctrl + Del].

To drop a domain use the DB Explorer right-click and select the menu item Drop Domain or [Ctrl + Del]. Alternatively, a domain can be dropped directly from the Domain Editor using the pull-down menu Domains or the '-' icon in the Domain Editor toolbar. IBExpert asks for confirmation:



before finally dropping the domain. Once dropped it cannot be retrieved; the domain has to be recreated if a mistake has been made!

Using SQL the syntax is:

```
DROP DOMAIN <domain_name>;
```

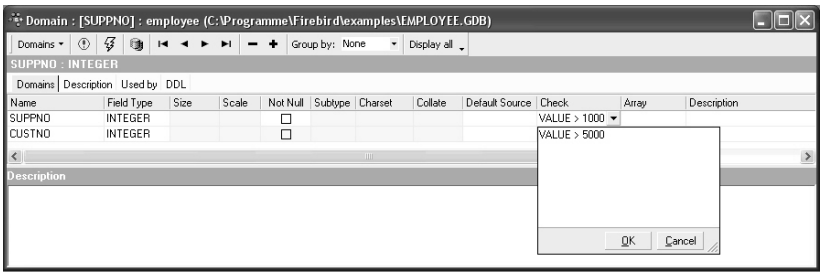
A domain can only be dropped by its creator or the SYSDBA.

3.1.5 Duplicate Domain

It is possible to create a new domain, based on an existing domain, using the Domain Editor's menu item Duplicate Domain, or the

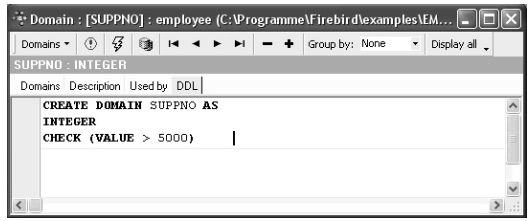
icon in the Domain Editor toolbar.

An exact copy of the selected domain is made, and can then be adapted as wished. For example a new domain, SUPPNO could be based on the CUSTNO domain in the EMPLOYEE database, by duplicating it and then, for example, renaming it and altering the CHECK VALUE to > 5000.



This saves time creating several similar domains; all you need to do is copy a domain, perform any minor alterations necessary, compile and finally commit.

The Domain Editor's DDL page displays the actual statement used to create the new domain:



Duplicating domains from one database to another

If you have already created a wide range of domains in one database, and would like to duplicate them in another new database, simply take the following steps in IBExpert:

- Copy the domain DDL (Data Definition Language) into the SQL Editor and execute it.
- Drag 'n' drop the domain from the source database into the Domain Editor of the target database.

3.2 Table

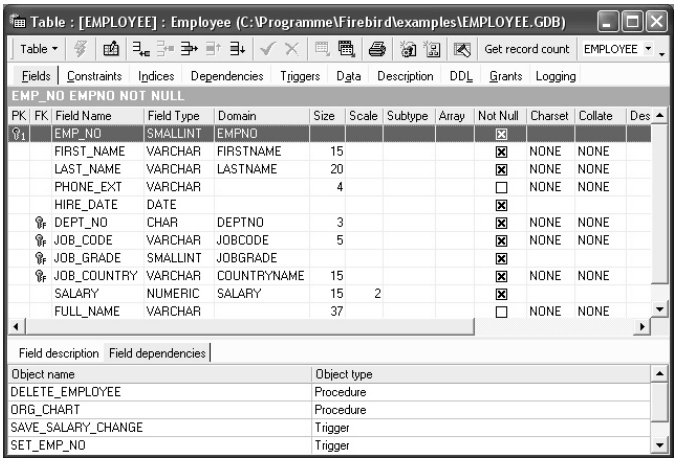
A table is a data storage object consisting of a two-dimensional matrix or grid of columns and rows, theoretically known as a mathematical relation. It is a fundamental element for data storage.

Relational databases store all their data in tables. A table consists of an unordered set of horizontal rows (tuples). Each of these rows contains the same number of vertical columns for the individual singular information types.

The intersection of an individual row and column is a field containing a specific, indivisible atomic piece of information. I.e. columns list the names of individual fields and rows are the data sets containing the input data. Each database column may be assigned a different data type.

A table is a database object that is part of the database's metadata.

Tables of connected databases can be viewed and manipulated in the IBExpert DB Explorer



We recommend restricting a table name to no more than 14 characters, so that foreign key names (which are limited to 32 characters up until InterBase 6 and Firebird 1.5; InterBase 7 allows 64 characters) can include both related table names in its name:

Prefix "FK" plus two separators plus both table names, e.g.

```
FK_Table1_Table2
```

Please note however that this is not an InterBase/Firebird restriction, but purely an IBExpert recommendation to enable a clear and logical naming convention for foreign keys.

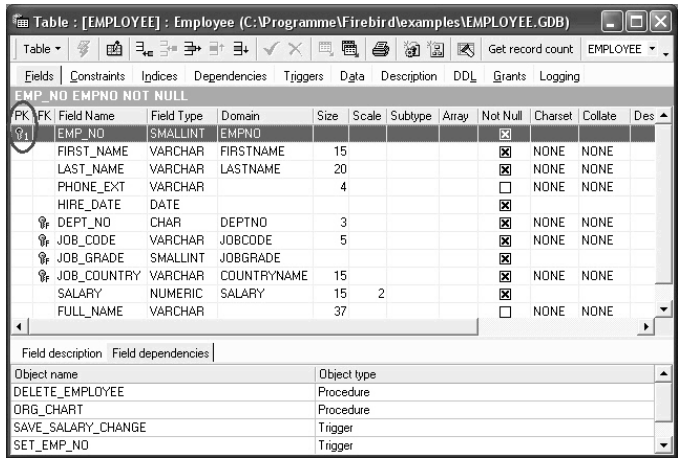
3.2.1 Keys

A key is used to organize data logically, so that a specific row can be uniquely identified. A key should not be confused with an index. In the relational model, a key is used to organize data logically, so that unique identification of a specific row is possible. An index is part of the table's physical structure on-disk. It is used to speed data access when queries are performed. Indices are therefore not a part of the relational model.

InterBase/Firebird automatically generates an index for primary and foreign key columns. On primary key columns, the index actually enforces the unique constraint required by the relational model. Links between tables usually occur on primary and foreign keys, so having an index on these columns ensures maximum performance.

Primary Key

A primary key is a column (= simple key) or group of columns (= composite key) used to uniquely define a data set/row in the table. A primary key should always be defined at the time of defining a new table for each table. If you have a database that does not contain primary keys in all tables, and need to add these subsequently, please refer to Adding Primary Keys to Existing Tables.

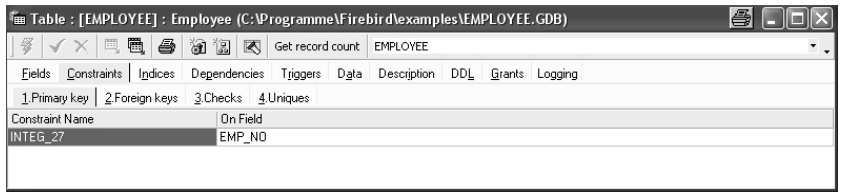


Relational theory states that a primary key should be designated for every table. It must be unique, and therefore cannot be Null. It provides automatic protection against storing multiple values. Each table can have only one designated primary key, although it can have other columns that are defined as unique and Not Null.

A primary key column is nothing other that a unique constraint complemented by a system index and the check constraint Not Null. Primary keys are always the preferred index of the InterBase/Firebird Optimizer.

When a data set is created or changed, Firebird/InterBase immediately checks the validity of the primary key. If the number already exists, a key violation results, and the storage process is immediately cancelled.

Unfortunately InterBase/Firebird allows tables to be created without a primary key, which is a mistake. Data tables should always be keyed.



Existing primary keys and their system names can be viewed on the IBExpert Table Editor / Constraints page.

It is wise to keep the primary key as short as possible to minimize the amount of disk space required, and to improve performance. IBExpert recommends the use of an autoincrement generator ID number, used as an internal primary key for all tables. Composite keys are not recommended, as these always slow performance and the sequence of the fields concerned must be identical in all referenced tables.

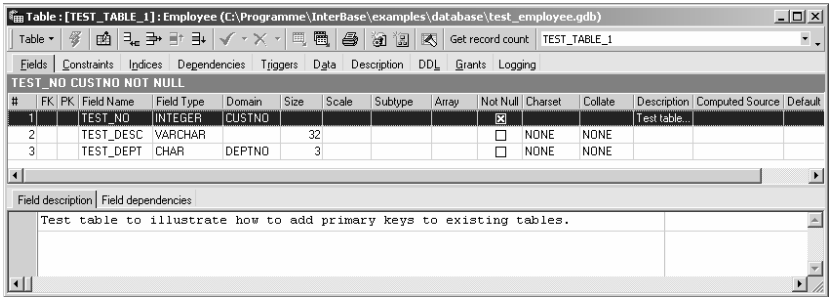
Adding primary keys to existing tables

This article was written by Melvin Cox, and provides a method of defining primary keys on existing tables using IBExpert:

Here is a viable workaround for those of us who do not wish to spend an eternity exporting data, dropping and recreating multiple tables, and finally import the data back into those tables.

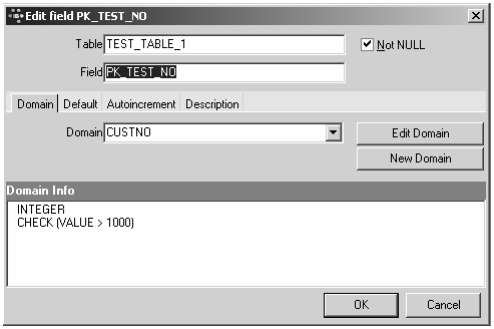
Working with a Firebird 1.5 database (dialect 1) created via ODBC export from a Microsoft Access database, I have successfully defined primary keys on tables by taking the following steps:

1) Bring up the table within the IBExpert interface's Table Editor window (double-click on the respective table in the DB Explorer or use [Ctrl. + O]). The Fields tab should be active.



2) Double click in the Not Null box corresponding to the field that you wish to designate as the primary key. This will call up the Edit Field dialog.

3) Within the Edit Field dialog, check Not Null and press OK. This will call up a Edit Field dialog. Check the Not Null option and select an existing or create a new domain.



4) Press OK and then, after checking the script produced by IBExpert, the Commit button. The field is now set to Not Null.

5) Bring up the SQL Editor. Tools -> SQL Editor (or press F12).

6) Enter the following command:

```
ALTER TABLE table_name ADD PRIMARY KEY (field_name);
```

For example, to define a primary key on the events table enter:

```
ALTER TABLE events ADD PRIMARY KEY (event_id);
```

7) Press the Execute Button or [F9].

8) Close the SQL Editor. This will call up the Active Transaction Found dialog. Select Commit.

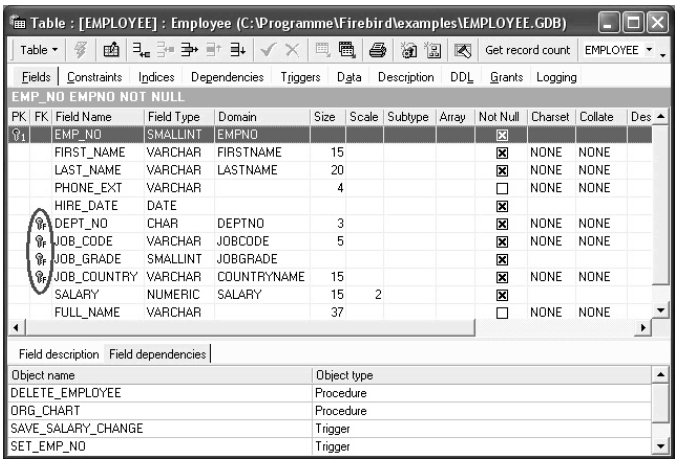
9) Close the Table Editor window.

10) Reopen the Table Editor window [Ctrl. + O]. The newly defined primary key will now be visible.

Foreign Key

A foreign key is composed of one or more columns that reference a primary key. Referencing meaning here that when a value is entered in a foreign key, Firebird/InterBase checks that the value also exists in the referenced primary key. This is used to maintain domain integrity.

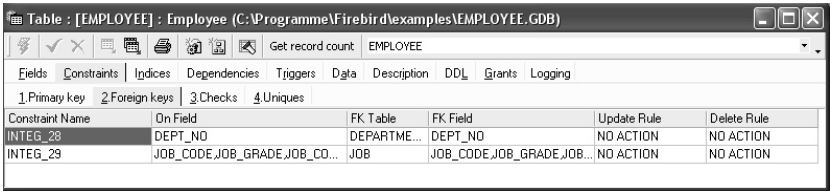
A foreign key relationship is defined in the IBExpert Table Editor (started from the DB Explorer) on the Constraints page. Please refer to Table Editor / Constraints for further information.



Foreign keys are used mainly for so-called reference tables. In a table storing, for example, employees, it needs to be determined, which department each employee belongs to. Possible entries for the department number of each employee data set are contained in the department table. As the employee table refers to the department number as the primary key for the department table, there is a foreign key relationship between the employee table and the department table. Foreign key relationships are automatically checked in Firebird/InterBase, and data sets with a non-existent department number cannot be saved.

When a primary key:foreign key relationship links to a single row in another table, what is known as a virtual row is created. The columns in that second table provide additional description about the primary key of the first table.

Foreign keys and their system names can be defined and viewed on the IBExpert Table Editor / Constraints page.



A primary key does not have to reference a foreign key. However a unique index is insufficient; a unique constraint needs to be defined (this definition also causes a unique index to be automatically generated). SQL syntax:

```
ALTER TABLE MASTER
ADD CONSTRAINT UNQ_MASTER UNIQUE (FIELD_FOR_FK);
```

Foreign key names are limited to 32 characters up until InterBase 6 and Firebird 1.5; InterBase 7 allows 64 characters. IBExpert therefore recommends limiting table names to 14 characters, so that the foreign key name can include both related table names:

Prefix "FK" plus two separators plus both table names, e.g.

```
FK_Table1_Table2
```

Please note however that this is not an InterBase/Firebird restriction, but purely an IBExpert recommendation to enable a clear and logical naming convention for foreign keys.

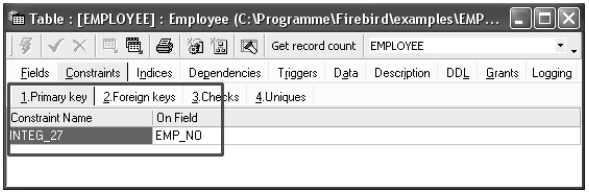
Note: if data has already been input in a table which is to subsequently be assigned a foreign key, this will not be allowed by InterBase/Firebird, as it violates the principle of referential integrity. It is however possible to filter and delete the old data (where no reference to a primary key has been made) using a `SELECT` statement and committing. It is important to then disconnect and reconnect the database in IBExpert, for this to work.

Candidate Key

Any column or group of columns which can uniquely identify a data set, and can therefore be considered for use as a primary key. Is always Not Null (i.e. must not be left undefined), and unique.

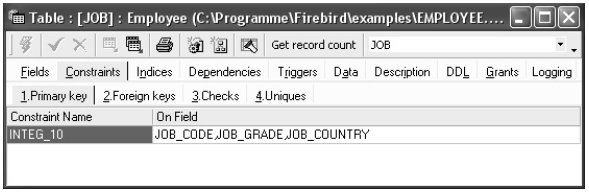
Simple Key

A simple key is composed of one column only, i.e. a single column is designated as a table's primary key.



Composite Key

A composite key consists of two or more columns, designated together as a table's primary key.



Unfortunately such keys have two huge disadvantages: firstly they slow the database performance considerably, as InterBase/Firebird needs to check all contents of all columns designated in such a composite key; secondly the sequence of the fields concerned must be identical in all referenced tables.

Basically composite keys should be avoided! It is much preferable to use an internal ID key (so-called artificial key) as the primary key for each table.

Unique

Unique fields are unequivocal, unambiguous, one-of-a-kind (i.e. there is no duplicate information allowed in the data sets of a unique field). Such fields must therefore also be Not Null.

Unique fields are given a unique index. Each unique field is a candidate key.

Artificial Key/Surrogate Key/Alias Key

An artificial or alias or surrogate key is created by the database designer/developer if there is no candidate key, i.e. no logical, simple field to be the primary key. An artificial key is a short ID number used to uniquely identify a record.

Such an internal primary key ID is recommended for all tables. They should always be invisible to the user, to prevent any potential external influence regarding their appearance and composition.

It is always wise to keep the primary key as short as possible to minimize the amount of disk space required, and to improve performance; therefore artificial keys should also be as short as possible. An ideal solution for the generation of an artificial key is the use of an autoincrement generator ID number. IBExpert recommends this solution be used as an internal primary key for all tables.

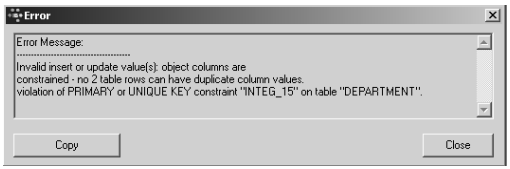
Usually such an artificial/alias/surrogate key is just an autoincrement integer field so that each record has it's own unique integer identifier. For example:

```
CREATE TABLE CUSTOMERS (
   CUSTOMER_ID INTEGER NOT NULL,
   FIRST_NAME VARCHAR(20),
   MIDDLE.NAME VARCHAR(20),
   LAST_NAME VARCHAR(20);
...);
```

In this case `CUSTOMER_ID` the artificial or surrogate key.

Key violation

When a data set is created or changed, InterBase/Firebird immediately checks the validity of the primary key. If the number already exists, or the field has been left blank, a key violation results, and the storage process is immediately cancelled.



InterBase/Firebird immediately sends an error message referring to the violation of a unique or primary key constraint.

Referential Integrity

The relationship between a foreign key and its referenced primary key is the mechanism for maintaining data consistency and integrity. Referential integrity ensures data integrity between tables connected by foreign keys. A foreign key is one or more columns that reference a primary key, i.e. when a value is entered in the foreign key, In-

terBase/Firebird checks that this value also exists in the referenced primary key, so maintaining referential integrity.

Referential integrity can occur in the following three cases:

1. In the master table a Data Set is deleted. For example, the deletion of a customer, for whom there are still existing orders could lead to order data sets without a valid customer number. This could falsify analyses and lists, as the internal relationships no longer appear. The prevention of data set deletion in the master table, when data sets still exist in the detail table, is called prohibited deletion. The relay of deletions to all detail tables is called cascading deletion.

2. The primary key is changed in the master table. For example a customer is given a new customer number, so that all orders relating to this customer need to also relate to the new customer number. This is known as a cascading update.

3. A new data set is created, and the foreign key does not exist in the master table. For example an order is input with a customer number, not yet allocated in the master table. A possible solution could be the automatic generation of a new customer. This is called a cascading insert.

Referential integrity is supported natively in InterBase/Firebird, i.e. all foreign key basic relationships are automatically taken into consideration during data alterations. Since Version 5, InterBase supports declarative referential integrity with cascading deletes and updates. In older versions, this could be implemented with triggers.

Cascading Referential Integrity

Since InterBase v5/Firebird, cascading referential integrity is also supported.

When a foreign key relationship is specified, the user can define which action should be taken following changes to, or deletion of its referenced primary key. `ON UPDATE` defines what happens when the primary key changes and `ON DELETE` specifies the action to be taken when the referenced primary key is deleted. In both cases the following options are available:

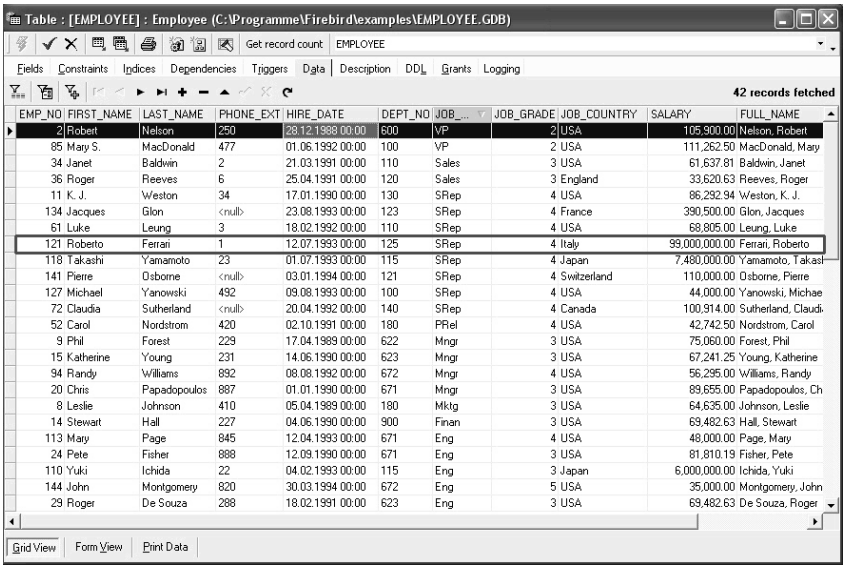
1. `NO ACTION`
2. `NULL` - the foreign key column should be set to its default value
3. `CASCADE` - the foreign key column is set to the new primary key value. The `CASCADE` option also deletes the foreign key row when the primary key is deleted.

3.2.2 Data

Data is the quantity of facts or information input, processed and stored in a computer. Data can consist of one single entry in one field, a data set comprises a series of fields or in fact, any data quantity.

3.2.3 Data Set

A data set is one complete data record, which is none other than a table Row (which can be viewed on the IBExpert Table Editor / Data page). It encompasses a single set of information, such as, for example, one customer address or one employee record.



EMP_NO	FIRST_NAME	LAST_NAME	PHONE_EXT	HIRE_DATE	DEPT_NO	JOB_...	JOB_GRADE	JOB_COUNTRY	SALARY	FULL_NAME
2	Robert	Nelson	250	28.12.1988 00:00	600	VP	2	USA	105,900.00	Nelson, Robert
85	Mary S.	MacDonald	477	01.06.1992 00:00	100	VP	2	USA	111,262.50	MacDonald, Mary
34	Janet	Baldwin	2	21.03.1991 00:00	110	Sales	3	USA	61,637.81	Baldwin, Janet
36	Roger	Reeves	6	25.04.1991 00:00	120	Sales	3	England	33,620.63	Reeves, Roger
11	K. J.	Weston	34	17.01.1990 00:00	130	SRep	4	USA	86,292.94	Weston, K. J.
134	Jacques	Glon	<null>	23.08.1993 00:00	123	SRep	4	France	390,500.00	Glon, Jacques
61	Luke	Leung	3	18.02.1992 00:00	110	SRep	4	USA	68,805.00	Leung, Luke
121	Roberto	Ferrari	1	12.07.1993 00:00	125	SRep	4	Italy	99,000,000.00	Ferrari, Roberto
118	Takashi	Yamamoto	23	01.07.1993 00:00	115	SRep	4	Japan	7,480,000.00	Yamamoto, Takas
141	Pierre	Osborne	<null>	03.01.1994 00:00	121	SRep	4	Switzerland	110,000.00	Osborne, Pierre
127	Michael	Yanowski	492	09.08.1993 00:00	100	SRep	4	USA	44,000.00	Yanowski, Michae
72	Claudia	Sutherland	<null>	20.04.1992 00:00	140	SRep	4	Canada	100,914.00	Sutherland, Claudi
52	Carol	Nordstrom	420	02.10.1991 00:00	180	PRel	4	USA	42,742.50	Nordstrom, Carol
9	Phil	Forest	229	17.04.1989 00:00	622	Mngr	3	USA	75,060.00	Forest, Phil
15	Katherine	Young	231	14.06.1990 00:00	623	Mngr	3	USA	67,241.25	Young, Katherine
94	Randy	Williams	892	08.08.1992 00:00	672	Mngr	4	USA	56,295.00	Williams, Randy
20	Chris	Papadopoulos	887	01.01.1990 00:00	671	Mngr	3	USA	89,655.00	Papadopoulos, Ch
8	Leslie	Johnson	410	05.04.1989 00:00	180	Mktg	3	USA	64,635.00	Johnson, Leslie
14	Stewart	Hall	227	04.06.1990 00:00	900	Finan	3	USA	69,482.63	Hall, Stewart
113	Mary	Page	845	12.04.1993 00:00	671	Eng	4	USA	48,000.00	Page, Mary
24	Pete	Fisher	888	12.09.1990 00:00	671	Eng	3	USA	81,810.19	Fisher, Pete
110	Yuki	Ichida	22	04.02.1993 00:00	115	Eng	3	Japan	6,000,000.00	Ichida, Yuki
144	John	Montgomery	820	30.03.1994 00:00	672	Eng	5	USA	35,000.00	Montgomery, John
29	Roger	De Souza	288	18.02.1991 00:00	623	Eng	3	USA	69,482.63	De Souza, Roger

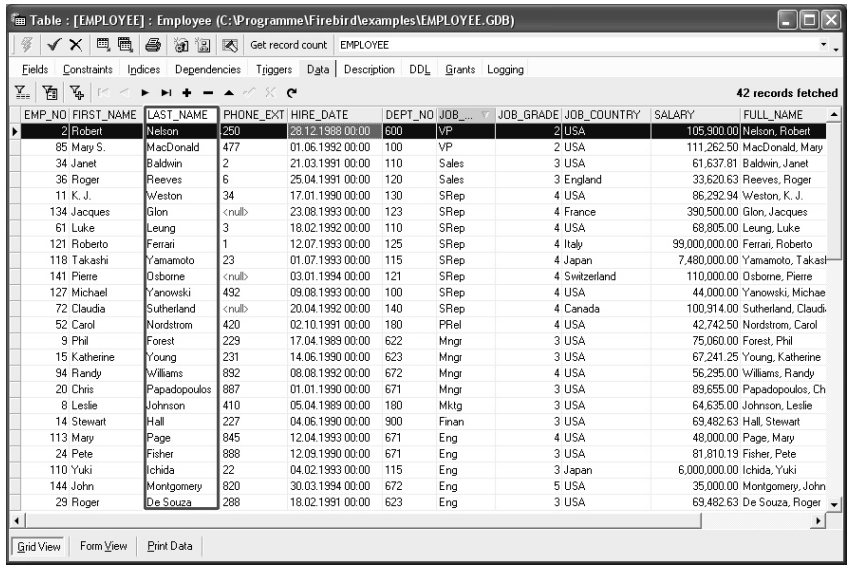
In a relational database the physical sequence of data sets is irrelevant.

Duplicate data sets or records (i.e. double rows) are not allowed in a relational database, as this is, in effect, storage of redundant information (see Database Normalization).

3.2.4 Column

A column is part of a database table, and is also known as an attribute or field. Columns list the names of the individual fields in a table.

A column describes an atomic or indivisible basic piece of information in the database, clearly differentiated from other data, e.g. zip code (and not zip code + city). Each column is assigned a certain data type, e.g. text, numeric, date or blob. The data can also be assigned properties, such as unique, contain check constraints, autoincrements, computed values, restricted to minimum and maximum values etc. etc.



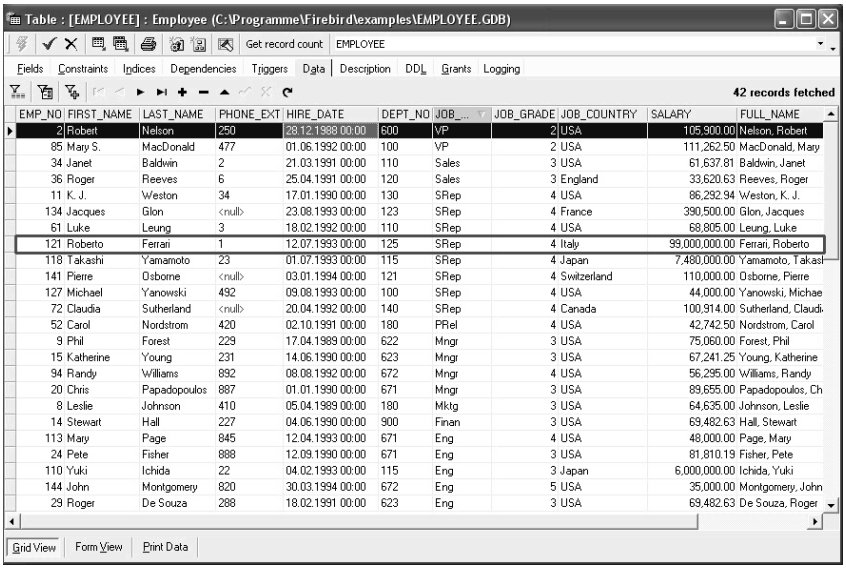
Columns are defined under the Field Definition in the Create Table Dialog or Table Editor, or their definition can be based on domains. They can, of course, also be defined directly in the SQL Editor. Each defined column has the following syntax:

```
ColumnName <data_type>
DEFAULT < Default value > | NULL | USER NOT NULL
CONSTRAINT <constraint name> <constraint def>
COLLATE <collation sequence>;
```

In a relational database the physical sequence of rows and columns is irrelevant.

3.2.5 Row

A row is also called a tuple, record or data set. Each row represents an instance of data, belonging together, composed of different columns. It encompasses a single set of information, such as, for example, one customer address or one employee record.



In a relational database the physical sequence of rows and columns is irrelevant.

Double rows (i.e. duplicate data sets or records) are not allowed in a relational table, as this is, in effect, storage of redundant information (see Database Normalization).

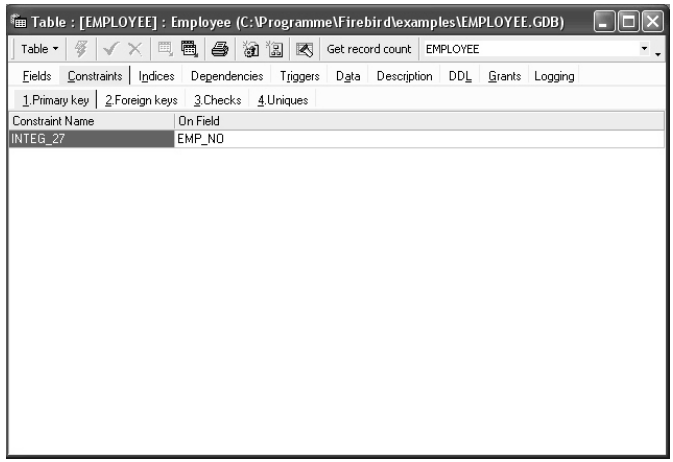
3.2.6 Constraints

A constraint is a database examination, which ensures data consistency in the tables and among each other. The constraint determines the range of acceptable values for a column (or columns) or data set in a database or application. This constraint can be executed automatically and so ensures that data contents are kept consistent by testing them as they are input.

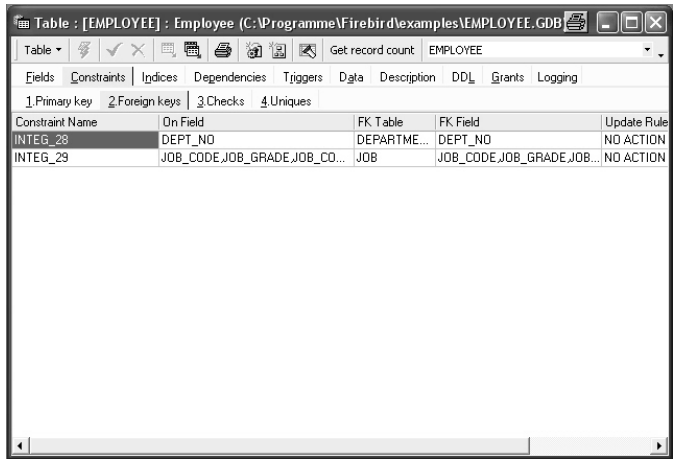
A constraint can be specified for each column (or columns) in a table, to guarantee the mechanism described above. Constraints can be domain- or column-based and the specified conditions must be met when new data sets are inserted, or existing data sets are modified. They are used to verify data integrity. If a condition is not met, an exception is raised.

InterBase/Firebird internally generates a trigger for each check condition. Constraints can be defined as follows:

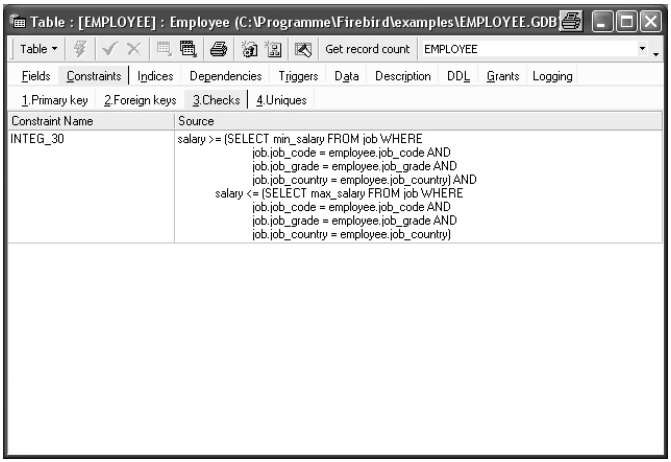
1. Primary Key/Unique - Specification of the unique option forces a unique entry in this column (these columns) for each data set (i.e. duplicate field entries are not allowed).



2. Foreign Key - The foreign key option determines that the column(s) is/are linked by a referential integrity relationship to the primary key of another table (i.e. the input data is only accepted if it already exists in the primary key column(s) in the referenced table).



3. CHECK - the check option enables each data set to be examined for validation of an expression specified in brackets. Check constraints in tables are identical to check constraints in domains.



Only one constraint is permitted per column. If the column including a constraint is based on a domain also containing a constraint, both constraints are active.

The specification of the keyword `CONSTRAINT` and the name are optional for all constraints. If no name is specified, InterBase/Firebird generates a name automatically. All constraint names are stored in a system table called `DB$RELATION_CONSTRAINTS`.

It is only necessary to name constraints, if they are to be deactivated at a later date using the `ALTER TABLE DROP` statement.

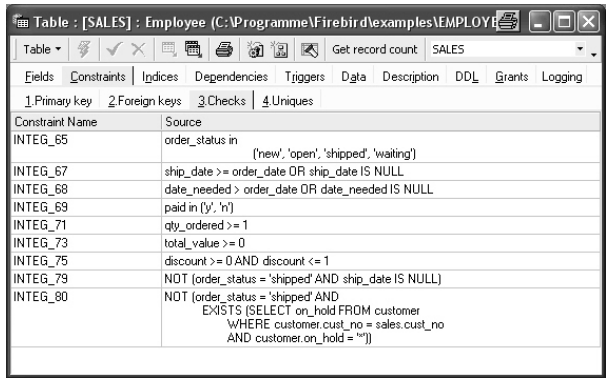
From InterBase 5 onwards, cascading referential integrity is also supported.

3.2.7 Check Constraint

A check is a database examination, which ensures data consistency in the tables among each other. It can be executed automatically and so ensures that data contents are kept consistent by testing them before they are stored in the database.

The check constraint option enables each data set to be examined for validation of the expression in brackets following the check constraint. Check constraints in tables are identical to check constraints in domains.

A check constraint can be specified for each column in a table, to guarantee the mechanism described above. It includes an expression that must be true, so that the data set following an insert or update can be written. The field contents must be included in the permissible values, which can be specified in a list. It is also possible to test the value for a minimum and maximum value. Furthermore the value can be compared to values in other columns, in order to test dependencies.

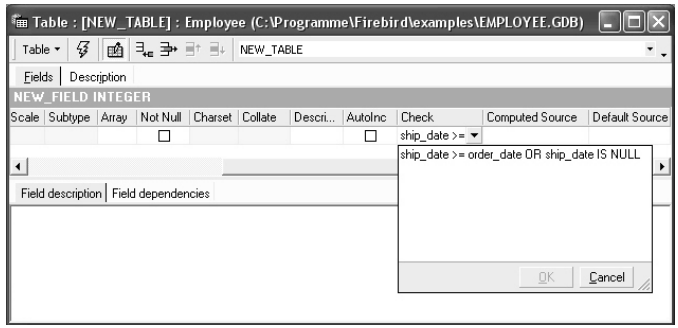


A check constraint can only examine the values in the current data set. When simultaneously inserting or altering multiple data sets, a check constraint can only guarantee one data integrity at a time at data set level.

If other data sets are referenced in the check, these could have been modified by another user at the time of entry, and therefore possibly have become invalid, even though the check constraint's test approved the data set. At the time of a check constraint validation, other data is only read for the check. For this reason, the values for the current operating sequence remain constant, even if another user has modified one of the values already referenced for validation.

A check constraint can be created directly when creating a table. When creating a check constraint, the following criteria should be taken into consideration:

- A check constraint cannot reference a domain.
- A table column can only contain one check constraint.
- A check constraint defined by a domain, cannot be overridden by a local check constraint. However additional constraints can be specified.



3.2.8 Index/Indices

An index can be compared to a book index enabling rapid search capabilities.

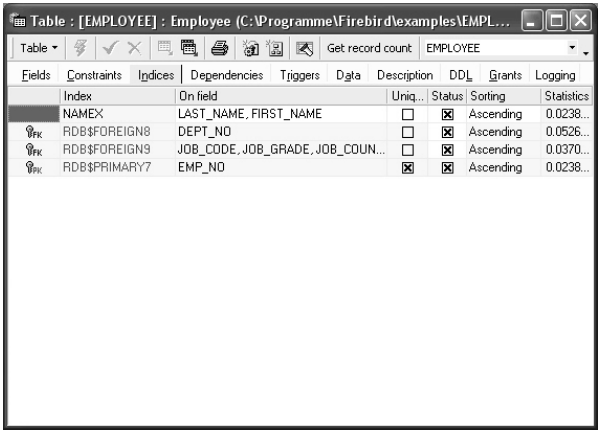
Indices are a sorted list of pointers into tables, to speed data access. They can be best described as an alphabetical directory with internal pointers, where what can be found. If the indexed field is unique there is only one pointer.

An index can be ascending or descending, and can also be defined as unique if wished.

Indices should not be confused with keys. In the relational model, a key is used to organize data logically, so that specific rows can be identified. An index, however, is part of the table's physical structure on-disk, and is used to increase the performance of tables during queries. Indices are therefore not a part of the relational model. In spite of this indices are extremely important for relational database systems.

For columns defined with a primary key or a foreign key in a table, InterBase/Firebird automatically generates a corresponding ascending index and enforces the uniqueness constraint demanded by the relational model.

An index can be defined in the IBExpert Table Editor (started from the DB Explorer):



Indices are updated every time a new data set is inserted, or rather, the index-referenced field is updated. InterBase/Firebird writes an additional second mini version of the data set in each index table.

An index has a sequence e.g. when an ascending index is assigned to a field (default), and a descending select on this field is requested, InterBase/Firebird does not sort using the ascending index. For this a second descending index needs to be specified for the same field.

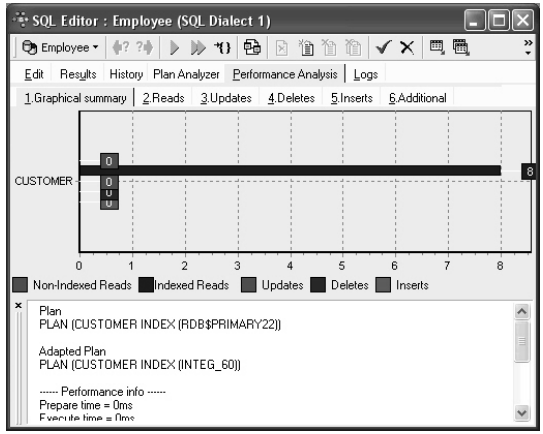
An index can be named as wished; consecutive numbers can even be used, as it is extremely rare that an index is named in SQL.

An index on two fields simultaneously only makes sense when both fields are to be sorted using `ORDER BY`, and this should only be used on relatively small quantities of results.

InterBase/Firebird decides automatically which index it uses to carry out SELECT requests. On the Table Editor / Indices page under Statistics, it can be seen that the index with the lowest value has a higher uniqueness, and is therefore preferred by InterBase/Firebird instead of other indices with a lower level of uniqueness.

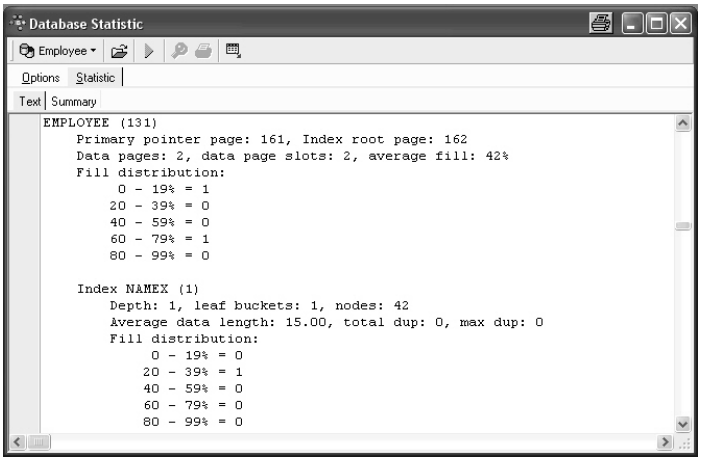
An index should only be used on fields, which are really used frequently as sorting criteria (e.g. fields such as STREET and MALE/FEMALE are generally unimportant) or in a WHERE condition. If a field is often used as a sorting criterion, a descending index should also be considered, e.g. in particular on DATE or TIMESTAMP fields. Care should also be taken that indexed CHAR fields are not be larger than approximately 80 characters in length (with Firebird 1.5 the limit is somewhat higher).

Indices can always be set after the database is actually in use, based on the performance requirements. For further details and examples please refer to Performance Analysis.



Using the IBExpert menu Services / Database Statistics the index depths can be viewed. Leaf buckets display the number of registration leaves, where InterBase/Firebird can access immediately. An index depth of 2, for example, indicates that InterBase/Firebird needs to perform two steps to obtain a result.

Normally the value should not be higher than three. Should this be the case, a database backup and restore should help.

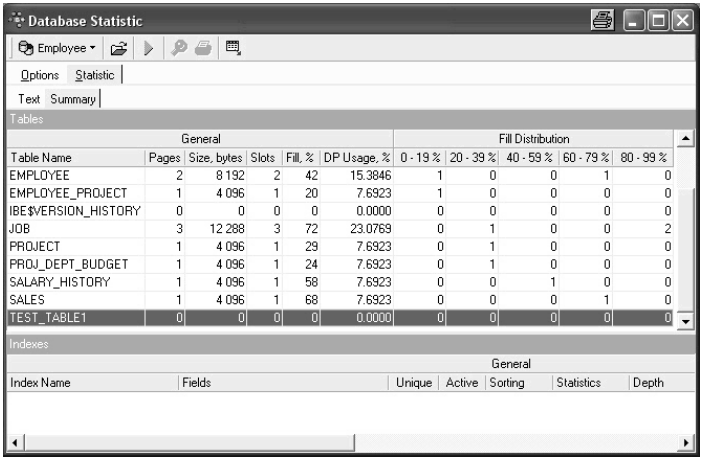


The selectivity is only computed at the time of creation, or when the IBExpert menu item Recompute Selectivity or Recompute All is used (found directly in the Statistic dialog, in the Database menu, or in the right-click DB Explorer menu). Alternatively the

```
SET STATISTIC INDEX {INDEX_NAME}
```

command can be used in the SQL Editor to recompute individual indices.

This is automatically performed during a database backup and restore, as it is not the index, but its definition that is saved, and so the index is therefore reconstructed when the database is restored.



The SQL plan used by the InterBase/Firebird Optimizer merely shows how the server plans to execute the query.

If the developer wishes to override InterBase/Firebird's automatic index selection, and determine the index search sequence himself, this must be specified in SQL.

For example, an index is created in the Employee database:

```
CREATE INDEX EMPLOYEE_IDX1 ON EMPLOYEE(PHONE_EXT);
```

Then:

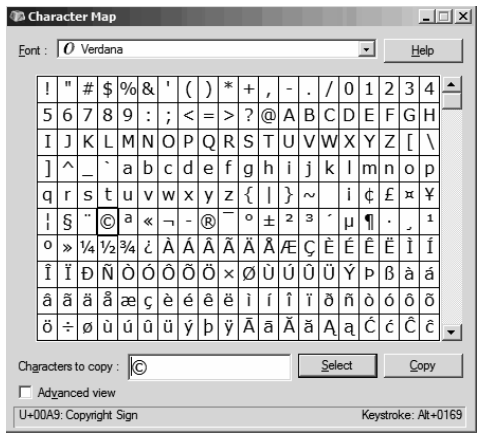
```
SELECT * FROM EMPLOYEE
WHERE EMPLOYEE.PHONE_EXT='250'
PLAN (EMPLOYEE INDEX (EMPLOYEE_IDX1));
```

Each index needs to be named and entered individually.

Indices should be prudently defined in a data structure, as not every index automatically leads to an acceleration in query performance. If in a table, for example, a column comprises data only with the value 0 or 1, an index could even slow performance down. A complex index structure can however have a huge influence upon insertion and alteration processes in the long run.

Ascending

An ascending index searches according to an ascending letter or numeric sequence, depending upon the defined character set (or, if no character set has been specified for the indexed field, the default character set).



Descending

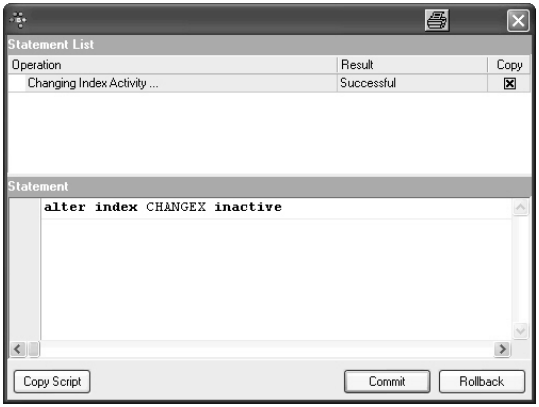
A descending index searches according to a descending letter or numeric sequence, depending upon the defined character set (or, if no character set has been specified for the indexed field, the default character set).

Alter Index

Once an index has been defined it is not possible to alter the following: indexed columns, sort direction or uniqueness constraints. The only way to change any of this information is to drop the index and then to recreate it (see Drop Index).

However the status of an index may be altered to active or inactive. An index should be deactivated when, for example, a large number of data sets are to be added, as an active index would recompute the index each time a data set is input. By deactivating the index, and then reactivating after all the data has been input, the index is only recomputed once.

This can be done simply and directly on the Table Editor / Indices page, by checking or unchecking the relevant boxes in the Status column, then compiling, using the respective Editor icon or [Ctrl + F9], and finally committing.



The SQL syntax is:

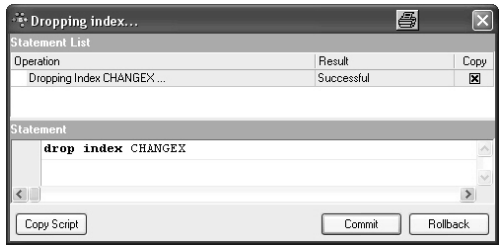
```
ALTER INDEX <index_name> ACTIVE | INACTIVE
```

An index can only be altered by the database creator or by the SYSDBA.

Drop Index/Delete Index

Only user-defined indices can be dropped. As the only alterations permitted on indices are activation and deactivation, indices often need to be dropped and then subsequently recreated, in order to alter certain index information such as indexed columns, sort direction or uniqueness constraints.

Indices can be dropped simply in IBExpert using the Table Editor / Indices page. Mark the index to be dropped and then right-click and select the menu item Drop Index <INDEXNAME> or use the DEL key:



Finally commit or roll back.

Using SQL the syntax is:

```
DROP INDEX Index_Name
```

`DROP INDEX` cannot be used for system-generated indices on primary or foreign keys, or on columns with a uniqueness constraint in the table definition.

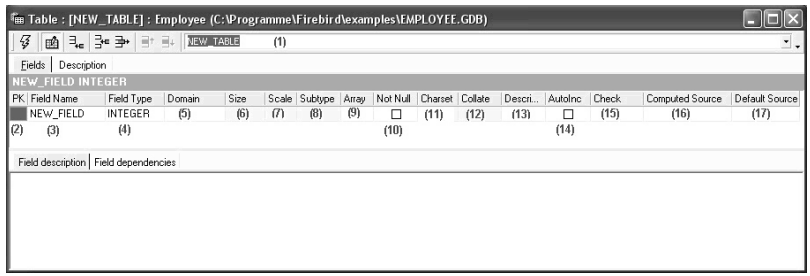
An index can only be dropped by the database creator or by the SYSDBA.

3.2.9 New Table

A new table can be created in a connected database, either by using the menu item Database / New Table, the respective icon in the New Database Object toolbar, or using the DB Explorer right-click menu (or key combination [Ctrl + N]), when the table heading of the relevant connected database is highlighted. A New Table dialog appears, with its own toolbar (Table Editor toolbar), and a pull-down menu (Table button).

When creating a table it is necessary to define a table name that is unique in the database. At least one column must be specified in order to create the table successfully.

Initially a table name is specified **(1)** in the upper row:



All data manipulation operations such as SELECT, INSERT, UPDATE and DELETE are carried out using this name.

Fields:
Furthermore, fields can be defined in the Table Editor. At least one field must be de-

fined, so that the table can be committed and registered as an object in the database (Ctrl + F9). This enables additional table definitions to be made.

An overview of the various input fields is listed below. For details regarding individual subjects, please refer to the "See Also" list below, or use the direct text links.

Since IBExpert version 2.5.0.61 it is also possible to drag 'n' drop fields from the Database Explorer tree and SQL Assistant into the Table Editor's field list, allowing field definitions to be quickly and easily copied from one table to another.

(2) Primary & Foreign Key: In the first column PK one or more fields can be defined as a primary key (double click). A primary key (PK) serves to uniquely identify a data set, and also acts as an index.

(3) Field Name: Each field should be given a logical name.

(4) Field Type: Here the data type can be specified.

(5) Domain: Fields can also be based upon domains. If no domain is specified, InterBase/Firebird generates a system domain for the field as specified.

(6) Size: Specifies the field size.

(7) Scale: Here the number of decimal places can be specified here for all numerical fields.

(8) Subtype: A subtype should be specified for blob fields.

(9) Array: Although arrays contradict all the rules of normalization, there are certain situations (for example storing measurement data), when they are necessary. For more information, please refer to Arrays.

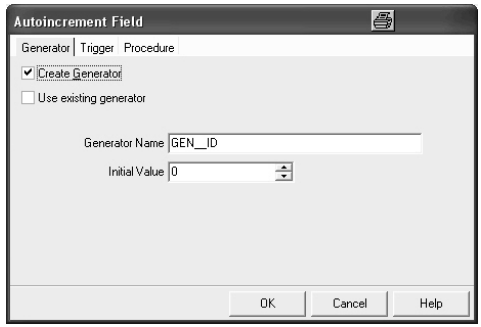
(10) Not Null: This check box can be marked by double-clicking or using the space bar. Not Null forces data to be entered in this field (i.e. the field may not be left empty).

(11) Charset: A character set may be specified for individual fields. This overrides the database default character set. Although this is seldom used, it may be necessary should, for example, Asian, Russian or Arabic addresses need to be input and collated in a database with a European default character set.

(12) Collate: This determines the collation for a character set specified for a field.

(13) Description: Useful for database documentation. The Description Register (i) should be used to describe the table; the Description field for describing the field.

(14) Autoinc: Using the space bar or double-click, a new dialog appears, allowing autoincrements (generator, trigger or stored procedure) to be defined.



(15) Check: Each data set is examined according to an expression defined in brackets for validity. Here certain conditions can be specified (see Check Constraint) causing an automatic database examination during data input, to ensure data consistency in the tables and among each other.

(16) Computed Source: SQL input window for calculations. This can be used for fields containing the results of calculations performed on other fields in the same or other tables in the database.

(17) Default Source: Here a default data entry (text or numeric, depending upon the specified data type) can be specified, e.g. the text NOT KNOWN can be entered as a default source, so that if an address field cannot be input by the user because the information is unavailable, the entry `NOT KNOWN` is automatically entered. It is important to note here, that once a default source has been defined for a field, InterBase/Firebird cannot subsequently alter it (nor subsequently add a default source). The field needs to be dropped, and a new field created.

However, since version 2003.11.6.1 IBExpert has found a way around this. Because the server itself doesn't allow the default value of a field to be altered using `ALTER TABLE` we have implemented a kind of workaround:

First, IBExpert creates the temporary field with the new `DEFAULT` value:

```
ALTER table ADD IBE$$TEMP_COLUMN column_type DEFAULT new_default
```

2. Secondly, IBExpert copies the `RDB$DEFAULT_SOURCE` and `RDB$DEFAULT_VALUE` values of the newly created temporary field into `RDB$DEFAULT_SOURCE` and `RB$DEFAULT_VALUE` of the field which should be altered:

```
UPDATE RDB$RELATION_FIELDS F1
SET
F1.RDB$DEFAULT_VALUE  = (SELECT F2.RDB$DEFAULT_VALUE
                         FROM RDB$RELATION_FIELDS F2
                         WHERE (F2.RDB$RELATION_NAME = 'table')
                               (F2.RDB$FIELD_NAME ='IBE$$TEMP_COLUMN')),
F1.RDB$DEFAULT_SOURCE = (SELECT F3.RDB$DEFAULT_SOURCE
                         FROM RDB$RELATION_FIELDS F3
                         WHERE (F3.RDB$RELATION_NAME = 'table')
                               (F3.RDB$FIELD_NAME ='IBE$$TEMP_COLUMN'))
```

```
WHERE (F1.RDB$RELATION_NAME = 'table')
      (F1.RDB$FIELD_NAME = 'column')
```

3. After that IBExpert drops the temporary field:

```
ALTER TABLE table DROP IBE$$TEMP_COLUMN
```

Tables can, of course, also be created using DDL directly in the SQL Editor, using the following syntax:

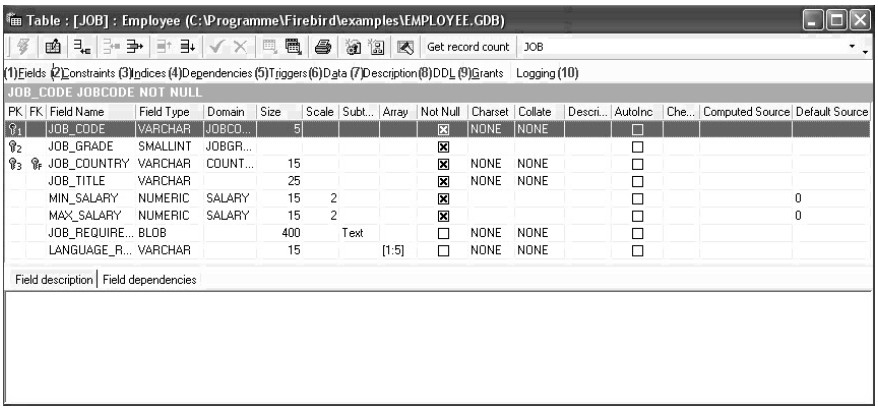

```
CREATE TABLE TABLE_NAME (
COLUMN_NAME1 <COLUMN_DEFINITION>,
COLUMN_NAME2 <COLUMN_DEFINITION>,
...
COLUMN_NAMEn <COLUMN_DEFINITION>;
TABLE_CONSTRAINT1,TABLE_CONSTRAINT2,
...
TABLE_CONSTRAINTn);
```

Once the table has been created do not forget to commit.

3.2.10 Table Editor

The Table Editor can be used to analyze existing tables and their specifications, or to add new fields, specifications etc, in fact, perform all sorts of table alterations. It can be started directly from the DB Explorer by simply double-clicking on the relevant table in the IBExpert DB Explorer, or using the DB Explorer right-click menu Edit Table ... (key combination [Ctrl + O]).

The Table Editor appears:



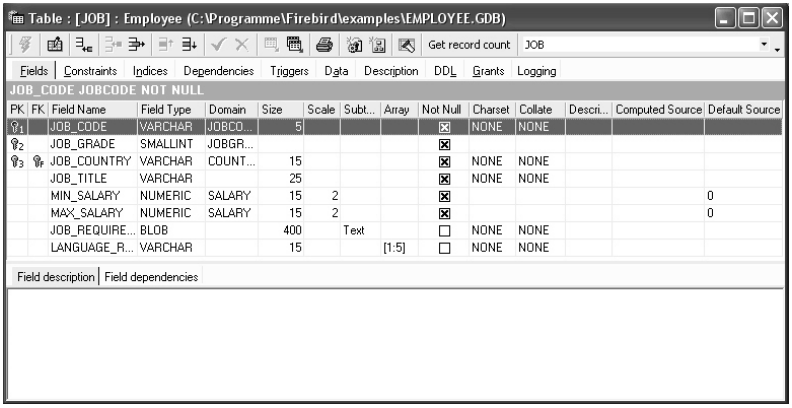
Note: the IBExpert status car shows how many remaining changes may be made to the table before a backup and restore is necessary. (A total of 255 changes may be made to a database object before InterBase/Firebird demands a backup and restore).

The Get Record Count button at the right of the Table Editor toolbar, displays the number of records in the table.

Alternatively for those competent in SQL - the SQL Editor [F12] can be used directly for making table alterations using SQL code.

Support for the InterBase 7.5 temporary tables feature was added in IBExpert version 2004.12.12.1.

Fields



The many possible field specifications are listed under the Fields tab. The individual columns are explained in detail under New Table. Fields can be amended by simply overwriting the existing specification where allowed. Please note that it is not always possible to alter certain fields once data has been entered, e.g. a field cannot be altered to NOT NULL, if data has already been entered which does not conform to the NOT NULL property (i.e. the field has been left undefined). Similarly a primary key cannot be specified following data entries with duplicate values.

New in IBExpert version 2.5.0.61: It is possible to drag 'n' drop fields from the Database Explorer tree and SQL Assistant into the Table Editor's field list, allowing you to quickly and easily copy field definitions from one table to another.

The contents of text blob fields can even be read in the IBExpert Table Editor; simply hold the mouse over the text field, and the full text appears.

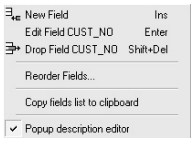
Tip: as with all IBExpert dialogs, the fields can be sorted into ascending or descending order based upon the column where the mouse is, simply by clicking on the column headers (i.e. PK, FK, Field Name etc.).

By double-clicking on the right edge of the column header, the column width can be adjusted to the ideal width.

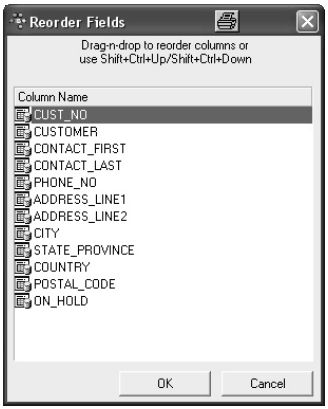
Since IBExpert version 2003.11.6.1 the new Grid menu offers a number of options when working in the Table Editor's Field and Data pages.

Table Editor Right-Click Menu:

The Table Editor Fields page has its own context-sensitive menu using the right mouse button:



This can be used to add a New Field, or edit or drop an existing highlighted field. Fields can also be reordered using drag 'n' drop:



This can also be done using the field navigator icons in the Navigation toolbar (or key combinations [Shift + Ctrl + Up] and [Shift + Ctrl + Down]).

A field list can also be copied to clipboard, and the pop-up Description Editor blended in or out.

New fields can be added using the

icon (or [Ins] key), to open the Adding New Field Editor (please refer to Insert Field for details).

In the lower part of the Table Editor the individual Field Descriptions and Field Dependencies can be viewed. Since IBExpert version 2003.11.6.1, the field dependencies list also includes indices, primary and foreign keys. This new version also enables you to alter the default value of a field.

The Table Editor comprises a number of pages, opened by clicking the tab description, each displaying already specified properties, and allows certain specifications to be viewed/added.

Constraints

Constraints are used to ensure data integrity. Each constraint has its own context-sensitive right mouse button menu, and a new toolbar is displayed offering the most common operations as shortcuts.

The menu offers New Foreign Key [Ins], Drop Foreign Key [Del] and Autowidth. Autowidth automatically adjusts the column widths to fit into the visible dialog width.

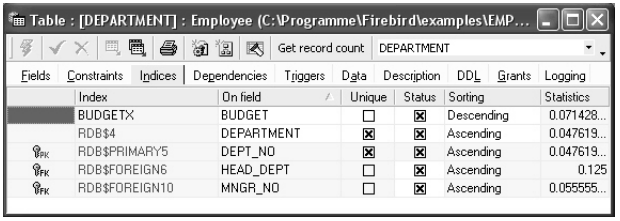
The following can be viewed, added or edited in the Table Editor under the Constraints tab:

- *Primary keys*: A primary key can only be defined at the time of defining a new table.
- *Foreign keys*: A foreign key is a link to another table and stores the primary key of another table.
- *Checks*: Further conditions can be specified by the user (check constraint).
- *Uniques*: All fields defined as unique are also candidate keys.

New to IBExpert version 2003.11.6.1 is the added support for the Firebird feature - user-defined constraint index names. And since IBExpert version 2005.01.12.1 the maximum constraint name length was expanded from 27 to 31.

Indices

Indices already defined for the table can be viewed on the Indices page.



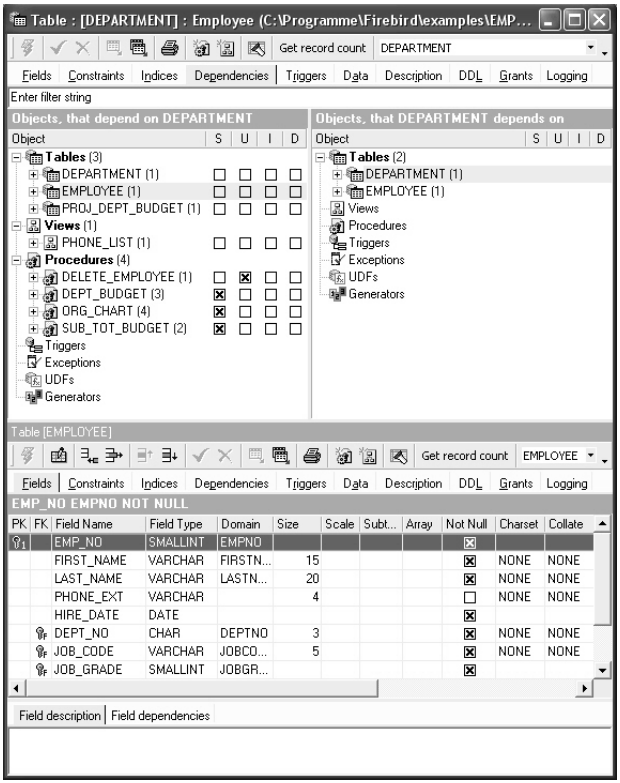
Information displayed includes key status, index name, upon which field the index has been set, whether it is unique, the status (i.e. whether active or inactive), which sorting order and the statistics (displayed in older versions under the column heading Selectivity). Those indices beginning with RDB$ in red, are InterBase/Firebird system indices.

Indices can be added or deleted using the right-click menu or [Ins] or [Del]. Further options offered in the right mouse button menu are:

- Recompute Selectivity
- Recompute All
- Show Statistics (blends the selectivity statistics in and out).

Dependencies

Here the dependencies between database objects can be viewed.



This summary can, for example, be useful if a database table should need to be deleted or table structures altered, or for assigning user rights to foreign key referenced tables. It displays both those objects that are dependent upon the table (left side), and those objects that the table depends upon (right side). The object tree can be expanded or collapsed by using the mouse or [+] or [-] keys, or using the context-sensitive right-click menu items Expand All or Collapse All.

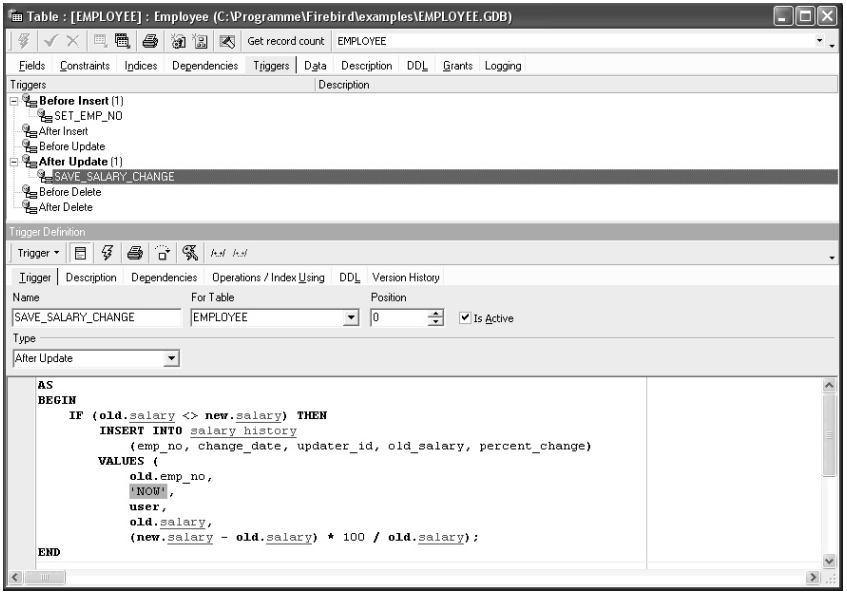
It even shows the actions (when blended in using the right mouse button menu item Show Actions) - S (=Select), U (=Update), I (=Insert) or D (=Drop).

The object code can be viewed and edited in the Table Editor lower panel, provided the Inplace Objects' Editors option has been checked under Options / Environment Options / Tools. If this option is not checked, then the code may only be viewed in the lower panel, and the object editor must be opened by double-clicking on the respective ob-

ject name, in order to make any changes to it. This also applies to all triggers listed on the Triggers page.

Triggers

Triggers are SQL scripts, which are executed automatically in the database when certain events occur.



Similar to dependencies, the triggers are listed in a tree structure according to the following events:

```
BEFORE INSERT
AFTER INSERT
BEFORE UPDATE
AFTER UPDATE
BEFORE DELETE
AFTER DELETE
```

The object tree can be expanded or collapsed by using the mouse or [+] or [-] keys (or using the right-click menu).

When a trigger is highlighted, the right mouse button menu offers options to create a new trigger, edit or drop the highlighted trigger, or set the marked trigger to inactive/active.

The trigger code can be viewed and edited in the Table Editor lower panel, provided the Inplace Objects' Editors options has been checked under Options / Environment Options / Tools. If this option is not checked, then the code may only be viewed in the lower panel, and the Trigger Editor must be opened by double-clicking on the respec-

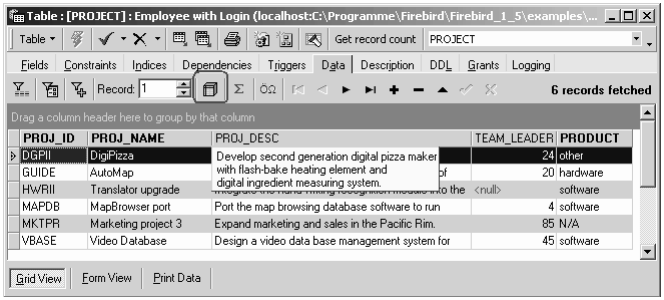
tive trigger name, in order to make any changes to the trigger. This also applies to all objects listed on the Dependencies page.

Data Grid

Here the data in the database table can be manipulated (i.e. inserted, altered or deleted) directly.

There are three modes of view:

1. Grid View - all data is displayed in a grid (or table form).



The data sets can be sorted according to any field in either ascending or descending order by simply clicking on the column header.

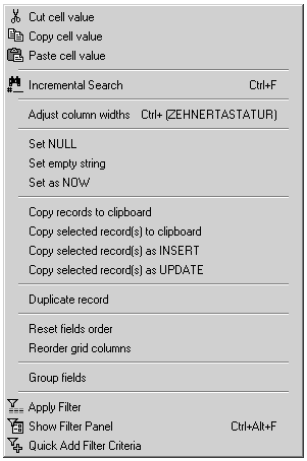
The contents of blob and memo fields can be read by simply holding the cursor over the respective field.

A new feature in IBExpert version 2004.10.30.1 is the OLAP and data warehouse tool, Data Analysis, opened using the Data Analysis icon (highlighted in red in the above illustration).

There are many options to be found under Options / Environment Options / 6. Grid, which allow the user to customize this grid. Under the IBExpert menu item Register Database or Database Registration Info there are additional options, for example, Trim Char Fields in Grids.

Since IBExpert version 2003.11.6.1 the new Grid menu offers a number of options when working in the Table Editor's Field and Data pages.

The Data page Grid View also has its own context-sensitive menu, opened by right-clicking with the mouse. This includes the following options:



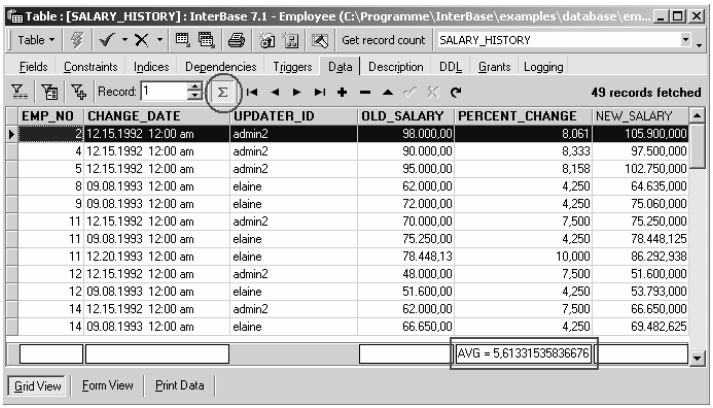
- **Cut, Copy and Paste** functions.
- **Incremental Search** [Ctrl + F] allows a simple search for individual entries by simply marking the desired column header, clicking the right mouse button menu item Incremental Search [Ctrl + F] and then typing the relevant digits/letters, until the required dataset(s) is/are found.
- **Adjust Columns widths** (or [Ctrl + '+' NUMBLOCK] adjusts all column widths in the grid view to the ideal width.
- **SET** commands - set field as `NULL` or empty.
- **Copying operations** - copies all or one or more selected records to clipboard, as `INSERT` or as `UPDATE`. Multiple records may only be selected if the *Allow Multiselect* option has been checked in the Options menu: Environment Options / Grid.
- **Duplicate record** option.
- **Reset fields order** - returns the field order to the original (not available in SQL Editor / Results).
- **Reorder grid columns** - simply using drag 'n' drop.
- **Group/Ungroup Fields** - offers an alternative visual option, whereby column headers may be dragged 'n' dropped into the new gray bar, which appears directly above the column headers, when this menu item is clicked, so allowing data to be grouped. Please note that the option *Allow Records Grouping* must be checked in the Options menu: Environment Options / Grid.
- **Filter options** - these can also be found in the data page toolbar (see below).

Both the Grid and Form Views offer a Navigation toolbar, allowing the data to be moved, inserted, altered and deleted. Furthermore data can be filtered using the Filter Panel toolbar. (Please refer to Filter Panel for further information.)

Since IBExpert version 2004.8.26.1 it is also possible to display data as Unicode. Simply click the relevant icon in the Navigation toolbar or use [F3] (see illustration below). It is not possible to edit the data directly in the grid. To edit data in Unicode, use the Form View or modal editor connected with string cell.

Since IBExpert version 2004.8.5.1 there is the added option to calculate aggregate functions (`COUNT`, `SUM`, `MIN`, `MAX`, `AVG`) on numeric and datetime columns. Simply click

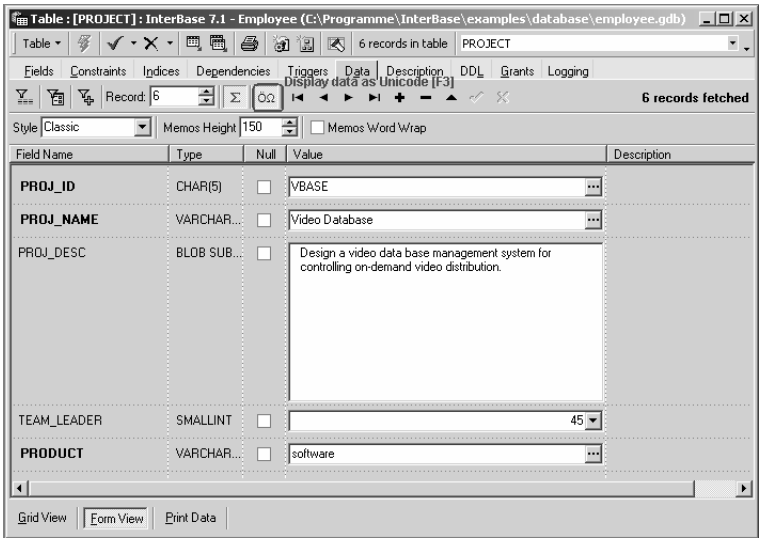
Show summary footer button on the toolbar of the data view to display the summary footer:



It is then possible to select an aggregate function for each numeric/datetime column separately.

IMPORTANT: this feature performs all calculations on the client side, so do not use this function on huge datasets with millions of records because IBExpert will fetch all records from the server before calculating.

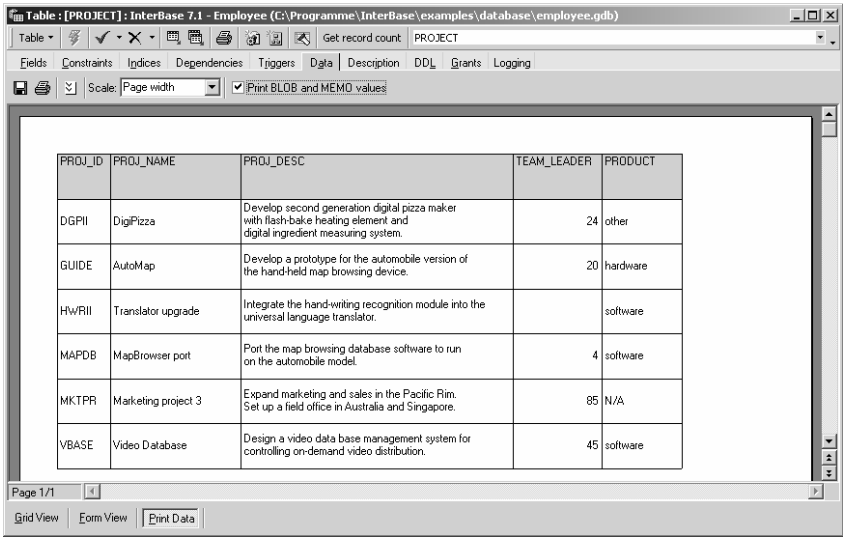
2. Form View - one data set is displayed at a time in a form.



New to version 2004.8.26.1: The Form View has been completely redesigned. It now also displays field descriptions. It is also possible to select alternative layouts (classic or compact), the compact alternative for those who prefer a more compact and faster

interface. Visual options now also include specification of Memo Height and Memo Word Wrap.

3. Print Data - displays data in WYSIWYG mode (the status bar showing which page number is currently visible and how many pages the data covers altogether). The data can be either saved to file or printed. The Print Data view also has its own right-click menu, enabling size adjustments (2 pages, whole page, page width, and scaling from 10% to 200%), this being also available as a pull-down list of options in the Print Preview toolbar. Further toolbar options include saving the information to file, printing directly, and specifying the page set up. There is even a check option to specify whether BLOB and MEMO values should be printed or not.



PROJ_ID	PROJ_NAME	PROJ_DESC	TEAM_LEADER	PRODUCT
DGPII	DigiPizza	Develop second generation digital pizza maker with flash-bake heating element and digital ingredient measuring system.	24	other
GUIDE	AutoMap	Develop a prototype for the automobile version of the hand-held map browsing device.	20	hardware
HWRII	Translator upgrade	Integrate the hand-writing recognition module into the universal language translator.		software
MAPDB	MapBrowser port	Port the map browsing database software to run on the automobile model.	4	software
MKTPR	Marketing project 3	Expand marketing and sales in the Pacific Rim. Set up a field office in Australia and Singapore.	85	N/A
VBASE	Video Database	Design a video data base management system for controlling on-demand video distribution.	45	software

IBExpert also offers a Test Data Generator (IBExpert Tools menu), should test data be required for testing query times etc.

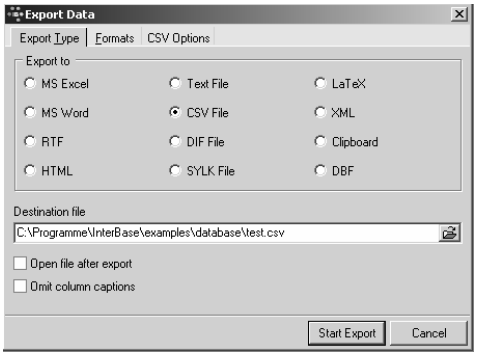
Note that when deleting data, the InterBase/Firebird database becomes larger, as the data is merely flagged as deleted, due to the rollback option, which is available until the drop commands are committed.

Export Data

Data can be exported from the Data page in the Table Editor and from the Results page in the SQL Editor, by simply clicking the

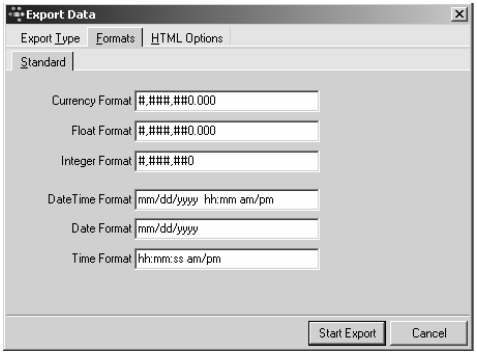
or using the key combination [Ctrl + E].

The first page in the Export Data dialog, Export Type, offers a wide range of formats, which can be simply and quickly specified per mouse click (or using the directional keys).



The destination file name must also be specified, and check options allow you to specify whether the resulting export file should be opened following the data export or not, and - for certain export formats - whether column headings should be omitted or not.

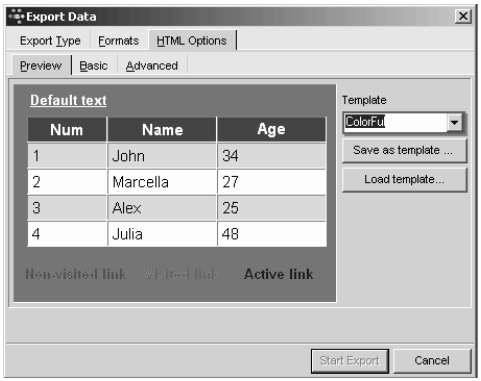
The Formats page is available for all export types, with the exception of XML.



Here it is possible to specify a range of numerical formats, including currency, float, integer, date, time or date and time. Please note that not all of these options may be altered for all export types (for example when exporting to DBF it is only possible to specify the formats for date/time and time).

Depending upon which format has been specified, additional options my be offered on the third page, for example:

- **Excel** - specification of page header and footer.
- **HTML** - template selection and preview, title, header and footer text as well as a wide range of advanced options.
- **CSV** - Quote String check option, and user specification of CSV separator.
- **XML** - Encoding format may be selected from a pull-down list. There are also check options to export String, Memo and DateTime fields as text.
- **DBF** - check options to export strings to DOS, long strings to Memo, and to extract DateTime as Date.

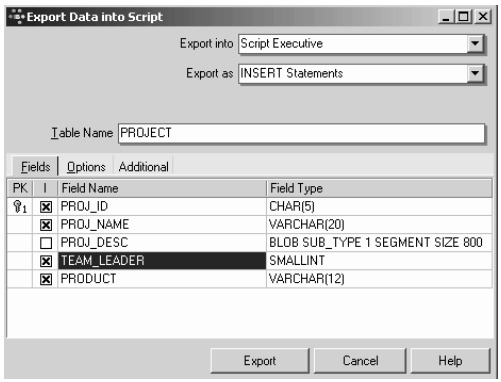


The export is then finally started using the Start Export button in the bottom right-hand corner. Following a successful export, a message appears informing of the total number of records exported.

Export Data into Script

The Export Data into Script dialog can be started using the

on the Data page in the Table Editor or the Results page in the SQL Editor.

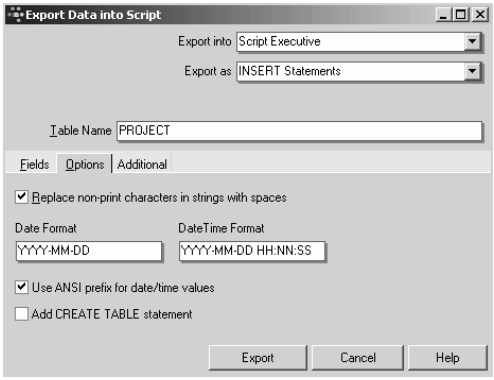


The following options may be selected before starting the export:

- Export into: File, Clipboard or Script Executive
- Export as: `INSERT` statements, `UPDATE` statements or since version 2003.12.18.1 there is also the added possibility to export data as a set of `EXECUTE PROCEDURE` statements.
- Specify the file name if exporting to file and the table name from which the data is to be exported.

The Fields page allows the table fields to be selected.

The Options page:



offers a number of options for date and time specification, how many records should be exported before committing, and whether the CREATE TABLE statement should be added into the script.

The Additional page allows additional definitions for query to be made, for example, ORDER BY or WHERE clauses.

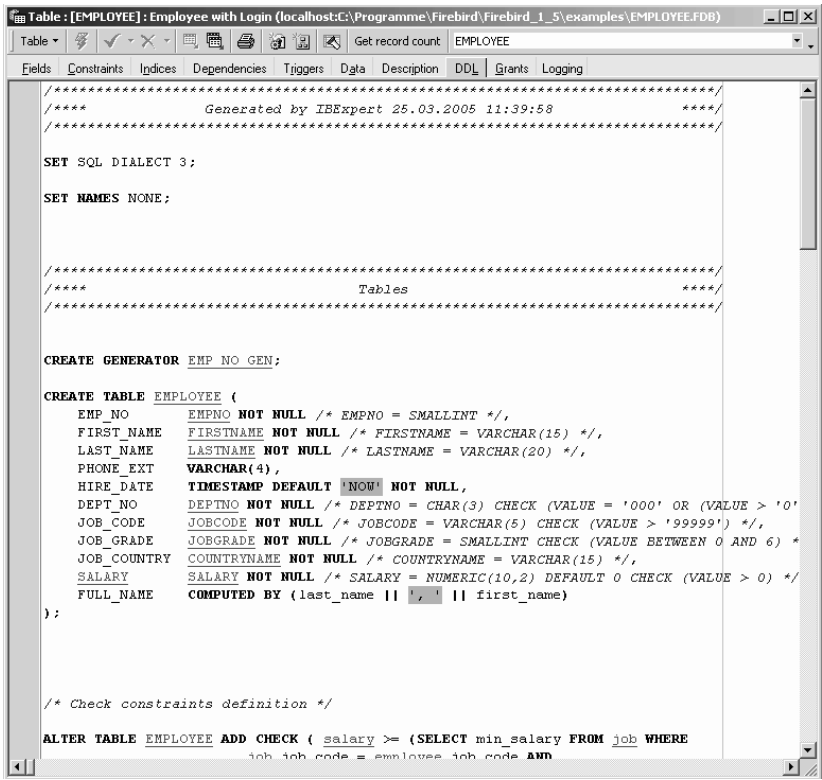
After completing all specifications as wished, simply click the Export button to perform the data export.

Description

As with the majority of the IBExpert Editors, the Table Editor's Description page can be used to insert, edit and delete text by the user as wished. It enables the database to be simply and quickly documented.

DDL

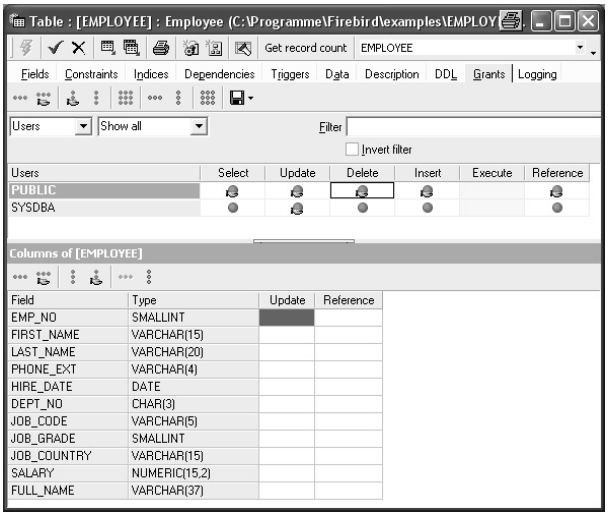
This displays the database table definition as SQL script.



This DDL text cannot be edited, but it can be copied to the clipboard.

Grants

Here individual users can be assigned rights to SELECT, UPDATE, DELETE and INSERT for the current table. In some cases rights can also be assigned to individual fields.



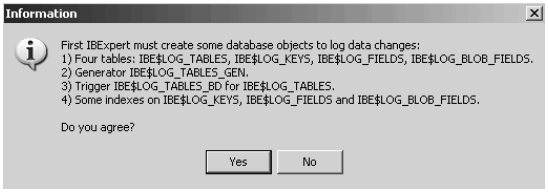
Using the pull-down list, grants can also be assigned for not just users and roles, but also for views, triggers and procedures in the same database, without having to leave the Table Editor.

For more details regarding this subject, please refer to Grant Manager.

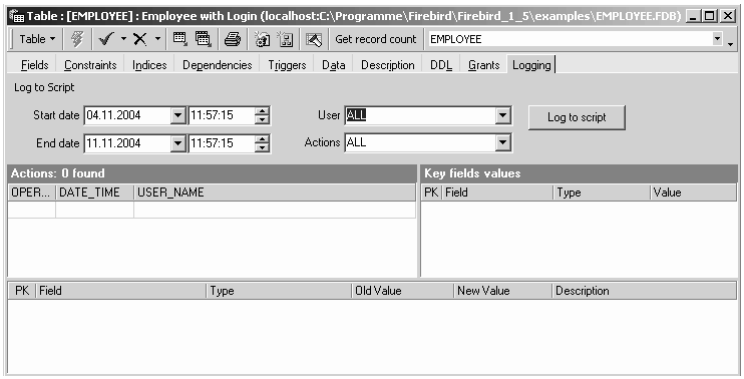
Logging

Data manipulation can be documented here in system tables generated by IBExpert.

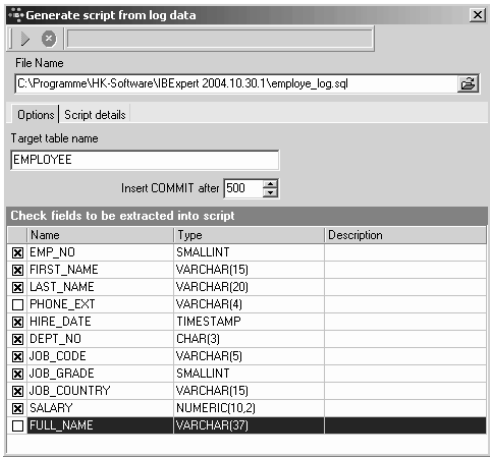
When this page is opened for the first time, IBExpert asks whether it should generate certain system tables:



After confirming and committing, work can be started immediately!



Log to script by clicking the button.



The log file name, how often should be committed and which fields should be logged can be stipulated on the Options page. And the beginning and end of script may be specified under Script Details if wished. The script can then simply be generated using the respective icon or [F9].

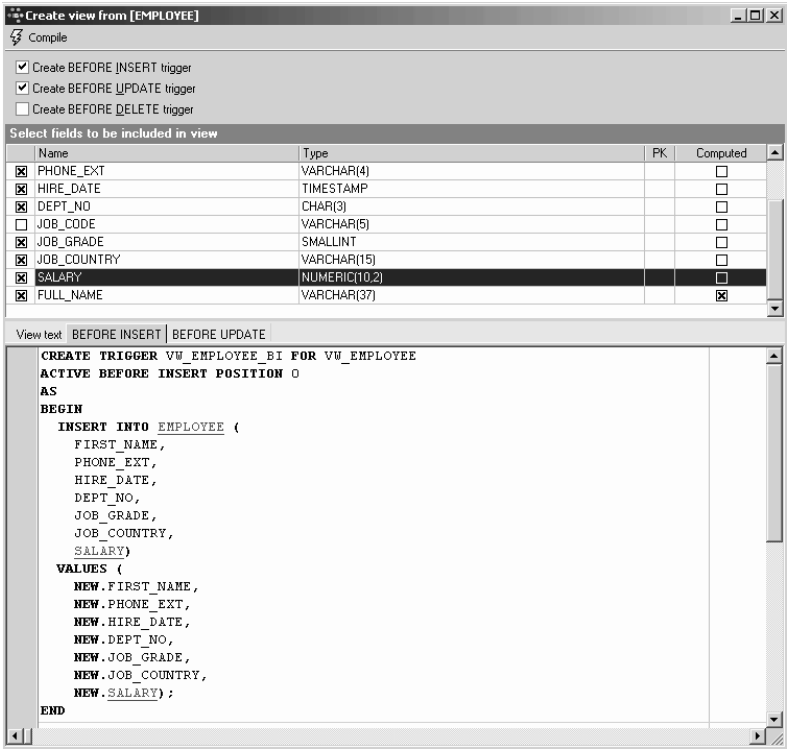
Create View from Table (Updatable View)

It is possible to create a view directly from a table, using the Table Editor's Create View icon:

A view default name is automatically generated by IBExpert, comprising the prefix VW_ followed by the table name. This can of course be overwritten if wished.

The list of fields to be included in the view may be specified by simply clicking on the check boxes to the left of the field names, or by double-clicking or using the space bar on a selected field.

The view text is displayed in the lower window and may also be amended as wished.



One or more trigger types may be specified - whereby further tabs appear in the lower area, allowing the pre-defined trigger code to be simply amended as wished, automatically creating an updatable view - this is, in fact, an extremely quick and simple way to create a view that is updatable, and which can otherwise only be realized with considerable manual labor! These triggers are already prepared, and require little work (click the trigger type checkbox, uncheck unnecessary fields) in order to create an updatable view.

As with the view default name, the trigger default name is automatically generated by IBExpert, comprising the prefix `VW_` followed by the table name and ending with the trigger-type suffix (`_BI` = Before Insert, `_BU` = Before Update, `_BD` = Before Delete). This can of course be overwritten if wished.

Finally compile and commit to create the new view or updatable view.

Create Procedure from Table

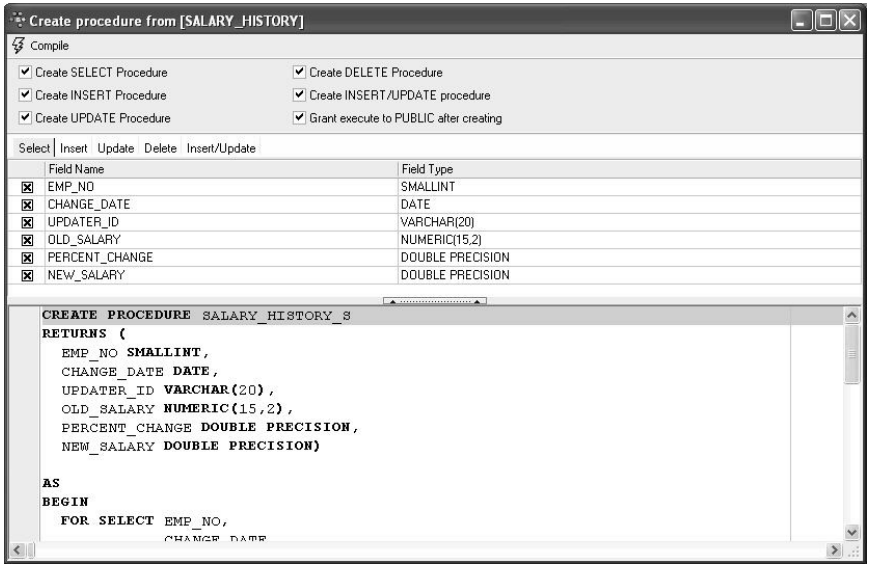
A procedure can be created directly from a table, using the Table Editor's Create Procedure icon:

The sort of procedure to be created can be specified by checking/unchecking the boxes in the upper area. Options include:

- SELECT
- INSERT
- UPDATE
- DELETE
- INSERT/UPDATE

with a further checkbox option to:

- Grant execute to PUBLIC after creating.



A procedure default name is automatically generated by IBExpert, comprising the table name followed by one of the following suffixes:

- S = Select
- I = Insert
- U = Update
- D = Delete
- IU = Insert/Update

This name can of course be overwritten or altered directly in the code if wished.

The list of fields to be included in the procedure may be specified as wished by simply clicking on the check boxes to the left of the field names, or by double-clicking or using the space bar on a selected field.

The procedure text is displayed in the lower window and may also be altered if wished. Switch from one page to the next by clicking on the tabs (displayed above the fields lists).

Finally compile and commit to create the new procedure.

Print Table

Please refer to the IBExpert Edit Menu item Print and the Table Editor Menu item Printing Options.

Print Preview and Print Design

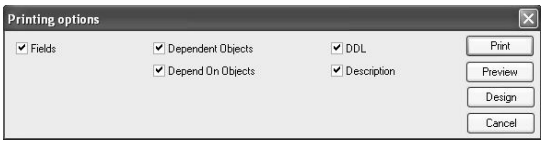
Please refer to the IBExpert Report Manager for further information.

Printing Options

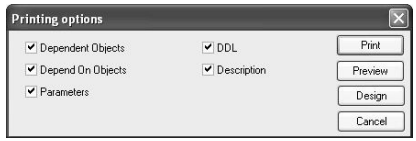
The Printing Options dialog can be started using the Print Table Metadata icon or [Shift + Ctrl + P].

The Printing Options dialog offers different options depending upon which Editor it is started from. For example, when started from the Table Editor:

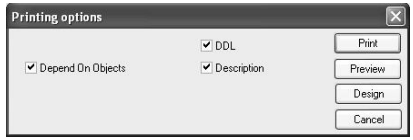
the View Editor:



the Procedure Editor:



the Trigger Editor:



These options include the following:

- Fields
- Constraints
- Indices
- Dependent Objects
- Depend On Objects
- Parameters
- DDL
- Description

Simply check as wished, and then click Preview (to view the report as it will be printed - see Print Preview for further information), Design (to customize the report - refer to Report Manager for further information) or Print to proceed to the standard Windows Print dialog.

3.2.11 Alter Table

A table can be altered to change its defined structure. It is even possible to perform multiple changes simultaneously.

Alterations can be made in the Table Editor, opened by double-clicking on the table name in the DB Explorer. Alternatively use the DB Explorer's right mouse-click menu item Edit Table or key combination [Ctrl + O].

The following operations may be performed when altering a table:

- Add fields
- Add table level constraints
- Drop fields
- Drop table level constraints
- Modify fields

When dropping fields, it is important to note that the column may not be part of the table's primary key, have a foreign key relationship with another table, contain a unique constraint, be part of a table constraint or part of another column's CHECK constraint.

For further details please refer to Table Editor.

The Constraints page in the Table Editor lists all such fields, so that the developer can quickly ascertain whether constraint alterations/deletions are necessary, before dropping the field in question (or whether, in fact, the field should be dropped at all!).

Using SQL the syntax is:

```
ALTER TABLE <table_name>
ADD <field_name> <field_definition>
ADD CONSTRAINT <constraint_name> <constraint_definition>
DROP CONSTRAINT <constraint_name>
DROP <field_name>;
```


3.2.12 Create SIUD Procedures

By right-clicking on a table in the DB Explorer, you will find a menu item called Create SIUD Procedures. SIUD is the abbreviation for `SELECT`, `INSERT`, `UPDATE` and `DELETE`.

If you want to prevent database users from directly manipulating data with insert, update and delete statements, you can use these procedures, which can be executed.

Please refer to Create Procedure from Table for details.

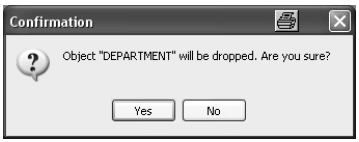
3.2.13 Drop Table/Delete Table

When a table is dropped, all data, metadata and indices in this table are also deleted from the database.

A table can only be dropped, if it is not being used at the time of execution of the DROP command and is not referenced by any other database object, such as in a foreign key relationship, a computed source column or a CHECK constraint for another table, or is a part of the definition of a view or a stored procedure or trigger.

Any existent dependencies can be easily viewed on the Table Editor / Dependencies page. Most database objects can be dropped here directly from the Dependencies page or the Dependencies Viewer by right-clicking on the selected object, and choosing the menu item Drop Object or [Ctrl + Del].

To drop a table use the DB Explorer, right-click and select the menu item Drop Table or [Ctrl + Del]. IBExpert asks for confirmation:



before finally dropping the table. Once dropped, it cannot be retrieved; the table has to be recreated, if a mistake has been made!

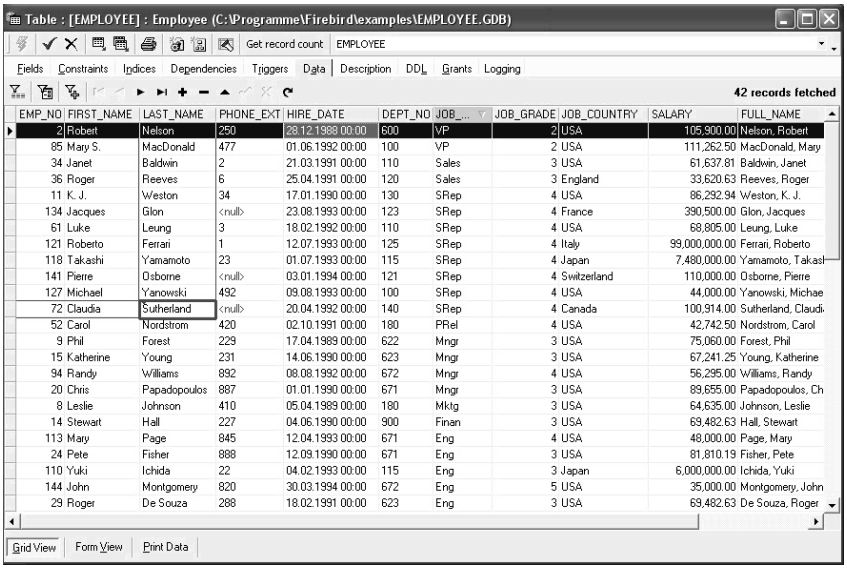
Using SQL the syntax is:

```
DROP TABLE <table_name>;
```

3.3 Field

A field can be defined as the intersection in a table where a row meets a column, containing a clearly differentiated atomic piece of information.

Each data field should be unique and represent and indivisible quantity of information.



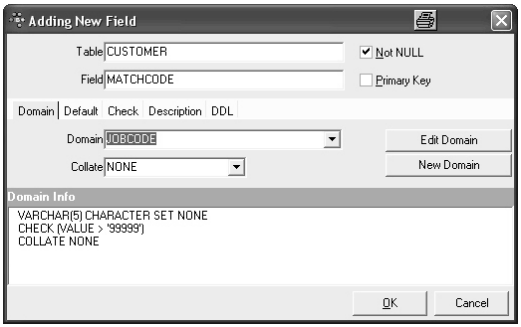
EMP_NO	FIRST_NAME	LAST_NAME	PHONE_EXT	HIRE_DATE	DEPT_NO	JOB_...	JOB_GRADE	JOB_COUNTRY	SALARY	FULL_NAME
2	Robert	Nelson	250	28.12.1988 00:00	600	VP	2	USA	105,900.00	Nelson, Robert
85	Mary S.	MacDonald	477	01.06.1992 00:00	100	VP	2	USA	111,262.50	MacDonald, Mary
34	Janet	Baldwin	2	21.03.1991 00:00	110	Sales	3	USA	61,637.81	Baldwin, Janet
36	Roger	Reeves	6	25.04.1991 00:00	120	Sales	3	England	33,620.63	Reeves, Roger
11	K. J.	Weston	34	17.01.1990 00:00	130	SRep	4	USA	86,292.94	Weston, K. J.
134	Jacques	Glon	<null>	23.08.1993 00:00	123	SRep	4	France	390,500.00	Glon, Jacques
61	Luke	Leung	3	18.02.1992 00:00	110	SRep	4	USA	68,805.00	Leung, Luke
121	Roberto	Ferrari	1	12.07.1993 00:00	125	SRep	4	Italy	99,000,000.00	Ferrari, Roberto
118	Takashi	Yamamoto	23	01.07.1993 00:00	115	SRep	4	Japan	7,480,000.00	Yamamoto, Takasl
141	Pierre	Osborne	<null>	03.01.1994 00:00	121	SRep	4	Switzerland	110,000.00	Osborne, Pierre
127	Michael	Yanowski	492	09.08.1993 00:00	100	SRep	4	USA	44,000.00	Yanowski, Michae
72	Claudia	Sutherland	<null>	20.04.1992 00:00	140	SRep	4	Canada	100,914.00	Sutherland, Claudi
52	Carol	Nordstrom	420	02.10.1991 00:00	180	PRel	4	USA	42,742.50	Nordstrom, Carol
9	Phil	Forest	229	17.04.1989 00:00	622	Mngr	3	USA	75,060.00	Forest, Phil
15	Katherine	Young	231	14.06.1990 00:00	623	Mngr	3	USA	67,241.25	Young, Katherine
94	Randy	Williams	892	08.08.1992 00:00	672	Mngr	4	USA	56,295.00	Williams, Randy
20	Chris	Papadopoulos	887	01.01.1990 00:00	671	Mngr	3	USA	89,655.00	Papadopoulos, Ch
8	Leslie	Johnson	410	05.04.1989 00:00	180	Mktg	3	USA	64,635.00	Johnson, Leslie
14	Stewart	Hall	227	04.06.1990 00:00	900	Finan	3	USA	69,482.63	Hall, Stewart
113	Mary	Page	845	12.04.1993 00:00	671	Eng	4	USA	48,000.00	Page, Mary
24	Pete	Fisher	888	12.09.1990 00:00	671	Eng	3	USA	81,810.19	Fisher, Pete
110	Yuki	Ichida	22	04.02.1993 00:00	115	Eng	3	Japan	6,000,000.00	Ichida, Yuki
144	John	Montgomery	820	30.03.1994 00:00	672	Eng	5	USA	35,000.00	Montgomery, John
29	Roger	De Souza	288	18.02.1991 00:00	623	Eng	3	USA	69,482.63	De Souza, Roger

Each database field has a name, which enables the data to be accessed. A database field can be based on a domain definition or defined individually in the IBExpert Create Table or Table Editors, in which case InterBase/Firebird automatically creates a system domain for the field definition.

3.3.1 Adding New Field (Insert Field) using the Field Editor

Fields can be inserted into a table at the time of table creation, using the IBExpert DB Explorer or menu item New Table. It is however often necessary to add new fields, after the table has been created. This can be easily done in IBExpert by opening the Table Editor (double-click on the relevant table in the IBExpert DB Explorer) or using the DB Explorer right-click menu Edit Table ... (or key combination [Ctrl + O]), and then inserting a field using the

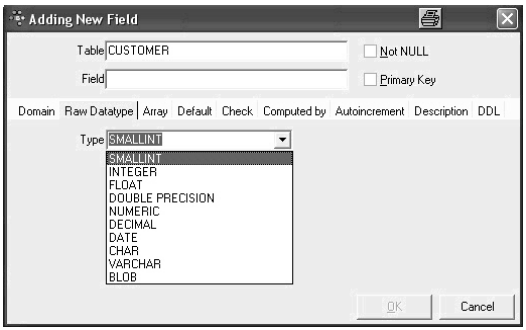
Add Field icon (or [Ins] key) or the Table Editor right-click menu Insert Field, to open the Adding New Field Editor.



The Adding New Field Editor displays the table name, into which the field is to be inserted. The new field name can be specified by the user, along with the parameters Not Null and Primary Key. Further options are to be found on the Default and Check pages, and the usual IBExpert Desc (=Description) and DDL (= Data Definition Language) information pages are also included.

The new field may be based upon an existing domain (which may be edited using the Edit button) or a New Domain can be created directly from the New Field Editor. All existing domains (in the connected database) can be viewed in the "Domain" pull-down list. The domain information can be viewed in the Editor's lower panel.

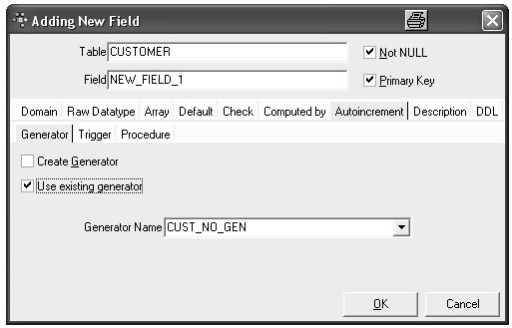
It is also possible to define certain numeric formats as standard using the Options menu, Environment Options / Grid / Display Formats, if wished. These format standards can be overwritten in individual fields here in the Field Editor.



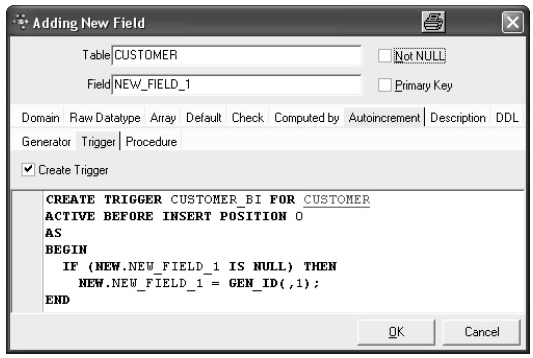
Of course a new field doesn't have to be based on a domain. The data type can be specified using the pull-down list under the Raw Data type tab. However, InterBase/Firebird automatically generates a system domain for all specified fields, so when a new field is inserted, or existing field altered, InterBase/Firebird inserts or alters the respective system domain.

Additional context-sensitive input fields appear, relevant to the data type selected (e.g. When Varchar is selected, options for specifying Length, Charset, and Collate are offered; in the case of Numeric, Precision and Scale can be specified).

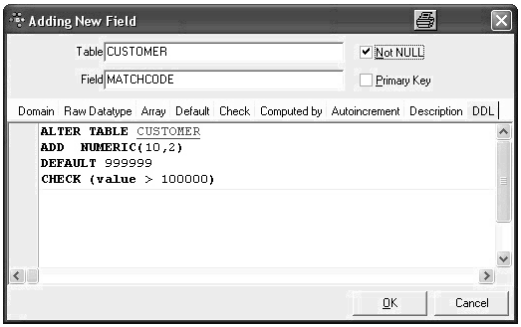
Furthermore arrays can be defined, as well as default values, check constraints, 'computed by' calculations and autoincrements.



The autoincrement tab allows new generators to be created, or an existing generator to be selected. New triggers and procedures can also be created directly here in this Editor for this field, if desired.



As with the majority of the IBExpert Editors, the last two pages display the object Description (which can be inserted, edited and deleted here by the user as wished), and the DDL page,



which displays the SQL code for the field as specified by the user.

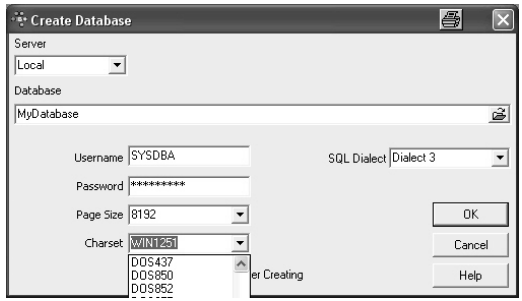
3.3.2 Charset / Character Set

A character set is specified in InterBase/Firebird to define which characters are allowed in a CHAR, VARCHAR or blob field. It also provides collation options when InterBase/Firebird needs to sort a column.

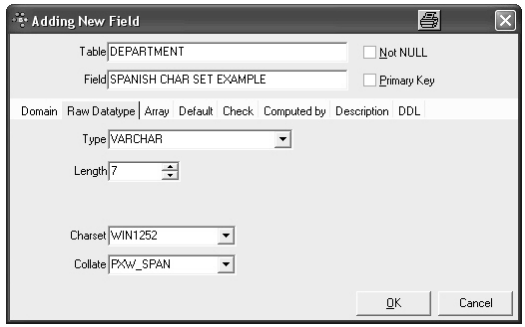
Character set definition becomes increasingly important as the world of database programming spreads more and more across national borders. Today it is often necessary for applications to also meet the requirements of other countries. The problem of multilingual interfaces is just one aspect of internationalization. A modern application needs to handle the particularities specific to individual countries such as, for example, sorting order (collation). In the German language the umlauts ä, ö und ü are integrated in the alphabet using the letter combinations ae, oe and ue. At the same time there are also special characters in the French language, which are not used in the German language such â, á and à. There are completely different problems with versions whose characters are not known in the European character sets, for example Korean or Chinese. These character sets also often contain many more characters, which cannot be incorporated in the 8 bit character sets, as the technical upper limit lies at 256 (=28) different characters. For this reason InterBase/Firebird implements character set support.

Important character sets are, for example, ISO8859_1, to be recommended is Win1252 - the West European character set. Unicode_FSS is the global character set, however there is hardly a program that can read this; Win1251 is the East European character set.

Character sets can be defined for the database (default character set):



or for domains and fields (where the collation can also be specified):



Overview of the main character sets

by Stefan Heymann

Character Sets are an issue every programmer has to deal with one day. This is an overview of the most important character sets.

Name	*Bytes per Character*	*Description*	*Range*	*IANA/MIME Code*
7-bit ASCII	1	The mother of all character sets. Contains 32 invisible control characters, the Latin letters A-Z, a-z, the Arabic digits 0-9 and a bunch of punctual characters. Code Range 0..127.	0..127	US-ASCII

Unicode-based Character Sets

Unicode, ISO 10646	N.A.	A universal code for all characters anyone can think of. Defines characters, assigns them a scalar value, but does not define how characters are rendered graphi-	U+0000..U+100000	N.A.

		cally or in memory.		
UTF-8	1..6	A Unicode transformation format which uses 1-Byte characters for all 7-bit US-ASCII characters and sequences of up to 6 bytes for all other Unicode characters.	All Unicode characters	UTF-8
UCS-2	2	A unicode transformation format which uses 2 Bytes (16 Bits) for every character. This character set is not able to render all Unicode scalars and is therefore obsolete. However, it is still used by a lot of systems (Java, NT)	U+0000..U+FFFF	ISO-10646-UCS-2
UTF-16	2	A unicode transformation format which uses 2 Bytes (16 Bits) for every character. Using the concept of "Surrogate Pairs", this format is able to render all Unicode characters.	All Unicode characters	UTF-16
UCS-4, UTF-32	4	Two unicode transformation formats which use 4 Bytes (32 Bits) for every character. UCS-4 and UTF-32 are the only character sets, which are able to render all Unicode characters in equally long words. UCS-4 and UTF-32 are technically identical.	All Unicode characters	ISO-10646-UCS-4 UTF-32

Single-byte Character Sets

ISO 8859-x	1	An extension of US-ASCII using the eighth bit.	0..127, 160..255	ISO-8859-x
Windows 125x	1	Equal to ISO 8859-x, plus additional characters in the 128..159 range.	0..255	Windows-125x

ISO 8859-x Character Sets

Name	*Covered Languages*	*MS Windows counterpart*	
ISO 8859-1	Latin-1	Windows-1252	
ISO 8859-2	Latin-2	Central and East European languages (Czech, Polish, etc.)	Windows-1250
ISO 8859-3	Latin-3	South European, Maltese, Esperanto	

ISO 8859-4	Latin-4	North European	
ISO 8859-9	Latin-5	Turkish	Windows-1254
ISO 8859-10	Latin-6	Nordic (Sami, Inuit, Icelandic)	
ISO 8859-13	Latin-7	Baltic	Windows-1257
ISO 8859-14	Latin-8	Celtic	
ISO 8859-15	Latin-9	Similar to ISO 8859-1, adds Euro sign (€) and a few other characters	

MS Windows Character Sets

Number	*Name*
1250	Latin 2
1251	Cyrillic
1252	Latin 1
1253	Greek
1254	Latin 5
1255	Hebrew
1256	Arabic
1257	Baltic
1258	Viet Nam
874	Thai

Declaring character sets in XML and HTML (IANA charset definitions)

by Stefan Heymann

Declaring character sets in XML

Every XML document or external parsed entity or external DTD must begin with an XML or text declaration like this:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
```

In the encoding attribute, you must declare the character set you will use for the rest of the document. You should use the IANA/MIME-Code from Character Set Overview.

Declaring character sets in HTML

In the head of an HTML document you should declare the character set you use for the document:

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-
1252">
  ...
</head>
```


Without this declaration (and, by the way, without an additional DOCTYPE declaration), the W3C Validator will not be able to validate your HTML document.

IANA Character Set Definitions

The Internet Assigned Numbers Authority IANA maintains a list of character sets and codes for them. This list is:

IANA-CHARSETS Official Names for Character Sets,
http://www.iana.org/assignments/character-sets

3.3.3 Data Type

InterBase/Firebird tables are defined by the specification of columns, which accommodate appropriate information in each column using data types, for example, numerical (numeric, decimal, integer), textual (char, varchar, nchar, nvarchar), date (date, time, timestamp) or blobs.

The data type is an elemental unit when defining data, which specifies the type of data which may be stored in tables, and which operations may be performed on this data. It can also include permissible calculative operations and maximum data size.

The data type can be defined in IBExpert using the DB Explorer, by creating a domain or creating a new field in the Create Table or Table Editors.

It can of course, also be defined using SQL directly in the IBExpert SQL Editor. The syntax for the data type definition is as follows:

```
<data_type> = {
{SMALLINT | INTEGER | FLOAT | DOUBLE PRECISION}
[<array_dim>]
| {DECIMAL | NUMERIC} [(precision [, scale])]
[<array_dim]
| DATE [<array_dim>]
| {CHAR | CHARACTER | CHARACTER VARYING | VARCHAR}
 [(int)] [<array_dim>] [CHARACTER SET charname]
| {NCHAR | NATIONAL CHARACTER | NATIONAL CHAR}
 [VARYING] [(int)] [<array_dim>]
| BLOB [SUB_TYPE {int | subtype_name}) (SEGMENT SIZE int]
 [CHARACTER SET charname]
| BLOB [(seglen [, subtype])]
}
```

The InterBase/Firebird data type definitions included in this section have been kept as close as possible to original InterBase definitions to avoid any potential misunderstanding or conflict with the data types of other database programs.

Blob – Binary Large OBject

A blob is a data type storing large binary information (Binary Large OBject).

Blobs can contain any binary or ASCII information, for example, large text files, documents for data processing, CAD program files, graphics and images, videos, music files etc.

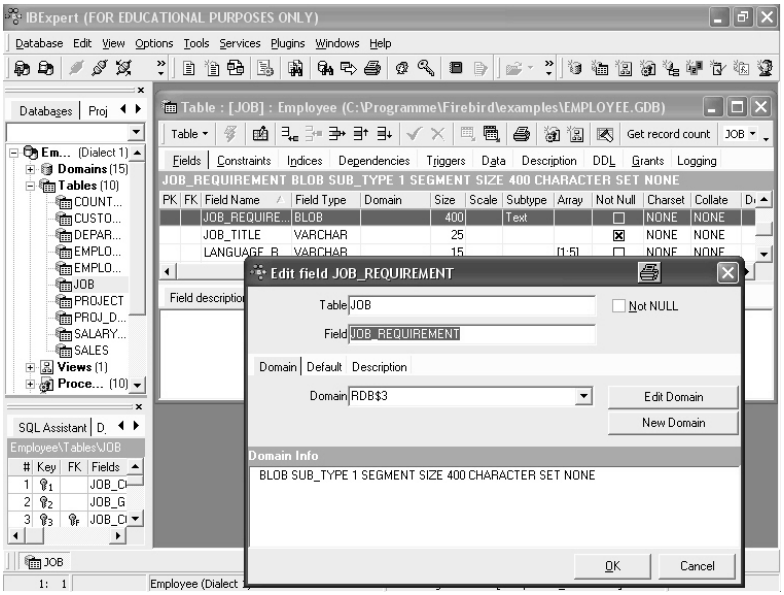
Blobs are defined as table columns. Their memory size is almost unlimited as they can be stored across several pages. This assumes however that a sufficient database page size has been specified. For example, using a 1k page, the blob may not exceed 0.5 GB, using a 4k page size, the blob size is limited to 8GB.

The ability to store such binary data in a database provides a high level of data security, data backup, version management, categorization and access control.

The advantage of blob text fields over varchar fields (e.g. `VARCHAR (32000)`) is that a network protocol transfers all 32,000 varchar characters when using an ISDN connection (analog lines compress the data to an extent). With a blob field, only the actual file size is transferred. Although - since Borland InterBase version 6.5/7 this disadvantage with varchar data type transfer has been solved, i.e. in these newer InterBase versions the full varchar length including spaces is no longer transferred each time across the network. However, even here, blobs are still more effective when working with such large data sizes.

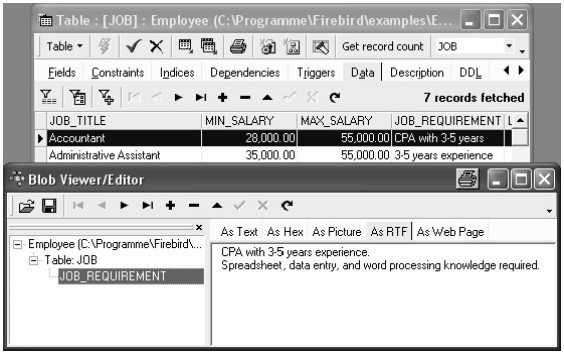
InterBase/Firebird supports quick and efficient algorithms for reading, writing and updating blobs. The user can manipulate blob processing with blob routines - also called blob filters. These filters are ideal tools for the compression and translation of blobs, depending upon the application requirements.

Blobs can be specified using the IBExpert DB Explorer or the IBExpert SQL Editor.



Blob specification includes the subtype, segment size and, if wished, the character set.

When the Data View (i.e. Data tab) in the Table Editor is selected, and the table shown contains a blob column, IBExpert can display the blob content of a selected data set as text (also as RTF), hex, images and web pages using the IBExpert menu item Tools / Blob Viewer/Editor.



It is important when using blobs in a database, to consider the database page size carefully. Blobs are created as part of a data row, but because a blob could be of unlimited length, what is actually stored with the data row is a BlobID, the data for the blob is stored separately on special blob pages elsewhere in the database.

The BlobID is an 8 byte value that allows InterBase/Firebird to uniquely identify a blob and locate it. The BlobIDs can be either temporary or permanent; a temporary blob is one which has been created, but has not yet been stored as part of a table, permanent

blobs have been stored in a table. The first 4 bytes represent the relation ID for the blob (like data rows, blobs are bound to a table), the second four bytes represent the ID of the blob within the table. For temporary blobs the relation ID part is set to 0.

A blob page stores data for a blob. For large blobs, the blob page could actually be a blob pointer page, i.e. be used to store pointers to other blob pages. For each blob that is created a blob record is defined, the blob record contains the location of the blob data, and some information about the blob's contents that will be useful to the engine when it is trying to retrieve the blob. The blob data could be stored in three slightly different ways. The storage mechanism is determined by the size of the blob, and is identified by its level number (0, 1 or 2). All blobs are initially created as level 0, but will be transformed to level 1 or 2 as their size increases.

A level 0 blob, is a blob that can fit on the same page as the blob header record, for a data page of 4096 bytes, this would be a blob of approximately 4052 bytes (page overhead - slot - blob record header).

Although the documentation states that the segment length does not affect the performance of InterBase/Firebird, the actually physical size of a blob, or its segment length can become useful in trying to improve I/O performance for the blob, especially if you can size the segment (typically) or blob to a page.

This is especially true if you plan to manipulate the blob using certain low level InterBase/Firebird blob calls. When a blob is too large to fit on a single page (level 1), and the data will be stored on one or more blob data pages, then the initial page of the blob record will hold a vector of blob page numbers.

A level 2 blob occurs when the initial page of the blob record is not big enough to contain the vector of all the blob data page numbers. Then InterBase/Firebird will create blob pointer pages, i.e. multiple vector pages that can be accessed from the initial blob header record, that now point to blob data pages. The maximum size of a level 2 blob is a product of the maximum number of pointer pages, the number of data pages per pointer page, and the space available on each data page.

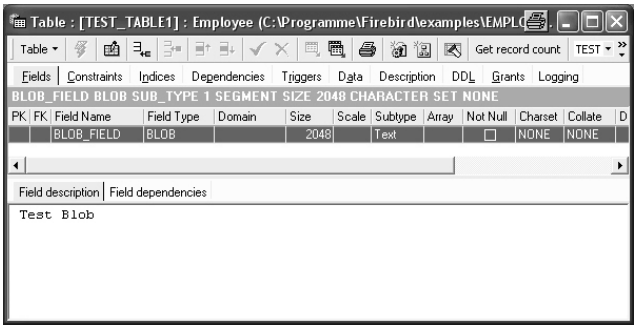
Max Blob Size:

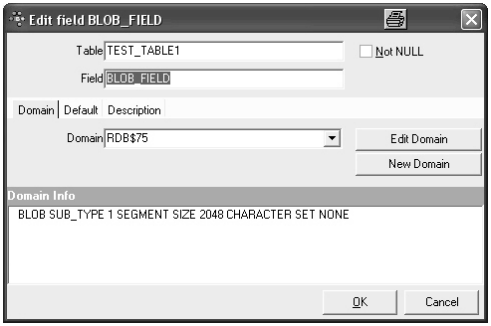
1Kb page size => 64 Mb
2Kb page size => 512 Mb
4Kb page size => 4 Gb
8Kb page size => 32 Gb
16kb page size => Big enough :-).

We would like to thank Paul Beach of IBPhoenix, for allowing us to reproduce excerpts of his session, Using and Understanding Blobs, held at the European Firebird Conference 2003.

Segment size

Segment sizes are specified for blob fields. This can be done using the Domain Editor or the Table Editor (started from the IBExpert DB Explorer or Database menu).





A blob segment size can be defined, to increase the performance when inputting and outputting blob data. This should roughly correspond to the data type size. With a memo field, for example, for brief descriptions which could however, in individual cases, be considerably longer, the segment length could be defined as 100 bytes, whereby the blob datatype is processed in 100 byte blocks.

When processing videos or large graphics in the database, a large segment length should be selected. The maximum length is 65536 bytes. This is because all blob contents are stored in blocks, and are fetched via these blocks. a typical segment size from the old days is 80 (because 80 characters fit onto one monitor line).

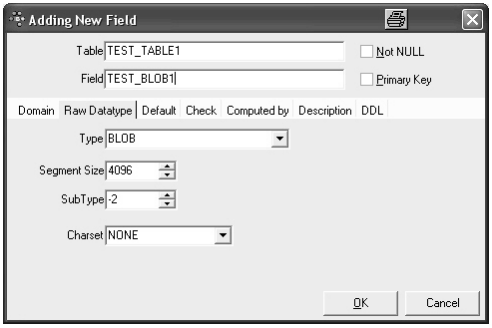
When a blob is extracted, the InterBase/Firebird server reads the number of segments that the client has requested. As the server always selects complete blocks from the database, this value can in effect be ignored on modern powerful computers. 2048 is recommended as a standard since version InterBase 6.

Subtype

Subtypes are specified for blobs. They are used to categorize the data type when defining blobs. A subtype is a positive or negative numerical value, which indicates the type of blob data. The following subtypes are predefined in InterBase/Firebird:

Subtype:	*Meaning:*
Standard blob, non-specified binary data	

1	Text blob, e.g. memo fields
Text	Alternative for defining subtype 1
Positive value	Reserved for InterBase
Negative value	User-defined blob subtypes



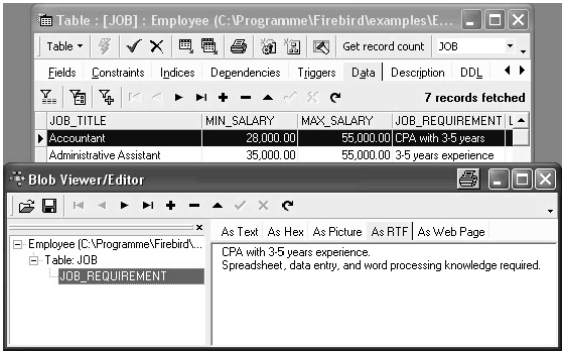
Blob fields can be specified using the Domain Editor or Table Editor (started from the IBExpert DB Explorer).

The specification of a user-defined blob subtype has no effect upon InterBase/Firebird, as the InterBase/Firebird server treats all blob fields the same, i.e. it simply stores the data and delivers it to the client program when required.

The definitions are however required by the client programs in order to display the blob content correctly. For example, `SUB_TYPE -200` could be defined as a subtype for GIF images and `SUB_TYPE -201` as a subtype for JPG images.

Subtype specification is optional; if nothing is specified, InterBase/Firebird assumes 0 = binary data.

Under the menu item Tools, the IBExpert Blob Viewer/Editor can display blob contents as text, hex, images, RTF and web pages.



CHAR and VARCHAR

InterBase/Firebird provides two basic data types to store text or character information: char and varchar (blobs also allow character storage using the subtype text).

char and varchar are data types which can store any text information. Numbers that are not calculated, such as zip codes, are traditionally stored in char or varchar columns. The length is defined as a parameter, and can be between 1 and 32,767 bytes. It is particularly useful for codes that typically have a fixed or predefined length, such a the zip code for a single country.

Compared to most other databases, InterBase/Firebird only stores significant data. If a column is defined as char(100), but only contains entries with 10 characters, the additionally defined bytes are not used, as InterBase/Firebird stores char and varchar types similarly, and does not fill unused spaces with blanks. Both char and varchar are stored in memory buffer in their full, declared length; but the whole row is compressed prior to storing i.e. chars, varchars, integers, dates, etc. all together.

Indeed, varchar columns require more storage than char columns, because when storing a varchar, InterBase/Firebird adds two bytes that state just how big the varchar actually is.

So a char will in fact be stored in a smaller space. However, when a SELECT is performed on a varchar column, InterBase/Firebird strips the 2 byte padding and returns the stored value. When a SELECT is performed on a char column, InterBase/Firebird returns the value and the "empty spaces". Thus the two bytes saved in storage of a char must be balanced against the subsequent need to strip the spaces on the client side. These two bytes however are, with today's hardware, too negligible to have an influence upon the database performance. This can however be disadvantageous when defining short text fields.

In practical terms consider just this one rule: only use chars if strings of few characters are to be stored; the exception to the rule being when working with intermediate tables that are required to export data to fixed length prn files. Then the fixed length field will be a positive advantage.

This efficient storage in InterBase/Firebird can lead to considerable confusion particularly when importing data, as Paradox or dBASE databases save all blank spaces, and after importing a 10MB dBASE file into InterBase, often only 3-6 MB remain, although all data sets were imported correctly.

For this reason columns can be defined generously in InterBase/Firebird without a problem, whereas in other databases each defined byte influences the size of the database, regardless of whether data is stored in these fields or not.

Please note however that indexed char fields should not be more than approx. 80 characters in length (with Firebird 1.5 the limit is somewhat higher).

The char data type definition can be written in two ways:

```
CHAR
CHARACTER
```

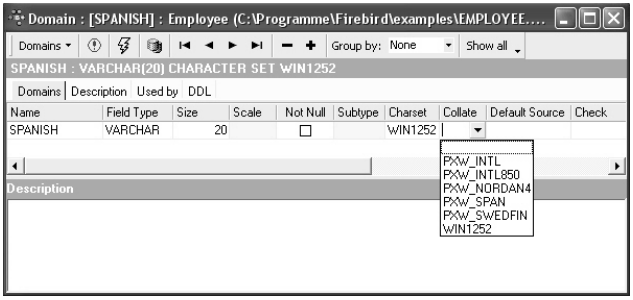
The varchar data type definition can be written as follows:

```
VARCHAR
CHARACTER VARYING
CHAR VARYING
```

Collate

A special collation sequence can be specified for char and varchar field columns. The collate parameter allows fields to be collated according to a certain language/group of languages e.g. collate according to the German language when using Win1252.

In IBExpert the collation sequence can be specified when defining the character set for a domain or field:



The collation options are offered in IBExpert in a pull-down list, after specifying the character set.

In DDL it is specified using the keyword COLLATE and the respective character set table, for example:

```
CREATE DOMAIN dom_city VARCHAR(20)
COLLATE PXW_INTL850;

CREATE DOMAIN User_Name VARCHAR(20)
CHARACTER SET DOS437
DEFAULT USER
NOT NULL
COLLATE PDOX_ASCII
```

The parameter sequence is important, as the collation sequence must be specified last.

NCHAR and NVARCHAR

NCHAR or NATIONALCHARACTER
NVARCHAR or NATIONAL CHAR VARYING or NATIONAL CHARACTER VARYING

nchar/nvarchar are data types, which can be defined as the char/varchar data types with a length of 1-32,767 bytes. The only difference to the char/varchar data type is

that nchar/nvarchar automatically defines a special character set for this table column: "CHARACTER SET ISO8859_1"

INTEGER and SMALL INTEGER (Int and SmallInt)

Integer data types are used to store whole numbers. SmallInt is the abbreviation for small integer. Values following the decimal point are not allowed. Depending upon the numeric area required, following integer types are supported:

Type	*Size*	*Value range*
SmallInt	2 bytes	-32,768 to +32,767
Integer	4 bytes	-2,147,483,648 to +2,147,483,647

4 bytes of data storage are required for the integer value, whereby 31 bits are for the number and 1 bit for the sign. 2 bytes of data storage are required for the small integer value, whereby 15 bits are for the number and 1 bit for the sign. It is usually preferable to use an integer data type as 2 bytes more or less are fairly irrelevant these days.

An integer is a 15-digit number and although extremely large, is by far not as large as the numeric(18).

Integer types are particularly suited for unique identification numbers, as InterBase/Firebird contains mechanisms for the automatic generation of whole number values (please refer to generator for further information). The resulting indices for the connection of multiple tables to each other are relatively small and offer extremely quick access, as the highest computer performance on all computer platforms is generally found in integer operations. It is possible to specify the display format of an integer under Environment Options / Grid / Display Formats.

SmallInts can also be used for Boolean data types e.g. true/false, male/female.

FLOAT and DOUBLE PRECISION

Float data types are used to store values with significant decimals. The following float types are supported:

Type	*Size*	*Value range*
Float	4 bytes	7 significant decimals; -3.4 x 10^-38 to 3.4 x 10^38
Double Precision	8 bytes	15 significant decimals; -1.7 x 10^-308 to 1.7 x 10^308

A column with the defined data type FLOAT can store a single-precision figure with up to 7 significant decimals. The decimal point can float between all seven of these digits. If a number with more than 7 decimal places needs to be saved, decimals beyond the seventh position are truncated. Float columns require 4 bytes of storage.

A column with the defined data type DOUBLE PRECISION can store numbers with 15 significant decimals. This uses 8 bytes of storage. As with the float column, the decimal point can float within the column. The double precision datatype is implemented in the majority of InterBase platforms as a 64 bit number.

Float types can be implemented for any calculative operations. They offer an optimal performance and sufficient range of values. It is possible to specify the display format of a float field under Environment Options / Grid / Display Formats.

The DOUBLE PRECISION data type can be written as follows:

```
DOUBLE PRECISION
DOUBLE
```

The main advantage of a DOUBLE PRECISION data type is the large number of decimal places e.g. 1/3 in DOUBLE PRECISION would be 0,33333333333333 in NUMERIC(18,4) it would be 0,3333. Please note: up until dialect 1 numeric and double precision were identical i.e. an SQL with the data type NUMERIC(15,2) results in the following:

Result with dialect 1:

```
CREATE TABLE TEST(WERT NUMERIC(15,2));
INSERT INTO TEST(WERT) VALUES(100);
SELECT * FROM TEST;  result 100
UPDATE TEST SET WERT=WERT/3;
SELECT * FROM TEST;  result 33,33
UPDATE TEST SET WERT=WERT*3;
SELECT * FROM TEST;  result 100
```

Result with dialect 3:

```
CREATE TABLE TEST(WERT NUMERIC(15,2));
INSERT INTO TEST(WERT) VALUES(100);
SELECT * FROM TEST;  result 100
UPDATE TEST SET WERT=WERT/3;
SELECT * FROM TEST;  result 33,33
UPDATE TEST SET WERT=WERT*3;
SELECT * FROM TEST;  result 99,99
```

Since dialect 3 NUMERIC data is rounded according to commercial rounding rules; up to dialect 1 NUMERIC data is rounded according to technical rounding rules.

NUMERIC and DECIMAL

The NUMERIC data type specifies a numeric column where the value has a fixed decimal point, such as for currency data. NUMERIC(18) is a 64 bit integer value in SQL dialect 3 and is almost infinite. Since SQL dialect 3 numeric and decimal data types are stored as integers of the respective size.

SQL dialect 1 offers NUMERIC(15).

Syntax:

```
NUMERIC(precision, scale);
```

or

```
DECIMAL(precision, scale);
```

PRECISION refers to the total number of digits, and SCALE refers to the number of digits to the right of the decimal point. Both numbers can be from 1 to 18 (SQL dialect 1: 1-15), but scale must be less than or equal to precision.

It is better to define NUMERIC always at its maximum length, as in this case, the 32 bit integer value is used. Otherwise a 16 bit value is used internally, for example with NUMERIC(4,2), and this is not always transformed back correctly by the client program environments (an older BDE version could, for example, transform Euro 12,40 with NUMERIC(4,2) into Euro 1240).

InterBase/Firebird supports a number of options for specifying or not specifying precision and scale:

1. If neither precision nor scale are specified, InterBase/Firebird defines the column as INTEGER instead of NUMERIC and stores only the integer portion of the value.

2. If just precision is specified, InterBase/Firebird converts the column to a SMALLINT, INTEGER or DOUBLE PRECISION data type, based on the number of significant digits being stored.

The NUMERIC data type should only be used for fields that are later to be used as part of a calculation.

InterBase/Firebird converts the columns as follows:

Definition	*Datatype Created*
Decimal(1)-Decimal(4)	Small Integer
Decimal(5)-Decimal(9)	Integer
Decimal(10)-Decimal(18)	Int (64)

Note that if a DECIMAL(5)data type is specified, it is actually possible to store a value as high as a DECIMAL(9) because InterBase/Firebird uses the smallest available data type to hold the value. For a DECIMAL(5) column, this is an INTEGER, which can hold a value as high as a DECIMAL(9).

DATE

The date data type stores values which represent a date. InterBase/Firebird supports a single date-type column that requires 8 bytes of storage space. It uses 4 bytes for the date and 4 bytes for the time. Valid dates are from January 1, 100 AD through February 28, 32,767 AD. Note: for DATE arithmetic purposes, DATE 0 (the integer value of zero) as a DATE in InterBase/Firebird is November 17, 1898.

Different date formats are supported. There are however slight differences between SQL dialect 1 and SQL dialect 3.

SQL dialect 1: DATE also includes a time slice (equivalent to TIMESTAMP in dialect 3).
SQL dialect 3: DATE does not include any time slice.

Using SQL dialect 1 the default 'NOW' for data type date means current time and date of the server; there is also 'TODAY' (only date; the time is always set at midnight, 'YESTERDAY', 'TOMORROW').

Example:

```
SELECT CAST ('NOW' AS DATE) FROM RDB$DATABASE
```

SELECT CAST is an SQL dialect 1 command (although it also functions in SQL dialect 3); SELECT is used in SQL dialect 3. These values are primarily compatible to older InterBase versions. When working with SQL dialect 3, the CURRENT_ constants (see below) should be used as far as possible.

From InterBase 6 upwards and Firebird there are the following for dialect 3: CURRENT_TIME, CURRENT_TIMESTAMP, CURRENT_DATE (without quotation marks and without CAST). Example:

```
SELECT CURRENT_DATE-1 FROM RDB$DATABASE
```
Result: the date yesterday, etc.

```
SELECT CURRENT_TIMESTAMP-(1/24) FROM RDB$DATABASE
```
Result: the current time minus one hour (one twenty-fourth of a day).

It is possible to specify the display format of a date field under Environment Options / Grid / Display Formats. For the various options available, please refer to Date Time Format.

TIME

The TIME data type is new to InterBase v 6.0. It is an SQL dialect 3 data type. TIME is a 32-bit field type of TIME values. The range is from 0:00 AM to 23:59:9999 PM.

It is possible to specify the display format of a date field under Environment Options / Grid / Display Formats. For the various options available, please refer to Date Time Format.

TIMESTAMP

TIMESTAMP is new to InterBase v 6.0. It is an SQL dialect 3 data type. TIMESTAMP is a 64-bit field type comprised of both date and time. The range is from January 1,100 AD to February 28, 32768 AD.

It is the equivalent of DATE in SQL dialect 1.

It is possible to specify the display format of a date field under Environment Options / Grid / Display Formats. For the various options available, please refer to Date Time Format.

3.3.4 Array

InterBase/Firebird allows a column to be defined as an array of elements, i.e. data information can be stored in so-called arrays. An array is a range of values determined by setting a lower and an upper limit. An array consists of any amount of information that can be split into different dimensions. The array can be managed as a whole, as a series of elements in one dimension of the array, or as individual elements.

Arrays should be used with caution. Database normalization usually supplies an alternative format for storing such data, so that normal table structures are just as suitable, and also preferable. There are however occasionally exceptions, for example for measurement value logging, when arrays are the preferred option.

The array data type is used relatively seldom, as it is not very simple to process, and does not really conform to the typical demands of an SQL database (usually one or more detail tables would be created, and not an array).

Arrays can be declared as a domain or directly in the table definition following the data type definition. Array data can be of any type except blob. Between 1 and 16 dimensions can be specified; each dimension can store as many elements as can be fitted into the database. The values are stored as a blob and are therefore almost unlimited in scope.

The only difference compared to the normal data type definition is the specification of the dimensions in square brackets, each dimension being separated by commas. By default, the lower bounds ID number is 1 and the upper bounds ID number is the maximum of that dimension. Alternate bounds IDs can be specified in place of the array size by separating them with a colon. For example, an array with 5 measurements with 2 dimensions starting at the default value 1 is defined as follows:

```
[2,5]
```

Counting begins at 1 and ends at the value entered by the user. In this case 2x5=10 measurements can be logged. If counting is to begin at, for example, 0, the array definition is as follows:

```
[0:2, 0:5]
```

One-dimensional Arrays:

Definition `NAME DATATYPE [LOWER_DIMENSION:UPPER_DIMENSION].`
Example: `LANGUAGE_REQ VARCHAR(15) [1:5]`
In this field 5 data entries of the VARCHAR(15) type can be stored.
LANGUAGE_REQ[1] up to LANGUAGE_REQ[5] can be accessed.

Multi-dimensional Arrays:

Definition NAME DATATYPE [LOWER_DIMENSION1:UPPER_DIMENSION1] [LOWER_DIMENSION2:UPPER_DIMENSION2].
Example: DAILY_MEASUREMENTS NUMERIC(18,2) [1:24][1:365].

When using arrays, it is important to be aware of the advantages and limitations.

Advantages of arrays:

1. InterBase operations can be performed upon the total data type as a single element. Alternatively operations can be executed on part of an array only for certain values of a dimension. An array can also be broken down into each single element.

2. Following operations are supported:
- SELECT statement from array data
- Insertion of data in an array
- Updating data in an array slice
- Selecting data from an array slice
- Examination of an array element in a SELECT statement

Array limitations:

1. A user-defined function can only access one element in an array.

2. The following operations are not supported:
- Dynamically referencing array dimensions using SQL statements
- Inserting data into an array slice
- Setting individual array elements to null
- Using aggregate functions such an MIN(), MAX(), SUM(), AVG() and COUNT() on arrays
- Referencing an array in the GROUP BY clause in a SELECT query
- Creating a view, which selects from array slices

3. The data stored in this way cannot be selected per index; each query always accesses the fields unindexed.

3.3.5 Boolean

InterBase/Firebird does not offer a native Boolean data type. However, they can be implemented using domains.

The first step is to define a domain (which should logically be named Boolean). The domain can be defined in one of two ways:

1. Using a SMALLINT (16 bits), defaulting to zero, with a check constraint to ensure only the values of zero or one are entered. i.e:

```
CREATE DOMAIN D_BOOLEAN AS SMALLINT DEFAULT 0
CHECK (VALUE BETWEEN 0 AND 1);
```

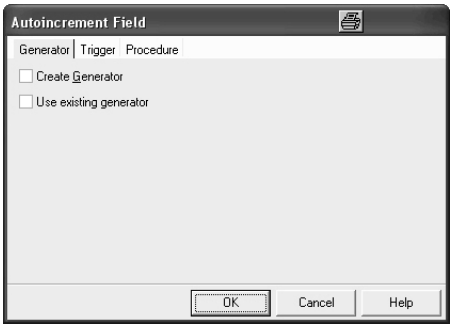
Once you have defined this domain you can forever use it as a Boolean data type without further concern. It is particularly suitable from a Delphi point of view, as Pascal Booleans work in a similar manner.

2. Alternatively, the domain can be defined as a CHAR(1) and appropriate single character values ensured using a check constraint. If 'T' and 'F' or 'Y' and 'N' are more meaningful for your application then use this approach.

We'd like to thank *Paul Beach of IBPhoenix* for this article.

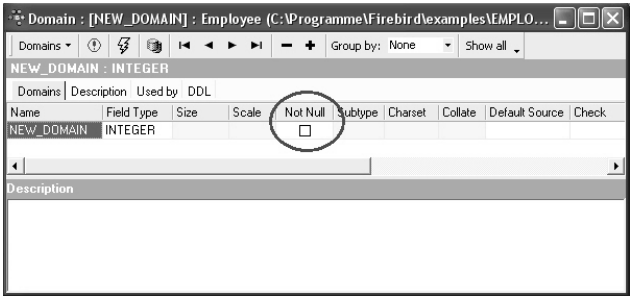
3.3.6 Autoincrement

An autoincrement is an automatic counter/calculator, such as a generator, trigger or stored procedure.



3.3.7 Not Null

NOT NULL is a parameter that does not allow a column field to be left blank. It can be defined for a field or a domain.

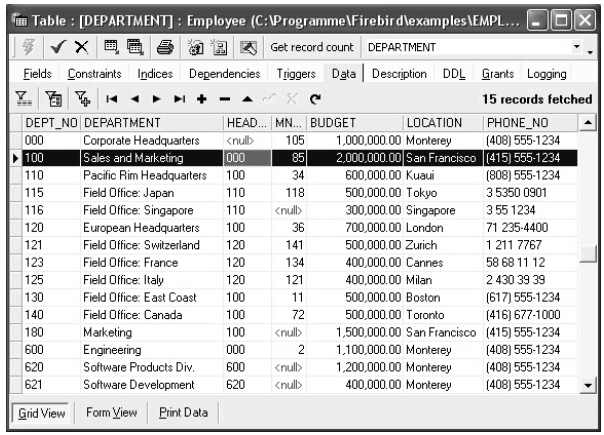


It forces a value to be entered into the column. It operates in the same way for tables as for domains. The parameter DEFAULT NULL and NOT NULL cannot be used in the same column definition. The NOT NULL parameter must be specified if the column is to be defined as PRIMARY KEY or UNIQUE.

3.3.8 Null

Null is the term used to describe a data field without a value, i.e. the field has been left blank because the information is either not known or not relevant for this record/data set. The null value can be stored in text, numeric and date data types.

A relational database is able to store null values as data content. A null value does not mean numerical zero. For example, a product can have zero sales () or unknown sales (<null>).



DEPT_NO	DEPARTMENT	HEAD...	MN...	BUDGET	LOCATION	PHONE_NO
000	Corporate Headquarters	<null>	105	1,000,000.00	Monterey	(408) 555-1234
100	Sales and Marketing	000	85	2,000,000.00	San Francisco	(415) 555-1234
110	Pacific Rim Headquarters	100	34	600,000.00	Kuaui	(808) 555-1234
115	Field Office: Japan	110	118	500,000.00	Tokyo	3 5350 0901
116	Field Office: Singapore	110	<null>	300,000.00	Singapore	3 55 1234
120	European Headquarters	100	36	700,000.00	London	71 235-4400
121	Field Office: Switzerland	120	141	500,000.00	Zurich	1 211 7767
123	Field Office: France	120	134	400,000.00	Cannes	58 68 11 12
125	Field Office: Italy	120	121	400,000.00	Milan	2 430 39 39
130	Field Office: East Coast	100	11	500,000.00	Boston	(617) 555-1234
140	Field Office: Canada	100	72	500,000.00	Toronto	(416) 677-1000
180	Marketing	100	<null>	1,500,000.00	San Francisco	(415) 555-1234
600	Engineering	000	2	1,100,000.00	Monterey	(408) 555-1234
620	Software Products Div.	600	<null>	1,200,000.00	Monterey	(408) 555-1234
621	Software Development	620	<null>	400,000.00	Monterey	(408) 555-1234

A null value can occur for the following reasons:

- The value is not yet known, but will be added at a future date.
- The value is not yet available for some reason, e.g. the date of receipt of payment.
- The value is not important, e.g. the credit card expiry date of someone who has paid cash.

InterBase/Firebird does not use a special byte sequence to indicate a null, but administrates this information internally. Null values can influence query contents considerably, for example, when a column average is calculated. The values filled by the null value, i.e. empty fields, are not taken into consideration. A field containing the value 0 is included in the calculation of the average.

Examples from the Firebird 1.5 Quick Start Guide:

- `1 + 2 + 3 + NULL = NULL`
- `not (NULL) = NULL`
- `'Home ' || 'sweet ' || NULL = NULL`
-
```
if (a = b) then
    MyVariable = 'Equal';
else
    MyVariable = 'Not equal';
```

After executing this code, `MyVariable` will be `'Not equal'` if both `a` and `b` are `NULL`. The reason is that the expression `'a = b'` yields `NULL` if at least one of them is `NULL`. In an "`if...then`" context, `NULL` behaves like `FALSE`. So the `'then'` block is skipped, and the `'else'` block executed.

-
```
if (a <> b) then

  MyVariable = 'Not equal';
```

```
else
  MyVariable = 'Equal';
```

Here, `MyVariable` will be `'Equal'` if a is `NULL` and `b` isn't, or vice versa. The explanation is
analogous to that of the previous example.

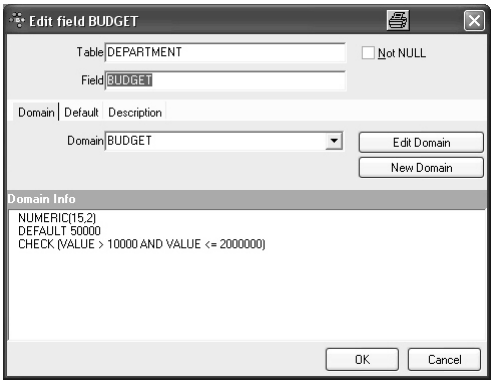
- `FirstName || ' ' || LastName`
  will return `NULL` if either `FirstName` or `LastName` is `NULL`.

Think of `NULL` as `UNKNOWN` and all these strange results suddenly start to make sense! If the value of `Number` is unknown, the outcome of `'1 + 2 + 3 + Number'` is also unknown (and therefore `NULL`). If the content of `MyString` is unknown, then so is `'MyString || YourString'` (even if `YourString` is non-`NULL`). Etcetera.

3.3.9 Alter Field

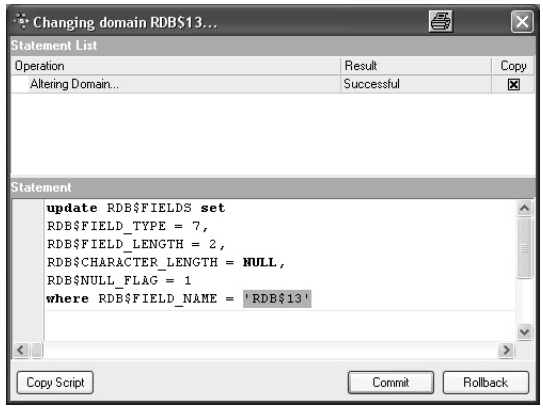
Similar to Alter Domain, only certain field attributes may be altered. For example, CHECK instructions and default values may be added, altered or deleted. However it is not possible to alter the basic data type (for example, from numeric to varchar). Neither is it possible to drop a NOT NULL constraint. To alter these the field has to be dropped and recreated (see Drop Field).

Fields can be altered in the Table Editor by double-clicking on the selected field, or right-clicking and selecting Edit Field from the menu, or pressing the [Enter] key, to open the Field Editor:



However you will notice that you need to switch to the Domain Editor to perform any actual changes, as even if the field is not based on a user-defined domain, InterBase/Firebird automatically creates a system domain for all field definitions.

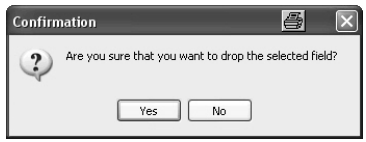
The desired alterations can however be easily made to the user-defined or system domain and executed and checked before finally committing.



3.3.10 Drop Field/Delete Field

Fields can be dropped directly in the Table Editor on the Fields page, by using the '-' icon in the Table Editor toolbar, selecting from the right-click menu or using the key combination [Shift + Del].

IBExpert asks for confirmation:



before finally dropping the field. Once dropped, it cannot be retrieved.

When dropping fields, it is important to note that the field may not be part of the table's primary key, have a foreign key relationship with another table, contain a unique constraint, be part of a table constraint or part of another column's CHECK constraint.

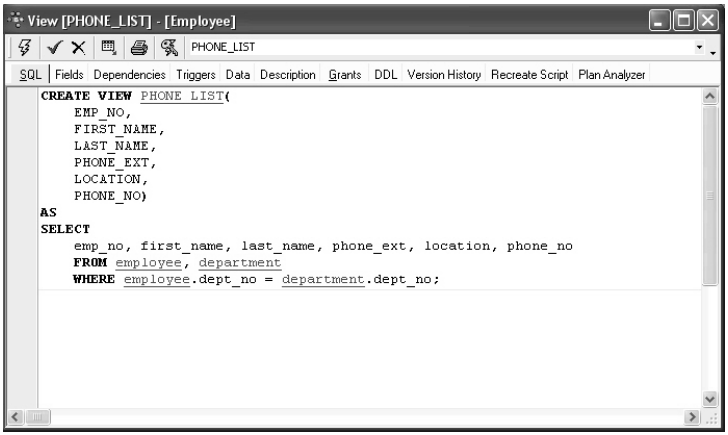
The Constraints page in the Table Editor lists all such fields, so that the developer can quickly ascertain whether constraint alterations/deletions are necessary, before dropping the field in question (or whether, in fact, the field should be dropped at all!).

Using SQL the syntax is:

```
ALTER TABLE <table_name>
DROP <field_name>;
```

3.4 View

A view is a stored SELECT of one or more tables. The rows to be returned are defined by the SELECT statement that lists columns from the source tables. Only the view definition is stored in the database, it does not directly represent physically stored data. The WHERE command can also be used. A view has no input parameters.



It can be likened to a virtual table. The view can be treated, in almost all respects, as if it were a table, using it as the basis for queries and even updates in some cases. It is possible to perform SELECT, PROJECT, JOIN and UNION operations on views as if they were tables.

Views give end users a personalized version of the underlying tables in the database and also simplify data access, by protecting them from the details of how information is spread across multiple tables. They also provide security by hiding certain columns in the table(s) from various users. InterBase/Firebird allows user rights to be granted to the view and not the underlying table(s).

Advantage of views (and stored procedures): as these are part of InterBase or Firebird, it is irrelevant which front end is subsequently used, be it Delphi, PHP or other.

They allow the developer to denormalize data, combining information from two or more tables into a single virtual table. Instead of creating an actual table with duplicate data, a view can be created using SELECT, JOIN and WHERE.

Views cannot be sorted, they merely display the result of a specified SELECT. (A view can therefore be compared to a saved query). The ORDER BY instruction cannot be used in a view (the data sets are displayed as determined by the optimizer, which is not always intelligent!). In such a case, a stored procedure would have to be used (stored procedures being more flexible in any case, and offering more control).

Views can be used, for example, for internal telephone lists, or when information from more than one table needs to be linked, e.g. the first modular result needs to be linked to the second result.

The underlying SELECT definition can contain all the performance features of a select query on tables, it is however subject to the following restrictions:

1. All columns must be explicitly specified, so that the view always returns the same columns in the correct order.

2. If reference is made to a SELECT * statement in a view, the result is returned in the column sequence of the definition of the underlying tables, and can therefore deliver different results should changes later be made to the table structure.

3. No ORDER BY statements may be used.

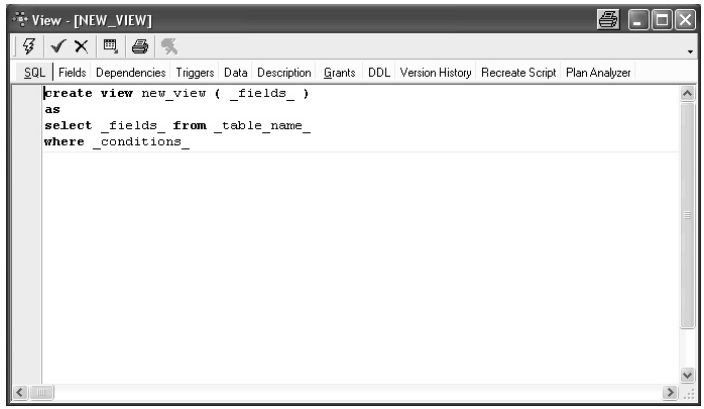
4. Indices can only be placed on the columns of the base tables, not the view columns. When the view is generated, these indices are automatically used.

Views allow a data modularization, particularly useful with complex data quantities, as another view can be incorporated in the view definition.

3.4.1 New View / View Editor

A new view can be created in a connected database, either by using the menu item Database / New View, , the respective icon in the New Database Object toolbar, or using the DB Explorer right mouse button (or key combination [Ctrl + N]), when the view heading of the relevant connected database is highlighted. Alternatively, a new view can be created directly in the IBExpert SQL Editor, and then saved as a view.

A New View dialog appears, with its own toolbar:



The view can be created directly in the SQL dialog, and subsequently committed using the respective icon or [Ctrl + F9].

Please refer to the following subjects for details regarding the individual editor pages.

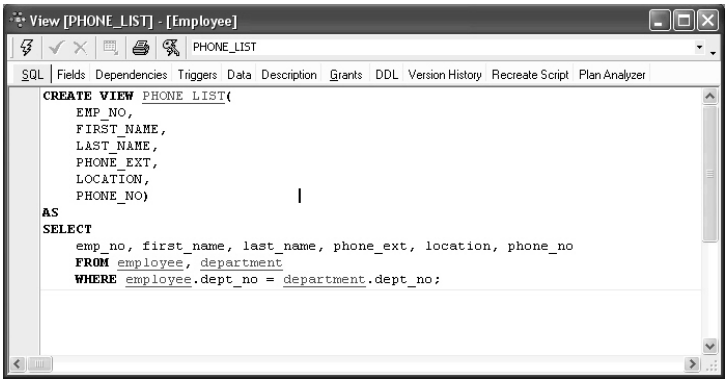
SQL

When creating a view it is necessary to define a view name that is unique in the database.
All data manipulation operations such as SELECT, INSERT, UPDATE and DELETE are carried out using this name.

The view can then be created in the SQL dialog using the following syntax:

```
CREATE VIEW ViewName (<List_of_field_names>)
AS
SELECT <fields_ from _table_name>
[WITH CHECK OPTION];
```

An example can be viewed in the InterBase/Firebird sample EMPLOYEE database:



The view name must be unique. As InterBase/Firebird only stores the view definition (i.e. it does not copy the data from the tables into the view), views depend a lot upon indices set in the base table, in order to locate data rapidly from the original tables. It is therefore important to analyze views carefully, and place indices on those columns that are used to join tables and to restrict rows.

The tables and fields can be easily inserted into the SQL script by dragging the relevant tables and fields from the DB Explorer and SQL Assistant, and dropping them in the respective position in the SQL dialog in the New View Editor. After naming the view fields and inserting the relevant base table fields, the new view can be committed using the respective icon or [Ctrl + F9].

The view contents result from the returns of the `SELECT` statement that corresponds, with few exceptions, to the SQL `SELECT` command. The `SELECT` statement specifies which tables, columns and rows are to be returned as part of the view.

If the view is an updateable view, the optional `WITH CHECK OPTION` parameter may also be used to control data input.

The field names, as they are to appear in the view, can be optionally specified under a different name to the field names in the base tables. If no specification is made, the original base table column names automatically become the view field names. If column names are specified, they must be unique within the view and a name must be specified for every column returned by the view (even if some of the view field names correspond to the original field names). Please note that if the `SELECT` statement includes derived columns, column names must be specified.

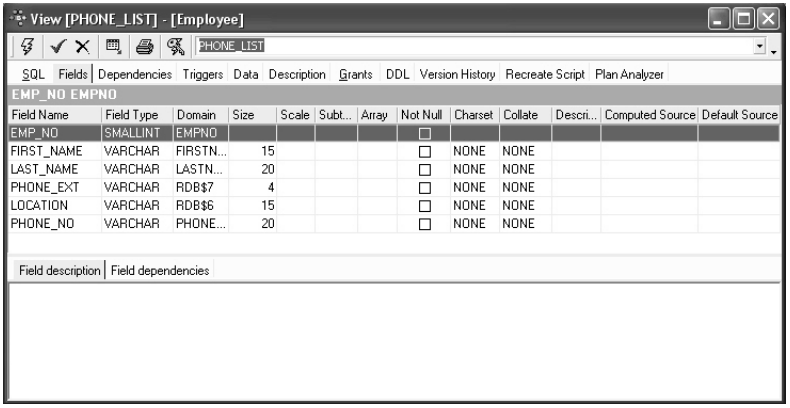
If the view is to be used as part of a query, or indeed any other SQL statement, InterBase/Firebird queries the original data directly. This important feature offers the flexibility of being able to make alterations to the underlying database structure without af-

fecting the user's view of the data or the view of any programs, which reference the view instead of the base tables.

Finally compile the new view using the respective toolbar icon or [F9], and, if desired, autograt privileges, again using the respective toolbar icon or key combination [Ctrl + F8].

Fields

The Fields page displays the fields selected from the base table (with their new view names, if they have been specified), along with their properties.

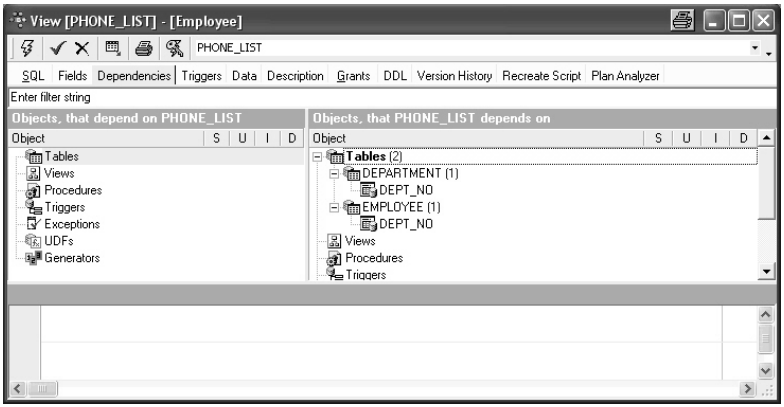


The individual fields may not be edited directly from this dialog; to alter fields, please refer to the Table Editor / Fields.

These fields can however be sorted here into ascending or descending order based upon the column where the mouse is, by clicking on the column headers (i.e. Field Name etc.).

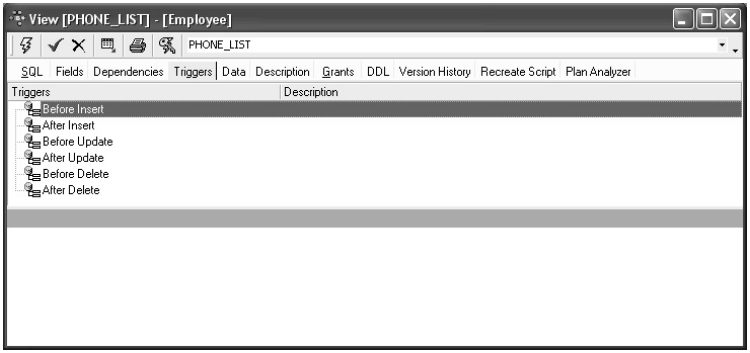
By double-clicking on the right edge of the column header, the column width can be adjusted to the ideal width.

Dependencies



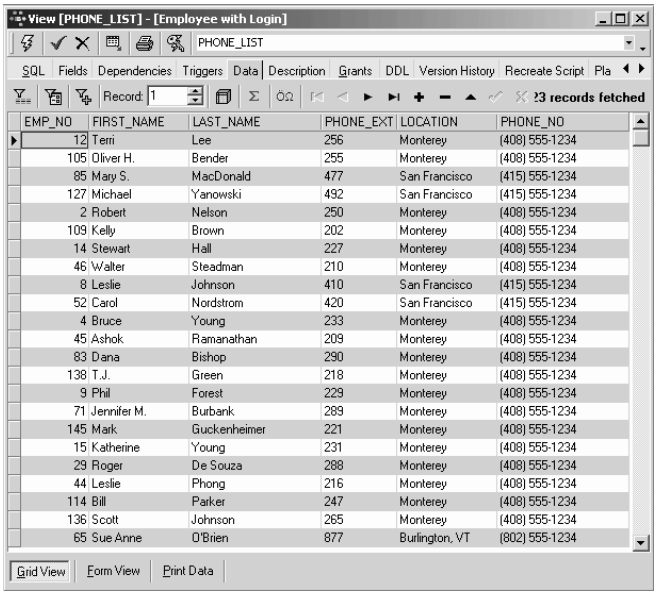
Please refer to Table Editor / Dependencies.

Triggers



Please refer to Table Editor / Triggers.

Data

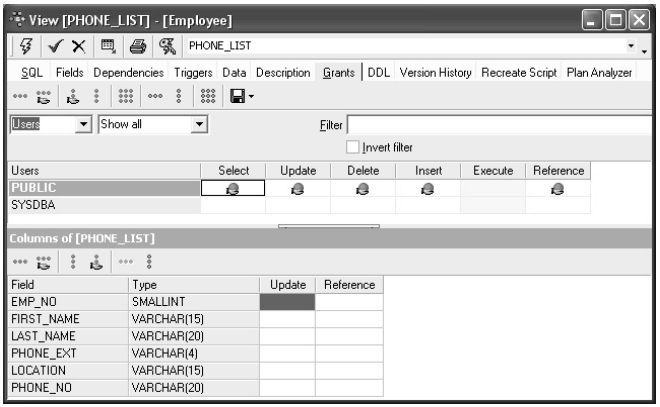


Please refer to Table Editor / Data. Please note that data may only be manipulated in this dialog if the view is defined as, and meets all conditions required by an updatable view.

Description

Please refer to Table Editor / Description.

Grants

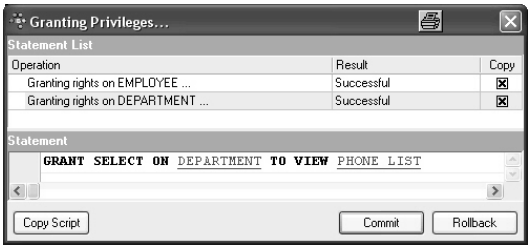


Please refer to Table Editor / Grants and Autogrant Privileges.

Autogrant Privileges

The Autogrant Privileges icon can be found in the View Editor, Procedure Editor and Trigger Editor toolbars. Privileges can also be autogranted using the key combination [Ctrl + F8]. It allows all privileges to be automatically granted for views, procedures and triggers.

(This feature is unfortunately not included in the Personal Edition.)

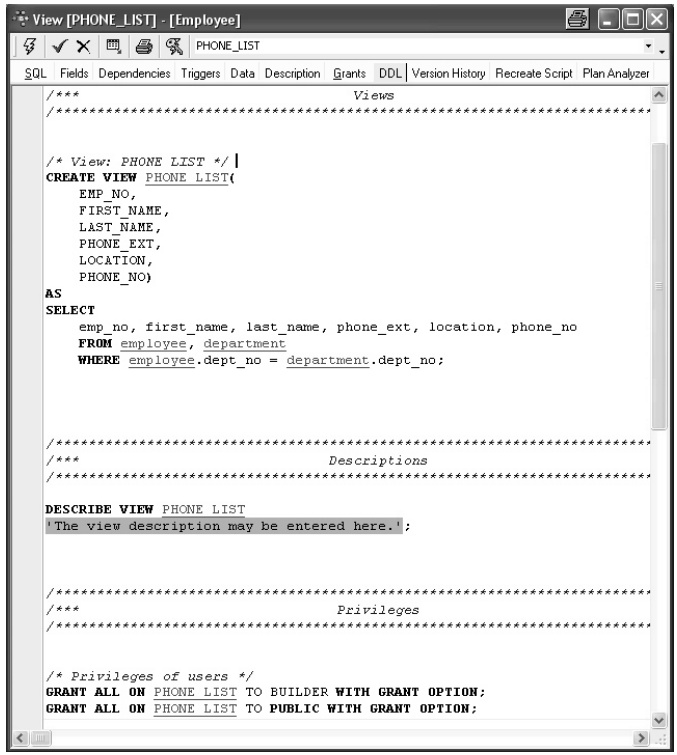


This assigns all rights for newly created objects for all users, and helps to prevent the frequent problem that developers often initially create multitudes of objects for their new database, and suddenly realize that they have not assigned any rights for these views, triggers or procedures.

For those preferring to limit the assignment of rights, please use the Grants page, offered in the majority of object editors, or the IBExpert Tools / Grant Manager.

Under the IBExpert Option menu item, Environment Options / Tools the default option, *Autogrant privileges when compiling procedures, triggers and views*, needs to be checked, for this function to work. Since IBExpert version 2005.02.12.1 it is also possible to specify here whether existing privileges should first be deleted, before new ones are granted.

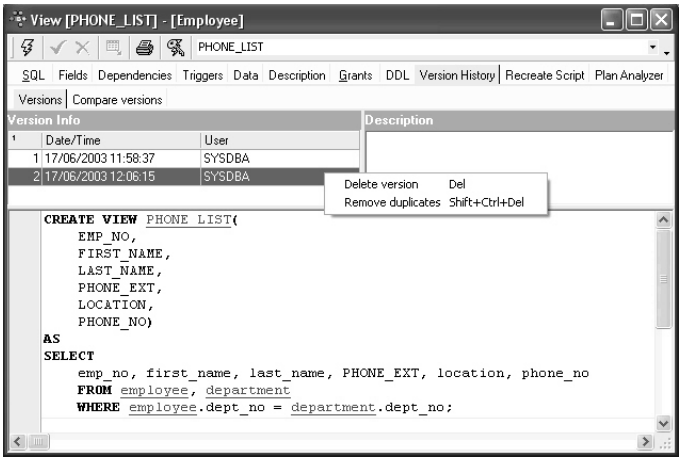
DDL



Please refer to Table Editor / DDL.

Version History

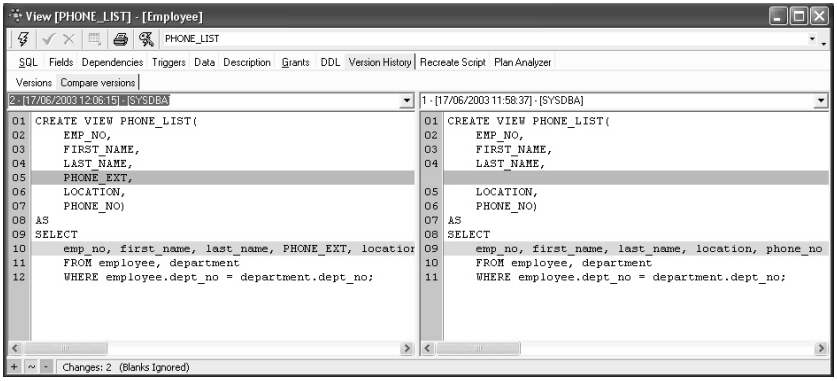
The Version History page offers a unique and automatic documentation. It is available in the View Editor, Procedure Editor and Trigger Editor. It displays different versions of the view, procedure or trigger (if existent), and lists the dates when changes were made, along with the person(s) responsible.



The first time the Version History is opened, IBExpert asks for confirmation, as it needs to create certain system tables for the version history logging. This only needs to be confirmed once. After this the Version History appears immediately in all relevant editors, and all object changes are automatically stored.

Versions listed in the Version Info panel can be marked, and deleted using the right mouse click menu (key combinations: Delete version [Del]; Remove duplicates [Shift + Ctrl + Del]).

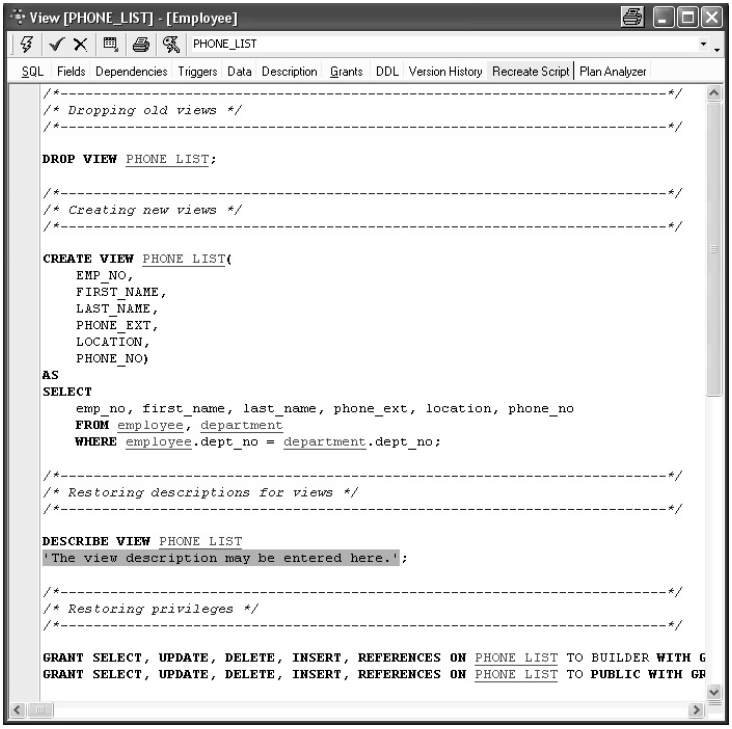
The SQL scripts of the different versions can even be compared, under the Compare Versions tab.



The pull-down list at the top of the two script panels, allows different versions to be selected, without having to switch back to the Versions page. Alterations are highlighted by colored bars, marking the line where an alteration has been made. The color code key can be viewed in the dialog's status bar, along with a note of the number of changes made between the two versions.

Recreate Script

The Recreate Script page displays the full SQL script for the view, beginning with the `DROP VIEW` command, and then recreating the current view. This is useful should errors arise in a view where it is almost impossible, due to the complexity of the view or the multitude of different versions, to detect the source.



The script can even be edited directly in this dialog, and the changes committed. The right-click menu is the same as that in the SQL Editor, allowing a number of further operations directly on the SQL script (please refer to SQL Editor Menu).

Plan Analyzer

Please refer to SQL Editor / Plan Analyzer. Please note that the performance information is not available here in the View Editor's Plan Analyzer.

Updatable views and read–only views

The simplest and quickest way to create an updateable view is to use the Create View from Table option in the IBExpert Table Editor, and create a trigger (checkbox options to create BEFORE INSERT, BEFORE UPDATE or BEFORE DELETE). Complete the trigger text in the lower code editor window (taking into consideration the notes below), and the updateable view is complete!

If the view is to be an updatable view, the optional parameter `WITH CHECK OPTIONS` needs to be used to control data input. If this parameter is used, only those values corresponding to the view's `SELECT` statement may be input. A view needs to meet all of the following conditions if it is to be used to update data in the base table:

- The view is based on a single table or on another updatable view. Joined tables result in a read-only view. (The same is true if a subquery is used in the `SELECT` statement.)
- Any columns in the base table that are not part of the view allow nulls. This condition requires that the base table's primary key be included in the view.
- The `SELECT` statement does not include a `DISTINCT` operator. This restriction might have the effect of removing duplicate rows, making it impossible for InterBase/Firebird to determine which row to update.
- The `SELECT` statement does not include aggregate functions or the `GROUP BY` or `HAVING` operators.
- The `SELECT` statement does not include stored procedures or user-defined functions.

In a normalized database, a view is usually updatable if it is based on a single table and if the primary key column or columns are included in the view definition.

However it is possible to input data into a view and then allocate the new data / data changes to several individual tables by using triggers.

Specifying a view with the CHECK OPTION

If a view is updatable, `INSERT`, `UPDATE`, or `DELETE` operations can be made on the view to insert new rows into the base table(s), or to modify or delete existing rows.

However, the update could potentially cause the modified row to no longer be a part of the view, and what happens if the view is used to insert a row that does not match the view definition?

To prevent updates or inserts that do not match the `WHERE` condition of the view, the `WITH CHECK OPTION` needs to be specified after the view's `SELECT` statement. This clause tells InterBase/Firebird to verify an `UPDATE` or `INSERT` statement against the `WHERE` condition. If the modified or inserted row does not match the view definition, the statement fails and InterBase/Firebird returns an error.

3.4.2 Alter View

A view can be altered in the View Editor, opened by double-clicking on the view name in the DB Explorer. Alternatively use the DB Explorer's right mouse-click menu item Edit View or key combination [Ctrl + O].

Alterations may be made directly in the SQL input page; fields, dependencies and triggers can be examined in their respective pages before field deletion.

When altering a view, IBExpert actually does nothing other than create a new view of the same name as the old one, replacing it after committing.

3.4.3 Drop View/Delete View

When a view is dropped it is deleted for good. A view cannot be dropped if it is used elsewhere in the database's metadata. For example, if the view to be dropped is included in the definition of another view, a stored procedure or any `CHECK` constraint, the dependent object must first be dropped before the view can be dropped. Any existent dependencies can be viewed on the View Editor / Dependencies page. Most database objects can be dropped here directly on the Dependencies page or the Dependencies Viewer by using the right-click menu on the selected object, and choosing the menu item Drop Object or [Ctrl + Del].

To drop a view, use the DB Explorer right mouse button menu item Drop View... (or [Ctrl + Del]).

IBExpert asks for confirmation:

before finally dropping the view. Once dropped, it cannot be retrieved.

Alternatively the `DROP VIEW` statement can be used in IBExpert's SQL Editor. It has the following syntax:

```
DROP VIEW <view_name>;
```

For example, to drop the `PHONE_LIST` view in the sample `EMPLOYEE` database, the following statement should be issued:

```
DROP VIEW PHONE_LIST;
```

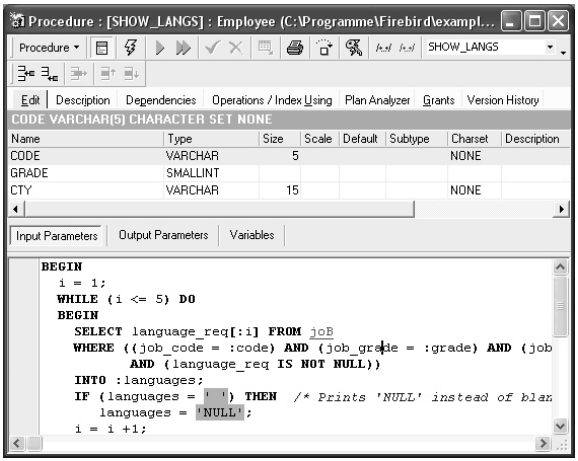
Please note that a view can only be dropped by its creator or the SYSDBA.

3.5 Stored Procedure

A stored procedure is a series of SQL commands stored as a self-contained program in the database as part of the database's metadata (also known as routine). They can be

started by the EXECUTE PROCEDURE command with specification of the procedure name and a list of parameters.

It is similar to a trigger, but is not automatically executed.



It is written in InterBase procedure and trigger language. It can perform special processing on the metadata and data within the database. Program execution occurs on the server.

Each stored procedure is a stand-alone module of code that can be executed interactively or as part of a SELECT statement, from another stored procedure or from another application environment.

They can be invoked directly from applications, or can be substituted for a table or view in a SELECT statement; they can receive input parameters and return values to applications.

With the Client/Server database concept, it is important that the database is not just used to store data, but is actively involved in the data query and data manipulation processes. As the database must also be able to guarantee data integrity, it is important that the database can also handle more complex operations that just simple comparisons. InterBase/Firebird uses stored procedures as the programming environment for integrating active processes in the database.

The stored procedure language is a language created to run in a database. For this reason its range is limited to database operations and necessary functions.

Stored procedures provide SQL enhancements that support variables, comments, declarative statements, conditional testing and looping as programming elements. They have full access to SQL DML statements allowing a multitude of command types; they cannot however execute DDL statements, i.e. a stored procedure cannot create a table.

Stored procedures offer the following advantages when implementing applications:

1. Reduction of network traffic by off-loading application processes from the client to the server. This is particularly important for remote users using slower modem connections. And for this reason of course, they are fast.

2. Splitting up of complex tasks into smaller and more logical modules. Stored procedures can be invoked by each other. Stored procedures allow a library of standardized database routines to be constructed, that can be called in different ways.

3. They're reusable. Rather than recreate a statement on the client each time it's needed, it's better to store it in the database. They can be shared by numerous applications using a single database. Alterations to the underlying data definitions only need to be implemented in the stored procedure and not in the individual applications themselves. Readability is enhanced, and redundancy, maintenance, and documentation are greatly reduced.

4. Full access to SQL and the database's metadata. This allows certain environments to perform extended operations on the database that might not be possible from another application language. The language even offers functions that are not available in SQL, e.g. `IF…WHEN…ELSE`, `DECLARE VARIABLE`, `SUSPEND`, etc.

5. Enhanced security: if database operations such as `INSERT`, `ALTER` or `DROP` can only be performed on a table by stored procedures, the user has no privileges to access the table directly. The only right the user has is to execute the stored procedure.

6. As stored procedures are part of InterBase or Firebird, it is irrelevant which front end is subsequently used, be it Delphi, PHP or other.

There are no disadvantages to using stored procedures. There are however, two limitations. Firstly, any variable information must be able to be passed to the stored procedure as parameters or the information must be placed in a table that the stored procedure can access. Secondly, the procedure and trigger language may be too limited for complex calculations. Stored procedures should be used under the following circumstances:

1. If an operation can be carried out completely on the server with no necessity to obtain information from the user while the operation is in process. When invoking a stored procedure these input parameters can be incorporated in the stored procedure.

2. If an operation requires a large quantity of data to be processed, whose transfer across the network to the client application would cost an enormous amount of time.

3. If the operation must be performed periodically or frequently.

4. If the operation is performed in the same manner by a number of different processes, or processes within the application, or by different applications.

The stored procedure must contain all statements necessary for the database connection, creation or alteration of the stored procedure, and finally the disconnection from the database.

All SQL scripts can be incorporated into a stored procedure and up to 10 SQLs in one procedure, as well as the additional functions already mentioned, making stored procedures considerably quicker and more flexible than SQL.

Stored procedures can often be used as an alternative to views (being more flexible and offering more control) as the `ORDER BY` instruction cannot be used in a view (the data sets are displayed as determined by the optimizer, which is not always intelligent!). In such a case, a stored procedure should be used.

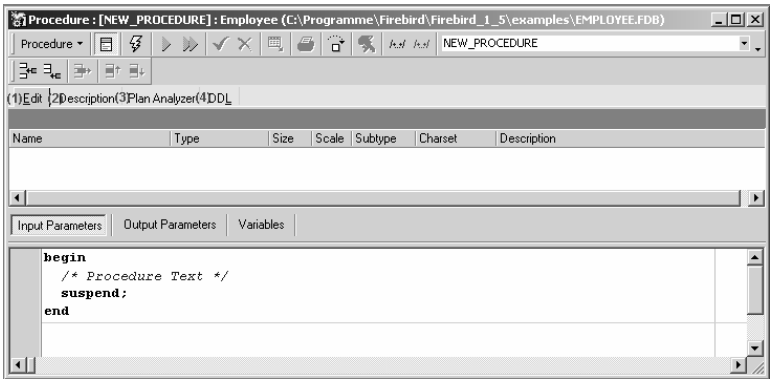
Stored procedures are almost identical to triggers, the only exception being the way they are called. Triggers are called automatically when a change to a row in a table occurs. Most of what is said about stored procedures applies to triggers as well.

3.5.1 New Procedure

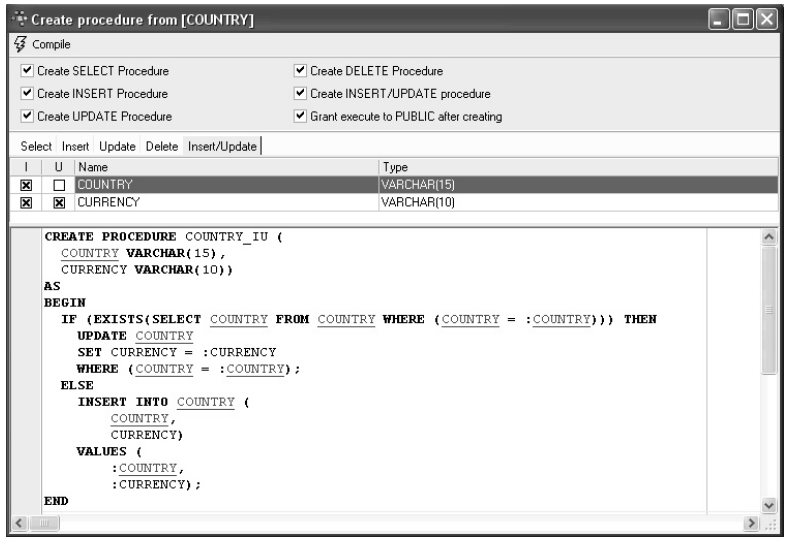
There are numerous ways to approach creating a new stored procedure:

1. Using the IBExpert menu item Database / New Procedure or using the New Procedure icon on the New Database Object toolbar to start the Procedure Editor.

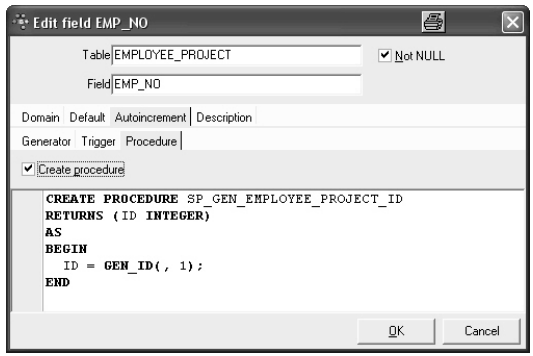
2. From the DB Explorer by right-clicking on the highlighted procedure branch of the relevant connected database (or key combination [Ctrl + N]) which also starts the Procedure Editor.



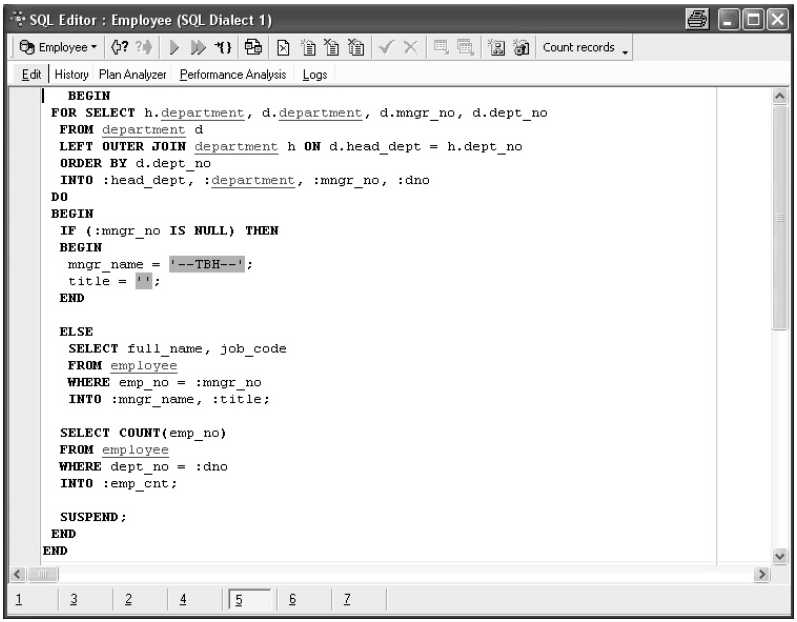
3. A stored procedure can also be created directly from a selected table in the DB Explorer, using the right-click pop-up menu item Create SIUD procedures.



4. Or created directly from the Field Editor.



5. Or created in the IBExpert SQL Editor, and then saved as a stored procedure. When an SQL script has been successfully committed, and the results are as wished, the script can be integrated into a stored procedure using the stored procedure button. The stored procedure script appears, and simply needs to be named and completed.



The CREATE PROCEDURE statement has the following syntax:

```
CREATE PROCEDURE <Procedure_Name>
<Input_Parameter_List>
RETURNS
<Return_Parameter_List>
AS
<Local_Variable_Declarations>
BEGIN
<Procedure_Body>
END
```

The CREATE and RETURNS statements (if there is a return statement) comprise the stored procedure's header. Everything following the AS keyword is the procedure's body.

There can also be statements between the AS and BEGIN keywords that are also considered part of the body. These statements declare local variables for the stored procedure, and are detailed under Stored Procedure Language.

Since IBExpert version 2005.03.12 there is added support for following Firebird 2 features:

- DECLARE <cursor_name> CURSOR FOR ...
- OPEN <cursor_name>
- FETCH <cursor_name> INTO ...
- CLOSE <cursor_name>
- LEAVE <label>

- NEXT VALUE FOR <generator>

Further information explaining the necessary components can be found under the subject Procedure Editor, started using the first two menu options (i.e. Database menu and DB Explorer right mouse button menu).

The Procedure Editor has its own toolbar (see Procedure Editor toolbar). To the right of the toolbar, the new procedure name can be specified. The procedure name follows the naming convention for any InterBase/Firebird object and must be unique.

The Lazy Mode icon can be used to switch the lazy mode on and off as wished:

The New Procedure Editor has four pages: **(1)** Edit, **(2)** Description, **(3)** Plan Analyzer and **(4)** DDL, described under Procedure Editor. A new procedure is created on the Procedure Editor / Edit page.

Stored Procedure parameters (input and output/returns)

Input parameters are a list of variables (=values) that are passed into the procedure from the client application. These variables can be used within the procedure to modify its behavior.

The return parameter (or output parameter) list represents values that the procedure can pass back to the client application, such as the result of a calculation. Each list is in the following format:

```
ParameterName1 ParameterType,
ParameterName2 ParameterType,
...
ParameterNameN ParameterType
```

ParameterType is any valid InterBase/Firebird data type except blob, domain and arrays of data types.

Local variables / DECLARE VARIABLE statement

Local variables can be defined Within the procedure body. Local variables of any InterBase/Firebird type can be declared within a stored procedure. As with any other structured programming environment, these variables only exist while the procedure is running, and their scope is local to the procedure. They are invisible outside the procedure and are destroyed when the procedure finishes. There are no global variables available with stored procedures and triggers. If values need to be shared by two or more procedures, they should either be passed as parameters or stored in a table.

Local variables are declared immediately after the AS clause, using the DECLARE VARIABLE statement. For example the variable ANY_SALES is declared in the EMPLOYEE database's DELETE_EMPLOYEE procedure:

```
DECLARE VARIABLE ANY_SALES INTEGER;
```

Each variable must be declared in its own DECLARE VARIABLE statement, as each statement can declare only one variable.

Procedure body

The procedure body consists of a compound statement, which can be any number of InterBase/Firebird procedure and trigger language statements. The procedure body starts with a BEGIN statement, followed by any local variable declarations, and ends with an END statement.

BEGIN and END must also be used to surround any block of statements that logically belong together, such as the statements within a loop.

BEGIN and END do not need terminating characters, except for the final END within the procedure.

Comment Procedure Body/Uncomment Procedure Body

It certain situations it may be necessary to disable certain commands or parts of SQL text. This can be easily done temporarily, without it being necessary to delete these commands.

Simply select the rows concerned in the SQL Editor, and select either the editor toolbar icons:

the right mouse button menu item Comment Selected, or key combination [Ctrl + Alt + .]. This alters command rows to comments. The commented text can be reinstated as SQL text by using Uncomment Procedure icon (above), the right mouse button menu item Uncomment Selected, or [Ctrl+ Alt + ,].

Lazy Mode

Using lazy mode, the programmer does not have to worry about which input and output parameters need to be considered. It can be switched between lazy mode and classic mode using the

icon in the Procedure Editor and Trigger Editor.

The possibility to select domains as a data type for input/output parameters and variables has been added in IBExpert version 2004.8.5.1. In this case IBExpert copies information from the domain definition to the native data type of the parameter/variable. It is now also possible to drag 'n' drop a domain from the Database Explorer.

3.5.2 Stored Procedure Editor

The Procedure Editor can be started using the Database / New Procedure menu item; from the DB Explorer, using the right mouse-click menu or double-clicking on an existing procedure.

Please refer to New Procedure when creating a stored procedure for the first time.

The Procedure Editor has its own toolbar (see Procedure Editor Toolbar) and offers the following options:

- Edit
- Description
- Dependencies
- Operations/Index Using
- Plan Analyzer
- DDL
- Grants
- Version History

Edit

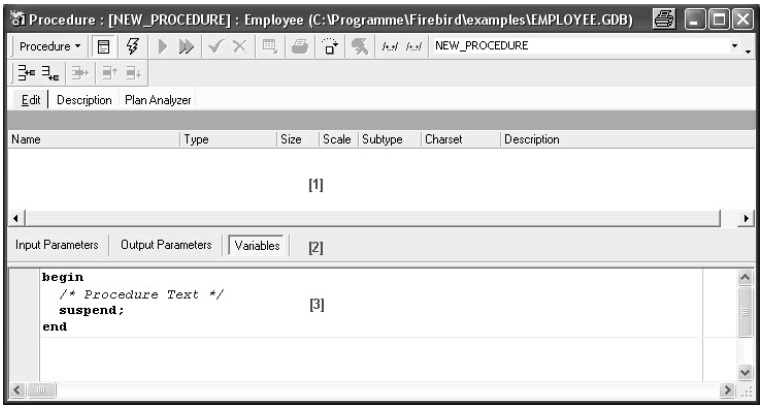
The `CREATE PROCEDURE` statement has the following syntax:

```
CREATE PROCEDURE <Procedure_Name>
<Input_Parameter_List>
RETURNS
<Return_Parameter_List>
AS
<Local_Variable_Declarations>
BEGIN
<Procedure_Body>
END
```

A stored procedure comprises the following components:

- input parameters
- output parameters (returns)
- variables
- procedure body
- comments (optional)

If the lazy mode is switched off, the Edit dialog offers a single SQL input area, with the procedure syntax already displayed. If the lazy mode is switched on, the Edit dialog consists of three areas:



(1) The field grid, where new parameters can be specified.

(2) In the middle are three buttons specifying the parameter type, i.e. input parameters, output parameters and variables. It is possible to drag 'n' drop parameters/variables from the field grid onto the corresponding button to move them. For example, click the Output Parameters button, drag a named variable from the field grid onto the Variable button. Click the Variable button to view the new variable in the field grid.

(3) Below this is the SQL panel for direct code input. Again the procedure syntax is already displayed to help the user.

For those who do not wish to use the basic syntax template, or wish to add certain statements themselves to create their own standard, this can be done using the IBExpert menu item Options / General Templates, and clicking on either the Standard Mode or Lazy Mode under New Procedure.

As with all SQL input windows, the SQL Editor Menu can be called using the right mouse button.

The basic parameters of the stored procedure are set here as SQL text for creating the procedure. A parameter can have any InterBase/Firebird data type except blob or array. The input parameters are set in brackets after the procedure name, the output parameters are set in brackets after the `RETURNS` statement, and the procedure body written in InterBase procedure and trigger language, bracketed by `BEGIN` and `END` statements.

New parameters can be quickly and easily specified, by clicking the respective button (i.e. input, output or variables), and inserting field information using the respective icon or right-click menu, in the same manner as creating a new table.

Local variables of any InterBase/Firebird type can be declared within a stored procedure (please refer to local variables), after the `AS` keyword and before the `BEGIN` (which marks the begin of the procedure body).

Alternatively, the required information can be entered directly in the editor's input panel and field names can be simply dragged from the DB Explorer or SQL Assistant into the procedure script. The code insight can be used to save time wasted searching for correct names, and to prevent any possible spelling errors. A right mouse-click within this area produces the SQL Editor menu.

The input parameters are set with their types in brackets after the procedure name. By checking the Code Parameter option under Options / Editor Options / Code Insight, a list of the necessary parameters automatically appears. Output parameters are specified in the same way after `RETURNS`. The operations to be performed by the procedure are described after the `BEGIN` statement. Please refer to stored procedure language for further details.

After inputting the required information, the stored procedure can be executed using [F9] or the relevant icon. The statement window appears, where the resulting SQL statement can be viewed before committing. If necessary the code can subsequently be debugged using the debugging icon or [Shift + Ctrl + D]. (Please refer to Debug Procedure for more details.).

Don't forget to finally compile the new procedure using the respective toolbar icon or [F9], and, if desired, autogrant privileges, again using the respective toolbar icon or key combination [Ctrl + F8].

Description

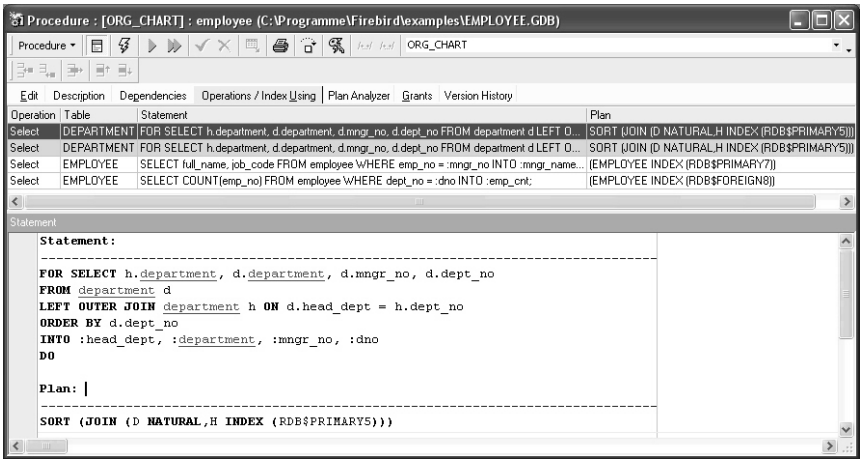
Please refer to Table Editor / Description.

Dependencies

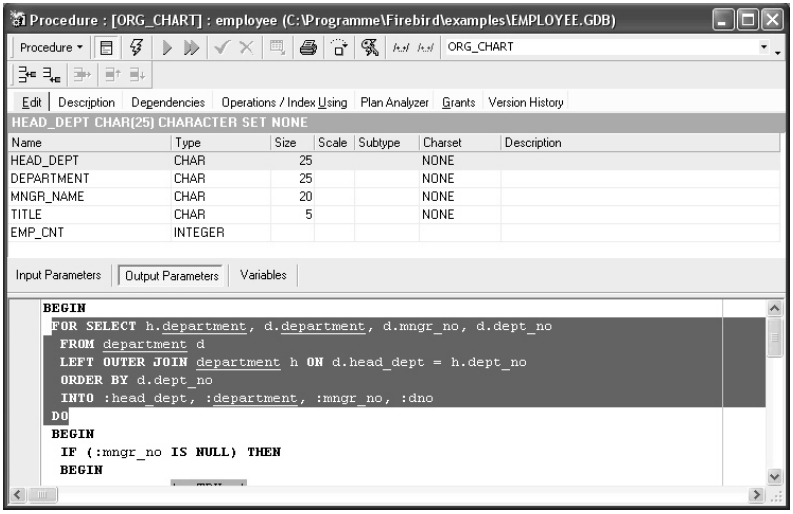
See Table Editor / Dependencies.

Operations/Index Using

This page dissects the procedure into single operations, and examines them to see whether they use a plan (i.e. index) or not. The `ORG_CHART` procedure in the sample `EMPLOYEE` database displays red-marked entries, which indicates a plan `NATURAL` (i.e. no indices are used). When an operation is selected, the statement for this operation is displayed in the lower window:



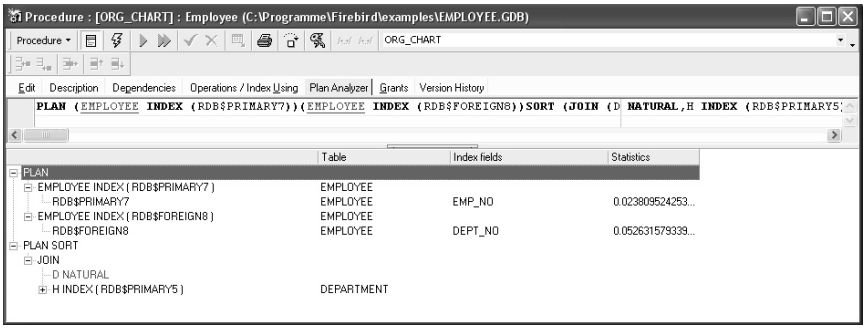
By double-clicking on a selected operation, the SQL panel appears, highlighting the SQL statements for this operation, enabling further analysis and amendments. For example, should perhaps the `ORDER BY` be altered, or perhaps a different `JOIN`?



Input and output parameters and variable fields can be displayed, by clicking on the buttons in the center of the editor. Alterations may be made directly in the SQL window and subsequently executed and committed.

New to IBExpert v. 2.5.0.47 is the SP/Triggers/Views Analyzer in the IBExpert Tools menu. This loads *all* stored procedures and triggers in the active database, and all `NATURAL` operations are highlighted.

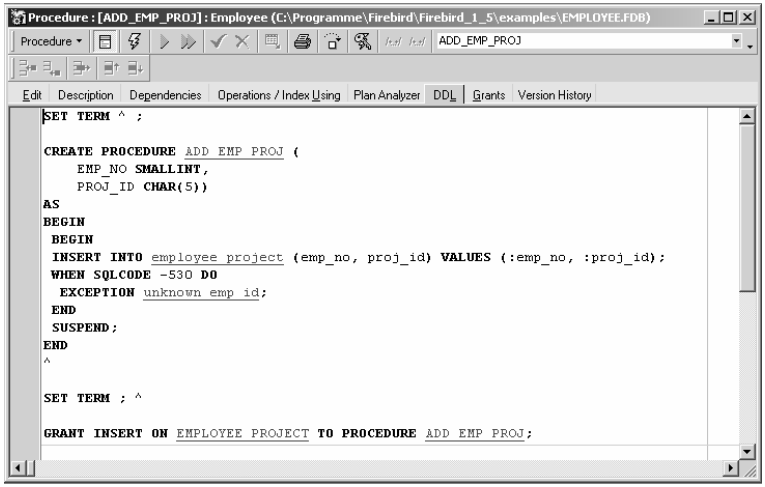
Plan Analyzer



Please refer to SQL Editor / Plan Analyzer.

DDL

The DDL page is new to IBExpert version 2004.6.17. It includes the CREATE PROCEDURE statement, stored procedure and parameter descriptions and GRANT statements.



Grants

Please refer to Table Editor / Grants and autogrant privileges.

Version History

Please refer to View / Version History.

3.5.3 Executing Stored Procedures

InterBase/Firebird stored procedures are divided into two groups with respect to how they are called. Select procedures return result values through output parameters, be-

cause they can be used in place of a table name in an SQL SELECT statement. Non-select procedures do not return values.

The simplest way to execute a stored procedure is to use the EXECUTE PROCEDURE statement. This statement can be used in one of the following ways:

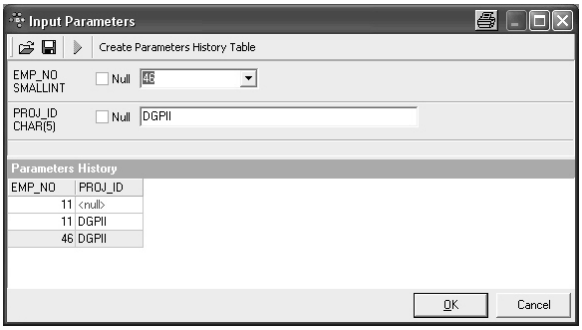
- From within another stored procedure.
- From within a trigger.
- From an application.

When a procedure is executed from within an InterBase/Firebird application, such as another procedure or a trigger, it has the following syntax:

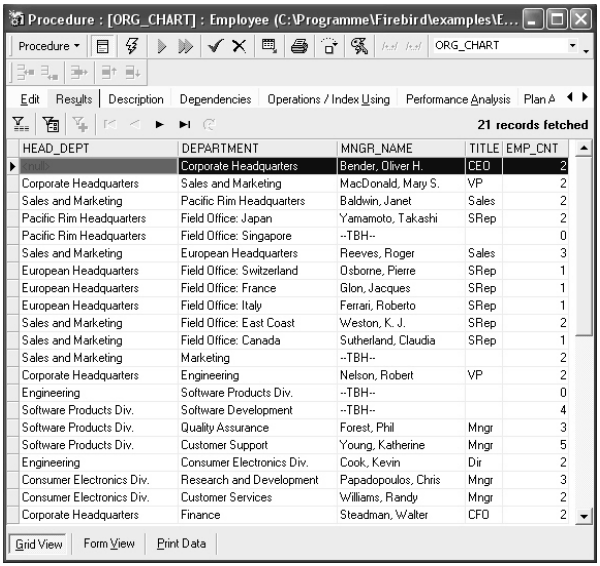
```
EXECUTE PROCEDURE
<procedure_name>
<input_parameter_list>
RETURNING_VALUES
<parameter_list>
```

If the procedure requires input variables, or if it is to return output variables, the relevant parameters need to be specified. In each case, <parameter_list> is a list of parameters, separated by commas (see stored procedure parameters for further information).

When using IBExpert's Procedure Editor to execute a procedure, IBExpert tells you whether input parameters need to be entered:



before displaying the return values (= output or results) on the Results page:



HEAD_DEPT	DEPARTMENT	MNGR_NAME	TITLE	EMP_CNT
<null>	Corporate Headquarters	Bender, Oliver H.	CEO	2
Corporate Headquarters	Sales and Marketing	MacDonald, Mary S.	VP	2
Sales and Marketing	Pacific Rim Headquarters	Baldwin, Janet	Sales	2
Pacific Rim Headquarters	Field Office: Japan	Yamamoto, Takashi	SRep	2
Pacific Rim Headquarters	Field Office: Singapore	--TBH--		0
Sales and Marketing	European Headquarters	Reeves, Roger	Sales	3
European Headquarters	Field Office: Switzerland	Osborne, Pierre	SRep	1
European Headquarters	Field Office: France	Glon, Jacques	SRep	1
European Headquarters	Field Office: Italy	Ferrari, Roberto	SRep	1
Sales and Marketing	Field Office: East Coast	Weston, K. J.	SRep	2
Sales and Marketing	Field Office: Canada	Sutherland, Claudia	SRep	1
Sales and Marketing	Marketing	--TBH--		2
Corporate Headquarters	Engineering	Nelson, Robert	VP	2
Engineering	Software Products Div.	--TBH--		0
Software Products Div.	Software Development	--TBH--		4
Software Products Div.	Quality Assurance	Forest, Phil	Mngr	3
Software Products Div.	Customer Support	Young, Katherine	Mngr	5
Engineering	Consumer Electronics Div.	Cook, Kevin	Dir	2
Consumer Electronics Div.	Research and Development	Papadopoulos, Chris	Mngr	3
Consumer Electronics Div.	Customer Services	Williams, Randy	Mngr	2
Corporate Headquarters	Finance	Steadman, Walter	CFO	2

Select Procedures

It is possible to use a stored procedure in place of the table reference in a `SELECT` statement. This type of procedure is known as a select procedure.

When a stored procedure is used in place of a table, the procedure should return multiple columns or rows, i.e. it assigns values to output parameters and uses `SUSPEND` to return these values. This allows the `SELECT` statement to filter the results further by different criteria.

The `SUSPEND` statement is used to suspend execution of the procedure and return the contents of the output variables back to the calling statement. If the stored procedure returns multiple rows, the `SUSPEND` statement needs to be used inside a `FOR SELECT … DO` loop to return the rows one at a time.

Non–Select Procedures

Non-select procedures are procedures that do not return any results.

3.5.4 Procedure using Substring() function (Susbstr Procedure)

Unfortunately Firebird 1.5 does not allow any variable parameters in the `SUBSTRING()` SQL function. Although there are diverse UDF implementations, for those preferring to use stored procedures, here is an example from Lucas Franzen:

(For those of you who may be wondering what on earth 'Donaudampfschiffahrtsgesellschaftskapitän' is, it is the German word for "Donau Steam Navigation Company Captain"!).

Call:

```
SELECT RESULT FROM SP_SUBSTRING
   ( INPUTSTRING, STARTPOS, NO_CHAR_FROM_STARTPOS ).
```

E.g.: `SELECT RESULT FROM SP_SUBSTRING`

`   ( 'Donaudampfschiffahrtsgesellschaftskapitän', 1, 10 )` --> Donaudampf

E.g.: `SELECT RESULT FROM SP_SUBSTRING`

`   ( 'Donaudampfschiffahrtsgesellschaftskapitän', 35, 8 )` --> kapitän

```
CREATE PROCEDURE SP_SUBSTRING (
   SRC                              VARCHAR (255),
   START_AT                         INTEGER,
   NLEN                             INTEGER
   )
RETURNS (
   RESULT                         VARCHAR (255)
   )
AS
   declare variable II INTEGER;
   declare variable VGL VARCHAR(255);
   declare variable PFX VARCHAR(255);
   declare variable C CHAR(1);
BEGIN

   /* Version : 1 */
   /* Author: LUC, 08.01.2003*/
   /* Description: */
   /*            */

  IF ( START_AT <= 0 ) THEN START_AT = 1;
  IF ( START_AT > 255 ) THEN START_AT = 255;

  IF ( NLEN > 255 ) THEN NLEN = 255;
  IF ( NLEN < 1 OR NLEN IS NULL ) THEN NLEN = 1;

  VGL = '';
  RESULT = '';
  PFX = '';

   IF ( START_AT > 1 ) THEN
   BEGIN
     II = 1;
     WHILE ( II < START_AT ) DO
     BEGIN
       PFX = PFX || '_';
       II = II + 1;
     END
   END
```

```
    II = START_AT;
    WHILE ( II < NLEN + START_AT ) DO
    BEGIN
      /* WHAT DOES THE STRING LOOK LIKE AT THE CURRENT POSITION, I.E.
QUERY THE CURRENT CHARACTER */
     C = ' ';

      IF ( SRC LIKE PFX || ' %' ) THEN C = ' ';
      ELSE IF ( SRC LIKE PFX || 'A%' ) THEN C = 'A';
      ELSE IF ( SRC LIKE PFX || 'B%' ) THEN C = 'B';
      ELSE IF ( SRC LIKE PFX || 'C%' ) THEN C = 'C';
      ELSE IF ( SRC LIKE PFX || 'D%' ) THEN C = 'D';
      ELSE IF ( SRC LIKE PFX || 'E%' ) THEN C = 'E';
      ELSE IF ( SRC LIKE PFX || 'F%' ) THEN C = 'F';
      ELSE IF ( SRC LIKE PFX || 'G%' ) THEN C = 'G';
      ELSE IF ( SRC LIKE PFX || 'H%' ) THEN C = 'H';
      ELSE IF ( SRC LIKE PFX || 'I%' ) THEN C = 'I';
      ELSE IF ( SRC LIKE PFX || 'J%' ) THEN C = 'J';
      ELSE IF ( SRC LIKE PFX || 'K%' ) THEN C = 'K';
      ELSE IF ( SRC LIKE PFX || 'L%' ) THEN C = 'L';
      ELSE IF ( SRC LIKE PFX || 'M%' ) THEN C = 'M';
      ELSE IF ( SRC LIKE PFX || 'N%' ) THEN C = 'N';
      ELSE IF ( SRC LIKE PFX || 'O%' ) THEN C = 'O';
      ELSE IF ( SRC LIKE PFX || 'P%' ) THEN C = 'P';
      ELSE IF ( SRC LIKE PFX || 'Q%' ) THEN C = 'Q';
      ELSE IF ( SRC LIKE PFX || 'R%' ) THEN C = 'R';
      ELSE IF ( SRC LIKE PFX || 'S%' ) THEN C = 'S';
      ELSE IF ( SRC LIKE PFX || 'T%' ) THEN C = 'T';
      ELSE IF ( SRC LIKE PFX || 'U%' ) THEN C = 'U';
      ELSE IF ( SRC LIKE PFX || 'V%' ) THEN C = 'V';
      ELSE IF ( SRC LIKE PFX || 'W%' ) THEN C = 'W';
      ELSE IF ( SRC LIKE PFX || 'X%' ) THEN C = 'X';
      ELSE IF ( SRC LIKE PFX || 'Y%' ) THEN C = 'Y';
      ELSE IF ( SRC LIKE PFX || 'Z%' ) THEN C = 'Z';

      ELSE IF ( SRC LIKE PFX || 'a%' ) THEN C = 'a';
      ELSE IF ( SRC LIKE PFX || 'b%' ) THEN C = 'b';
      ELSE IF ( SRC LIKE PFX || 'c%' ) THEN C = 'c';
      ELSE IF ( SRC LIKE PFX || 'd%' ) THEN C = 'd';
      ELSE IF ( SRC LIKE PFX || 'e%' ) THEN C = 'e';
      ELSE IF ( SRC LIKE PFX || 'f%' ) THEN C = 'f';
      ELSE IF ( SRC LIKE PFX || 'g%' ) THEN C = 'g';
      ELSE IF ( SRC LIKE PFX || 'h%' ) THEN C = 'h';
      ELSE IF ( SRC LIKE PFX || 'i%' ) THEN C = 'i';
      ELSE IF ( SRC LIKE PFX || 'j%' ) THEN C = 'j';
      ELSE IF ( SRC LIKE PFX || 'k%' ) THEN C = 'k';
      ELSE IF ( SRC LIKE PFX || 'l%' ) THEN C = 'l';
      ELSE IF ( SRC LIKE PFX || 'm%' ) THEN C = 'm';
      ELSE IF ( SRC LIKE PFX || 'n%' ) THEN C = 'n';
      ELSE IF ( SRC LIKE PFX || 'o%' ) THEN C = 'o';
      ELSE IF ( SRC LIKE PFX || 'p%' ) THEN C = 'p';
      ELSE IF ( SRC LIKE PFX || 'q%' ) THEN C = 'q';
```

```
ELSE IF ( SRC LIKE PFX || 'r%' ) THEN C = 'r';
ELSE IF ( SRC LIKE PFX || 's%' ) THEN C = 's';
ELSE IF ( SRC LIKE PFX || 't%' ) THEN C = 't';
ELSE IF ( SRC LIKE PFX || 'u%' ) THEN C = 'u';
ELSE IF ( SRC LIKE PFX || 'v%' ) THEN C = 'v';
ELSE IF ( SRC LIKE PFX || 'w%' ) THEN C = 'w';
ELSE IF ( SRC LIKE PFX || 'x%' ) THEN C = 'x';
ELSE IF ( SRC LIKE PFX || 'y%' ) THEN C = 'y';
ELSE IF ( SRC LIKE PFX || 'z%' ) THEN C = 'z';

ELSE IF ( SRC LIKE PFX || '0%' ) THEN C = '0';
ELSE IF ( SRC LIKE PFX || '1%' ) THEN C = '1';
ELSE IF ( SRC LIKE PFX || '2%' ) THEN C = '2';
ELSE IF ( SRC LIKE PFX || '3%' ) THEN C = '3';
ELSE IF ( SRC LIKE PFX || '4%' ) THEN C = '4';
ELSE IF ( SRC LIKE PFX || '5%' ) THEN C = '5';
ELSE IF ( SRC LIKE PFX || '6%' ) THEN C = '6';
ELSE IF ( SRC LIKE PFX || '7%' ) THEN C = '7';
ELSE IF ( SRC LIKE PFX || '8%' ) THEN C = '8';
ELSE IF ( SRC LIKE PFX || '9%' ) THEN C = '9';

ELSE IF ( SRC LIKE PFX || 'ä%' ) THEN C = 'ä';
ELSE IF ( SRC LIKE PFX || 'ö%' ) THEN C = 'ö';
ELSE IF ( SRC LIKE PFX || 'ü%' ) THEN C = 'ü';
ELSE IF ( SRC LIKE PFX || 'Ä%' ) THEN C = 'Ä';
ELSE IF ( SRC LIKE PFX || 'Ö%' ) THEN C = 'Ö';
ELSE IF ( SRC LIKE PFX || 'Ü%' ) THEN C = 'Ü';
ELSE IF ( SRC LIKE PFX || 'ß%' ) THEN C = 'ß';

ELSE IF ( SRC LIKE PFX || '!%' ) THEN C = '!';
ELSE IF ( SRC LIKE PFX || '"%' ) THEN C = '"';
ELSE IF ( SRC LIKE PFX || '§%' ) THEN C = '§';
ELSE IF ( SRC LIKE PFX || '$%' ) THEN C = '$';
ELSE IF ( SRC LIKE PFX || '&%' ) THEN C = '&';
ELSE IF ( SRC LIKE PFX || '/%' ) THEN C = '/';
ELSE IF ( SRC LIKE PFX || '(%' ) THEN C = '(';
ELSE IF ( SRC LIKE PFX || ')%' ) THEN C = ')';
ELSE IF ( SRC LIKE PFX || '=%' ) THEN C = '=';

ELSE IF ( SRC LIKE PFX || '@%' ) THEN C = '@';
ELSE IF ( SRC LIKE PFX || '+%' ) THEN C = '+';
ELSE IF ( SRC LIKE PFX || '*%' ) THEN C = '*';
ELSE IF ( SRC LIKE PFX || '~%' ) THEN C = '~';
ELSE IF ( SRC LIKE PFX || '#%' ) THEN C = '#';
ELSE IF ( SRC LIKE PFX || ''''%' ) THEN C = '''';
ELSE IF ( SRC LIKE PFX || '-%' ) THEN C = '-';

ELSE IF ( SRC LIKE PFX || 'Á%' ) THEN C = 'Á';
ELSE IF ( SRC LIKE PFX || 'É%' ) THEN C = 'É';
ELSE IF ( SRC LIKE PFX || 'Í%' ) THEN C = 'Í';
ELSE IF ( SRC LIKE PFX || 'Ó%' ) THEN C = 'Ó';
ELSE IF ( SRC LIKE PFX || 'Ú%' ) THEN C = 'Ú';
```

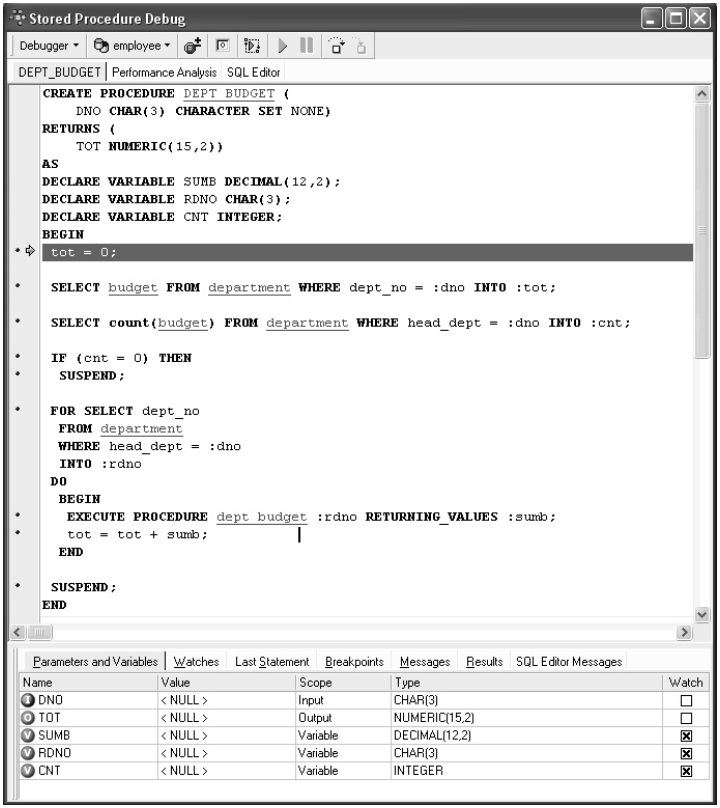
```
    ELSE IF ( SRC LIKE PFX || 'á%' ) THEN C = 'á';
    ELSE IF ( SRC LIKE PFX || 'é%' ) THEN C = 'é';
    ELSE IF ( SRC LIKE PFX || 'í%' ) THEN C = 'í';
    ELSE IF ( SRC LIKE PFX || 'ó%' ) THEN C = 'ó';
    ELSE IF ( SRC LIKE PFX || 'ú%' ) THEN C = 'ú';

    ELSE IF ( SRC LIKE PFX || 'À%' ) THEN C = 'À';
    ELSE IF ( SRC LIKE PFX || 'È%' ) THEN C = 'È';
    ELSE IF ( SRC LIKE PFX || 'Ì%' ) THEN C = 'Ì';
    ELSE IF ( SRC LIKE PFX || 'Ò%' ) THEN C = 'Ò';
    ELSE IF ( SRC LIKE PFX || 'Ù%' ) THEN C = 'Ù';
    ELSE IF ( SRC LIKE PFX || 'à%' ) THEN C = 'à';
    ELSE IF ( SRC LIKE PFX || 'è%' ) THEN C = 'è';
    ELSE IF ( SRC LIKE PFX || 'ì%' ) THEN C = 'ì';
    ELSE IF ( SRC LIKE PFX || 'ò%' ) THEN C = 'ò';
    ELSE IF ( SRC LIKE PFX || 'ù%' ) THEN C = 'ù';

    ELSE IF ( SRC LIKE PFX || 'Â%' ) THEN C = 'Â';
    ELSE IF ( SRC LIKE PFX || 'Ê%' ) THEN C = 'Ê';
    ELSE IF ( SRC LIKE PFX || 'Î%' ) THEN C = 'Î';
    ELSE IF ( SRC LIKE PFX || 'Ô%' ) THEN C = 'Ô';
    ELSE IF ( SRC LIKE PFX || 'Û%' ) THEN C = 'Û';
    ELSE IF ( SRC LIKE PFX || 'â%' ) THEN C = 'â';
    ELSE IF ( SRC LIKE PFX || 'ê%' ) THEN C = 'ê';
    ELSE IF ( SRC LIKE PFX || 'î%' ) THEN C = 'î';
    ELSE IF ( SRC LIKE PFX || 'ô%' ) THEN C = 'ô';
    ELSE IF ( SRC LIKE PFX || 'û%' ) THEN C = 'û';

    ELSE IF ( SRC LIKE PFX || '{%' ) THEN C = '{';
    ELSE IF ( SRC LIKE PFX || '}%' ) THEN C = '}';
    ELSE IF ( SRC LIKE PFX || '[%' ) THEN C = '[';
    ELSE IF ( SRC LIKE PFX || ']%' ) THEN C = ']';

    RESULT = RESULT || :C;

    PFX = PFX || '_';
    II = II + 1;
    IF ( II > 255 ) THEN
    BEGIN
      SUSPEND;
      EXIT;
    END
  END
   SUSPEND;
END
```

3.5.5 Debug Procedure or Trigger (IBExpert Debugger)

A stored procedure or trigger can simply and quickly be debugged in IBExpert. (This feature is unfortunately not included in the Personal Edition.)

Simply open the procedure or trigger in the Procedure or Trigger Editor by double-clicking on the procedure/trigger name in the DB Explorer and click the Debug icon on the Procedure or Trigger Editor toolbar (or [Shift + Ctrl + D]) to start the Debugger window.

The Debug Procedure/Trigger Editor comprises 3 pages, the Debug page (described here), Performance Analysis (please refer to SQL Editor / Performance Analysis for more details) and SQL Editor (please refer to Tools / SQL Editor for further information).



The upper half of this dialog displays the SQL text. The lower area displays a number of tabs:

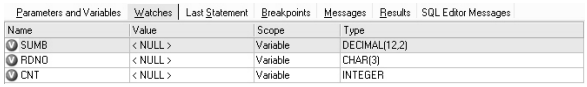
1. Parameters and Variables

The parameters are listed in a grid. The circular symbols to the left of the name indicate whether the parameters are input (I) or output (O). Variables logically have the key (V). Further information displayed here includes the parameter value, scope and data type. The Watch boxes can be checked, to specify which variables should be observed.

Since IBExpert version 2004.9.12.1 there is the added possibility to initialize parameters/variables using values of any data grid. Just drag and drop a cell value from any data grid onto the corresponding node in the parameters/variables list to initialize the variable with the value of the data cell. It is also possible to initialize multiple variables/parameters by holding the [Ctrl] key when dropping. In this case IBExpert searches for the corresponding parameter/variable (by name) for each field in the data record, and if the parameter/variable is found it will be initialized with the value of the field with the same name.

And since IBExpert version 2004.04.01.1 there is added support for default values of input parameters (Firebird 2).

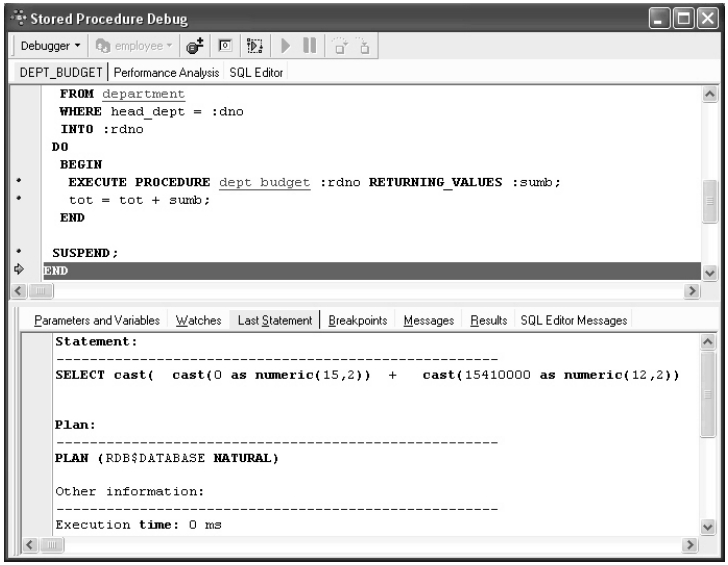
2. Watches



The Watches tab displays those parameters and variables that have been checked for particular observation in the previous window.

3. Last Statement

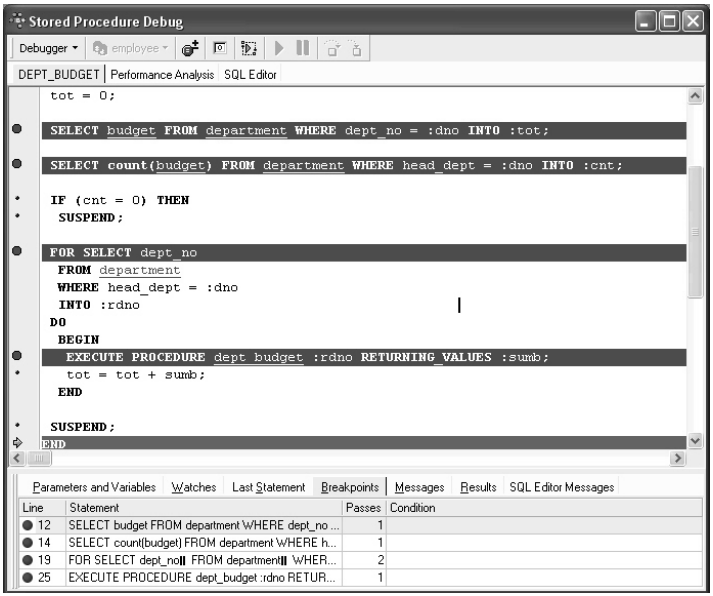
Following execution, the last internal statement is displayed here, along with additional information such as execution time:



4. Breakpoints

This page displays the positions where breakpoints have been specified, using the respective icon in the Debug Procedure toolbar, the [F5] key, or by clicking on the blue points in the SQL left margin.

When the procedure is executed (using the respective icon or [F9]), it always stops automatically at these breakpoints. The procedure can thus be executed step by step, either using [F8] (or the respective toolbar icon) to continue execution step by step (not including the next sublevel), or [F7] (or the respective toolbar icon) to continue step by step including the next sublevel. Please note that this Trace Into [F7] function is new to IBExpert version 2004.04.01.1.

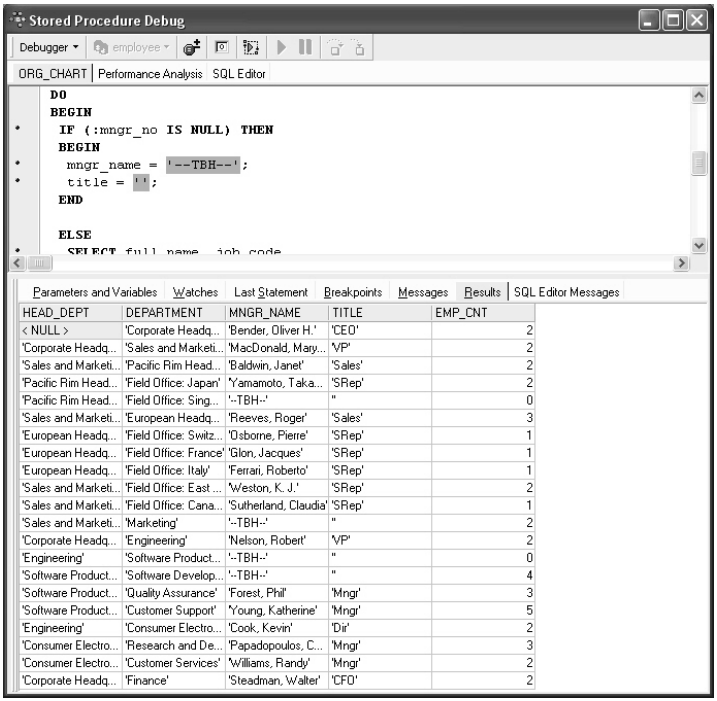


5. Messages

These indicate the sort of error that has occurred and where, by highlighting the relevant SQL row.

6. Results

This page only appears if there are output parameters in the procedure.



HEAD_DEPT	DEPARTMENT	MNGR_NAME	TITLE	EMP_CNT
< NULL >	'Corporate Headq...	'Bender, Oliver H.'	'CEO'	2
'Corporate Headq...	'Sales and Marketi...	'MacDonald, Mary...	'VP'	2
'Sales and Marketi...	'Pacific Rim Head...	'Baldwin, Janet'	'Sales'	2
'Pacific Rim Head...	'Field Office: Japan'	'Yamamoto, Taka...	'SRep'	2
'Pacific Rim Head...	'Field Office: Sing...	'--TBH--'	''	0
'Sales and Marketi...	'European Headq...	'Reeves, Roger'	'Sales'	3
'European Headq...	'Field Office: Switz...	'Osborne, Pierre'	'SRep'	1
'European Headq...	'Field Office: France'	'Glon, Jacques'	'SRep'	1
'European Headq...	'Field Office: Italy'	'Ferrari, Roberto'	'SRep'	1
'Sales and Marketi...	'Field Office: East ...	'Weston, K. J.'	'SRep'	2
'Sales and Marketi...	'Field Office: Cana...	'Sutherland, Claudia'	'SRep'	1
'Sales and Marketi...	'Marketing'	'--TBH--'	''	2
'Corporate Headq...	'Engineering'	'Nelson, Robert'	'VP'	2
'Engineering'	'Software Product...	'--TBH--'	''	0
'Software Product...	'Software Develop...	'--TBH--'	''	4
'Software Product...	'Quality Assurance'	'Forest, Phil'	'Mngr'	3
'Software Product...	'Customer Support'	'Young, Katherine'	'Mngr'	5
'Engineering'	'Consumer Electro...	'Cook, Kevin'	'Dir'	2
'Consumer Electro...	'Research and De...	'Papadopoulos, C...	'Mngr'	3
'Consumer Electro...	'Customer Services'	'Williams, Randy'	'Mngr'	2
'Corporate Headq...	'Finance'	'Steadman, Walter'	'CFO'	2

7. SQL Editor Messages

These are displayed here when applicable.

When debugging a procedure, first take a look at the values of the parameters and then use [F8] to go through the procedure step by step ([F9] executes fully). After each step, all variable values can be seen. Don't forget to work with breakpoints [F5]. Of course, the Debug Procedure toolbar offers all these operations and more.

3.5.6 Alter Procedure

Procedures can be altered directly in the Procedure Editor, started by double-clicking directly on the procedure name in the DB Explorer. Alternatively use the DB Explorer's right mouse-click menu item Edit Procedure or key combination [Ctrl + O].

ALTER PROCEDURE has exactly the same syntax as CREATE PROCEDURE. In fact, when procedures are altered the original procedure definition is replaced. It may seem that ALTER PROCEDURE is therefore not necessary, as a procedure could be dropped and then recreated to carry out any changes. However this will not work if the procedure to be changed is called by another procedure. If procedure A calls procedure B, procedure B cannot be dropped because procedure A depends on its existence.

The SQL syntax for this command is:

```
ALTER PROCEDURE <procedure_name>
<revised_input_parameter_list>
RETURNS
<revised_return_parameter_list>
AS
<local_variable_declarations>
BEGIN
<procedure_body>
END
```


A procedure can only be altered by the original creator or by the SYSDBA user.

3.5.7 Drop Procedure/Delete Procedure

A procedure may only be dropped, if it is not being used at the time of deletion. Also it may not be dropped if it is used by other procedures, triggers, views or SELECTs, until this dependency is removed. The Procedure Editor / Dependencies page displays which database objects use this procedure, and which objects this procedure uses. Most database objects can be dropped directly on the Dependencies page or the Dependencies Viewer by using the right-click menu on the selected object, and choosing the menu item Drop Object or [Ctrl + Del].

To drop a procedure use the DB Explorer right mouse-click menu item Drop View... (or [Ctrl + Del]).

IBExpert asks for confirmation:

before finally dropping the procedure. Once dropped, it cannot be retrieved; the procedure has to be recreated, if a mistake has been made!

Using SQL the syntax is:

```
DROP PROCEDURE <procedure_name>;
```

A procedure can only be dropped by its creator or the SYSDBA.

3.6 Trigger

A trigger is an independent series of commands stored as a self-contained program (SQL script) in the database. Triggers are executed automatically in the database when certain events occur. For example, it is possible to check before an insert, whether a primary key already exists or not, and if necessary allocate a value by a generator. These events are table- and row-based.

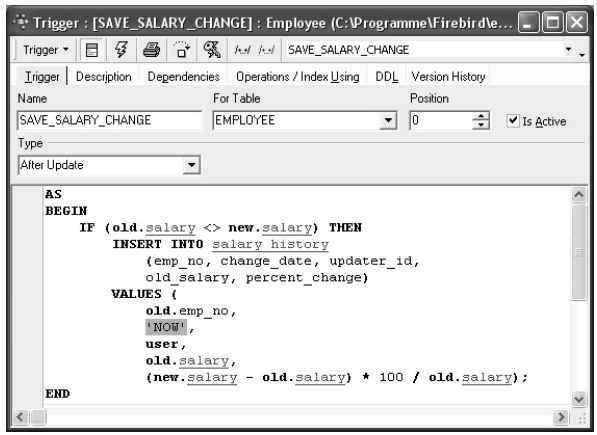
A trigger is a database object mainly used for integrity and security. It can include one or more execute commands. They can also be used as an alarm (= event alerter), that sends an event of a certain name to the InterBase/Firebird Event Manager.

The sequence in which triggers are specified is determined by the term `TRIGGER POSITION`.

When defining a trigger, the trigger type is specified by special keywords:

- `ACTIVE` or `INACTIVE`
- `INSERT` or `UPDATE` or `DELETE`
- `BEFORE` or `AFTER`

Please refer to trigger types for more details.



Triggers take no input parameters and do not return values.

They can be created, edited and deleted using the IBExpert DB Explorer or directly in the IBExpert SQL Editor.

Firebird 1.5 even offers universal triggers (which can be used simultaneously for insert and/or update and/or delete).

An example of a trigger:

```
CREATE TRIGGER TEST_TRIG FOR TEST
ACTIVE BEFORE INSERT POSITION 0
AS
begin
   if(new.id is null) then
      new.id=gen_id(GLOB_ID,1);
end
```

Several triggers can be created for one event. The `POSITION` parameter determines the sequence, with which the trigger is executed (it starts at 0; up to 254 are possible).

Triggers are almost identical to stored procedures, the only exception being the way they are called. Triggers are called automatically when a change to a row in a table occurs. Most of what is said about stored procedures applies to triggers as well.

3.6.1 Trigger Types

Trigger types refer to the trigger status (ACTIVE or INACTIVE), the trigger position (BEFORE or AFTER) and the operation type (INSERT or UPDATE or DELETE).

They are specified following the definition of the table or view name, and before the trigger body.

ACTIVE or INACTIVE

ACTIVE or INACTIVE is specified at the time a trigger is created. ACTIVE is the default if neither of these keywords is specified. An inactive trigger does not execute.

BEFORE or AFTER

A trigger needs to be defined to fire either BEFORE or AFTER an operation. A BEFORE INSERT trigger fires before a new row is actually inserted into the table; an AFTER INSERT trigger fires after the row has been inserted.

BEFORE triggers are generally used for two purposes:

- They can be used to determine whether the operation should proceed, i.e. certain parameters can be tested to determine whether the row should be inserted, updated or deleted or not. If not, an exception can be raised and the transaction rolled back.
- BEFORE triggers can also be used to determine whether there are linked rows that might be affected by the operation. For example, a trigger might be used to automatically reassign sales before deleting a sales employee.

AFTER triggers are generally used to update columns in linked tables that depend on the row being inserted, updated or deleted for their values. For example, the PERCENT_CHANGE column in the SALARY_HISTORY table is maintained using an AFTER UPDATE trigger on the EMPLOYEE table.

To summarize: Use BEFORE until all data manipulation operations have been completed. The EMPLOYEE database trigger SET_CUST_NO is an example of a BEFORE INSERT, as a new customer number is generated before the data set has been inserted.

When manipulation of own data has been concluded, then use an AFTER trigger. The EMPLOYEE database trigger SAVE_SALARY_CHANGE is an example of AFTER UPDATE, as the changes to the data have already been completed.

INSERT, UPDATE, DELETE

A trigger must be defined to fire on one of the keywords INSERT, UPDATE or DELETE.

- An INSERT trigger fires before or after a row is inserted into the table.
- An UPDATE trigger fires when a row is modified in the table.

- A DELETE trigger fires when a row is deleted from the table.

If the same trigger needs to fire on more than one operation, separate triggers need to be defined for each operation. If they share any processing, this code should be placed in a stored procedure and the procedure called from each trigger. Although - since Firebird 1.5 triggers are no longer restricted to either insert or update or delete actions, but only one trigger needs to be created for all of these. For example:


```
AS
   BEGIN
      if (new.bez<>'')
      then new.bez=upper(new.bez);
   END
```

The '' UPPER applies to INSERT and UPDATE operations.

Please note that special characters, such as German umlauts, are not recognized and altered to upper case, as the character is treated technically as a special character, and not an alphabetical letter.

For further information regarding NEW variables, please refer to NEW and OLD context variables.

NEW and OLD Context Variables

In triggers (but not in stored procedures), InterBase/Firebird provides two context variables that maintain information about the row being inserted, updated or deleted:

- OLD.columnName refers to the current or previous values in a row being updated or deleted. It is not relevant for INSERT triggers.
- NEW.columnName refers to the new values in a row being inserted or updated. It is not relevant for DELETE triggers.

Using the old and new values you can easily create history records, calculate the amount or percentage of change in a numeric value, find records in another table that match either the old or new value or do pretty well anything else you can think of.

It is possible to read to or write from these trigger variables.

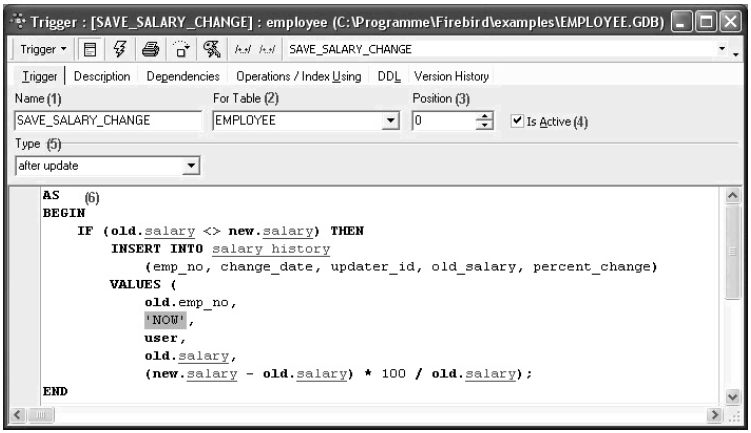
3.6.2 New Trigger

There are numerous ways to create a trigger in IBExpert.

1. Using the menu item Database / New Trigger or the respective icon on the New Database Object toolbar.

2. From the DB Explorer by right-clicking on the highlighted trigger branch of the relevant connected database (or key combination [Ctrl + N]).

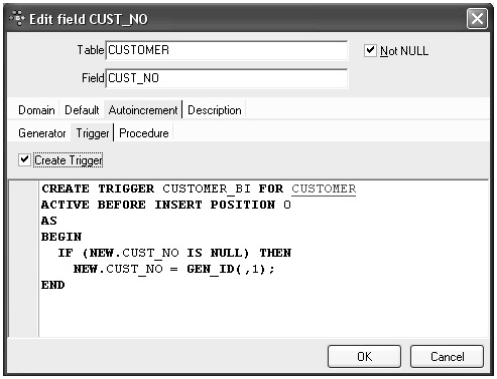
Both these options open the Trigger Editor:



The Trigger Editor's first page allows the **(1)** trigger name, **(2)** table or view name, **(3)** position, **(4)** active/inactive, and **(5)** trigger type to be specified simply and quickly, with the aid of pull-down lists, provided the lazy mode has been switched on. The trigger body **(6)** can be completed in the SQL window.

3. A trigger can also be created in the Table Editor or View Editor, on the Triggers page by selecting the desired BEFORE/AFTER operation and using the mouse right-click menu item New Trigger. This opens the New Trigger Editor shown above.

4. Or in the Field Editor on the Autoincrement page. For example, a trigger text for a new generator can be simply and quickly created using the Edit Field / Autoinc, Create Generator and then Create Trigger, .



For those preferring direct SQL input, the `CREATE TRIGGER` statement has the following syntax:

```
CREATE TRIGGER <trigger_name>
FOR <table_name>
<keywords_for_trigger_type>
AS
```

```
<local_variable_declarations>
BEGIN
<body_of_trigger>
END
```

The trigger name needs to be unique within the database, and follow the InterBase/Firebird naming conventions used for columns, tables, views and procedures.

Triggers can only be defined for a single table or updateable view. Triggers that should apply to multiple tables need to be called using a stored procedure. This can be done simply by creating a stored procedure which refers to the trigger.

Triggers fire when a row-based operation takes place on the named table or view.

Position:

255 positions are allowed per table, (starting at 0, up to 254). Several triggers may share a single position.

Trigger Types:

Trigger status: `ACTIVE` or `INACTIVE`
Trigger position: `BEFORE` or `AFTER`
Operation type: `INSERT` or `UPDATE` or `DELETE`

Please refer to the links for further information regarding these types.

Local Variable Declarations:

Triggers use the same extensions to SQL that InterBase/Firebird provides for stored procedures. Therefore, the following statements are also valid for triggers:

```
DECLARE VARIABLE
BEGIN … END
SELECT … INTO : variable_list
Variable = Expression
/* comments */
EXECUTE PROCEDURE
FOR select DO …
IF condition THEN … ELSE …
WHILE condition DO …
```

As with stored procedures, the `CREATE TRIGGER` statement includes SQL statements that are conceptually nested inside this statement. In order for InterBase/Firebird to correctly parse and interpret a trigger, the database software needs a way to terminate the `CREATE TRIGGER` that is different from the way the statements inside the `CREATE TRIGGER` are terminated. This can be done using the `SET TERM` statement.

Since IBExpert version 2005.03.12 there is added support for following Firebird 2 features:

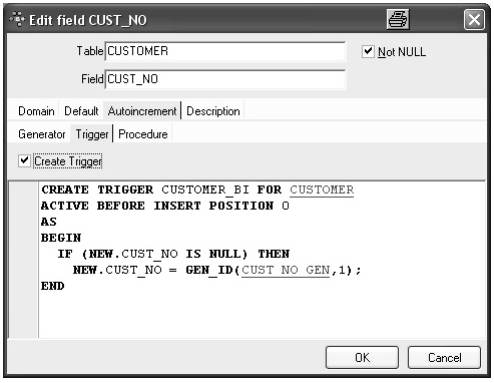
- `DECLARE <cursor_name> CURSOR FOR ...`

- OPEN <cursor_name>
- FETCH <cursor_name> INTO ...
- CLOSE <cursor_name>
- LEAVE <label>
- NEXT VALUE FOR <generator>

Don't forget to finally compile the new trigger using the respective toolbar icon or [F9], and, if desired, autogrant privileges, again using the respective toolbar icon or key combination [Ctrl + F8].

Create a trigger for a generator

Generally a generator is used to determine unique identification numbers for primary keys. A BEFORE INSERT trigger can be defined for this to generate a new ID, increasing the current value using the GEN_ID() function, and automatically entering it in the respective table field.



The above illustrates the Field Editor, started from the Table Editor.

Create a trigger for a view

It is possible to create a trigger for a view directly in the View Editor on the Trigger page. This is particularly interesting for read-only views.

For example, BEFORE INSERT, insert into Table1 new_fields and table2 new_data for fields. BEFORE UPDATES and BEFORE DELETE triggers should also be added, in order to distribute the data manipulations made in the view into the respective base tables.

3.6.3 Trigger Editor

The Trigger Editor can be started using the Database / New Trigger menu item; from the DB Explorer, using the right mouse-click menu or double-clicking on an existing trigger, or alternatively directly from the View or Table Editor / Triggers tab.

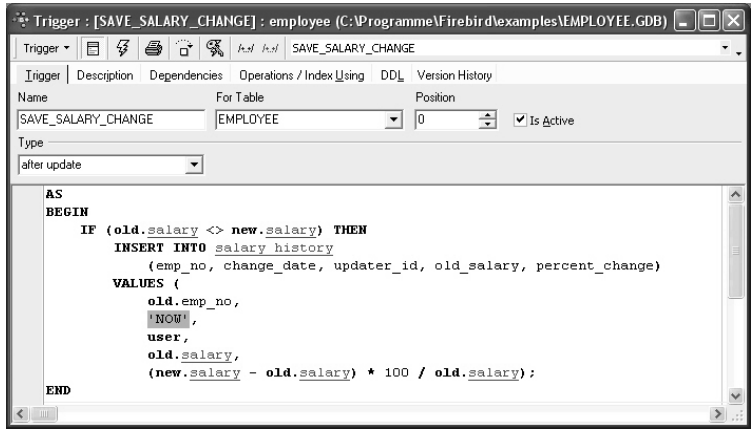
Please refer to New Trigger when creating a trigger for the first time.

The Trigger Editor has its own toolbar (see Trigger Editor toolbar) and offers the following options:

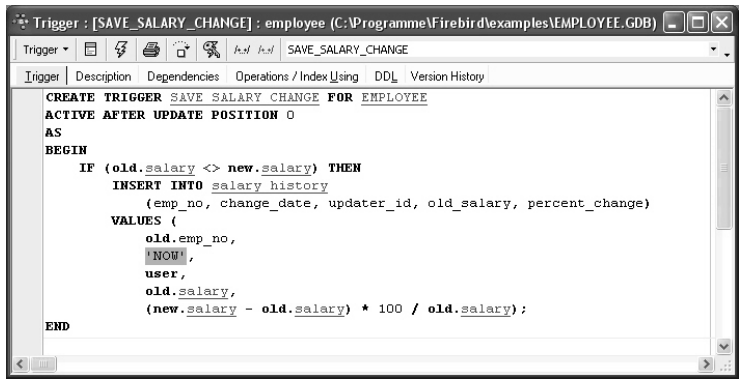
- Trigger
- Description
- Dependencies
- Operations/Index Using
- DDL
- Version History

Trigger

The Trigger Editor's first page allows the trigger name, table or view name, position, active/inactive, and trigger type to be specified simply and quickly, with the aid of pull-down lists, provided the lazy mode has been switched on:



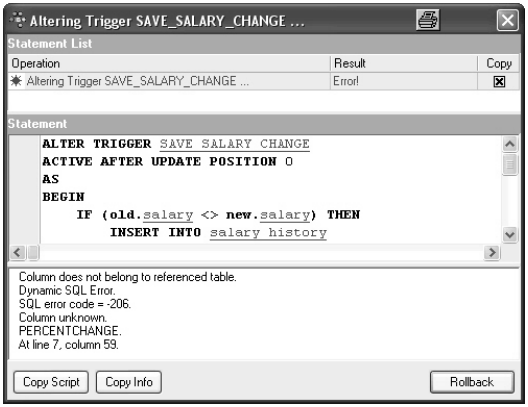
If this is switched off, the above information all needs to be specified in the SQL window:



The SQL window provides a template, for both standard (for the whole trigger) and lazy mode, where the trigger body can be input. These templates can be altered if wished, using the IBExpert menu item Options / General Templates / New Trigger.

As with all SQL input windows, the SQL Editor Menu can be called using the right mouse button. The keyboard shortcuts available in the SQL Editor are also available here.

When the trigger or trigger alterations are complete, it can be compiled using the respective icon or [Ctrl + F9]. If errors are found, click YES when the Compile Anyway query appears, to produce an SQL error script (below the trigger text), to detect the error source.



If the problem is more complicated, the options Copy Script or Copy Info can be used before finally rolling back the trigger.

The Trigger Editor also has its own Debug Trigger icon. For more information regarding this, please refer to Debug Procedure or Trigger.

Description

Please refer to Table Editor / Description.

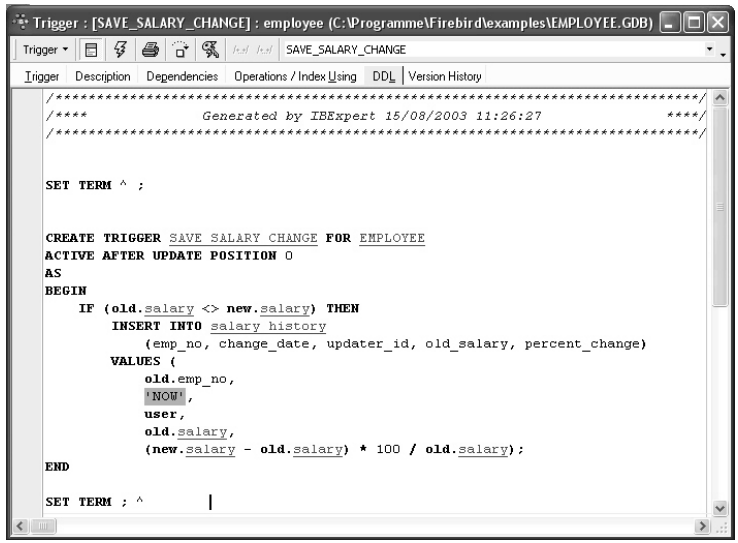
Dependencies

Please refer to Table Editor / Dependencies.

Operations/Index Using

Please refer to Procedure Editor / Operations / Index Using.

DDL



Please refer to Table Editor / DDL.

Version History

Please refer to View Editor / Version History.

Comment Trigger Body/Uncomment Trigger Body

It certain situations it may be necessary to disable certain commands or parts of trigger code. This can be easily done temporarily, without it being necessary to delete these commands.

Simply select the rows concerned in the SQL Editor, and select either the editor toolbar icons:

the right mouse button menu item Comment Selected, or key combination [Ctrl + Alt + .]. This alters command rows to comments. The commented text can be reinstated as SQL text by using Uncomment Procedure icon (above), the right mouse button menu item Uncomment Selected, or [Ctrl+ Alt + ,].

It can not only be used to add comments and documentary notes to more complex stored procedures and triggers; but also to factor out selected parts of code during the testing phase, or even for customer applications, where certain features are not currently needed but may be required at a future date. The code can be reinstated by simply uncommenting as and when required.

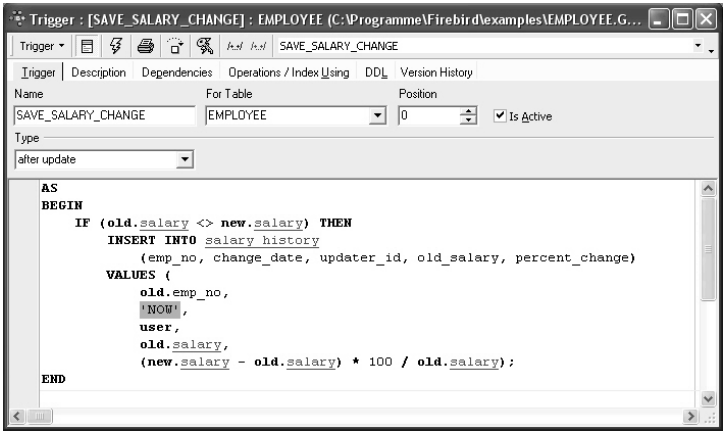
3.6.4 Alter Trigger

Both the trigger header and the trigger body may be altered.

The trigger header may be activated or deactivated, or its position changed (in relation to other triggers).

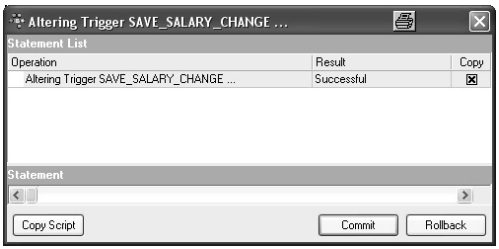
If the trigger body needs to be altered, there is no need to make any alterations to the header, unless you wish to of course! Although in this case, it would probably make more sense to drop the trigger and create a new one. Any amendments to the trigger body override the original contents.

Triggers can easily be altered in the DB Explorer's Trigger Editor, opened either by double-clicking on the trigger name, or right-clicking and selecting Edit Trigger [Ctrl + O]. The header information can be changed as wished using the pull-down lists to alter position, active/non-active and type:



(Image shows lazy mode). The body text may be altered in the SQL panel as wished.

Finally the revised trigger needs to be compiled and committed, for the alterations to become effective.



The SQL syntax for alterations to the trigger header is as follows:

```
ALTER TRIGGER <trigger_name> INACTIVE | ACTIVE
```

```
ALTER TRIGGER <trigger_name> POSITION n
```

where n is the new position number. Or to alter the trigger body:

```
ALTER TRIGGER <trigger_name>
AS
BEGIN
   <new_trigger_body>
END
```

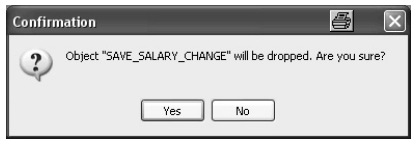
A trigger can only be altered by the database owner or by the SYSDBA.

3.6.5 Drop Trigger/Delete Trigger

A trigger can only be dropped if other users are not performing any changes to any tables which may relate to the specified trigger, at the time of deletion.

In IBExpert, a trigger can be dropped from the DB Explorer by selecting the trigger to be deleted and using the right-click menu item Drop Trigger or [Ctrl + Del].

IBExpert asks for confirmation



before finally dropping.

For those preferring to use SQL, the syntax is as follows:

```
DROP TRIGGER <trigger_name>
```

An alternative solution to dropping triggers is to alter them to the `INACTIVE` status. That way they are left in the database, but disabled from firing, just in case they might be needed after all at a later date.

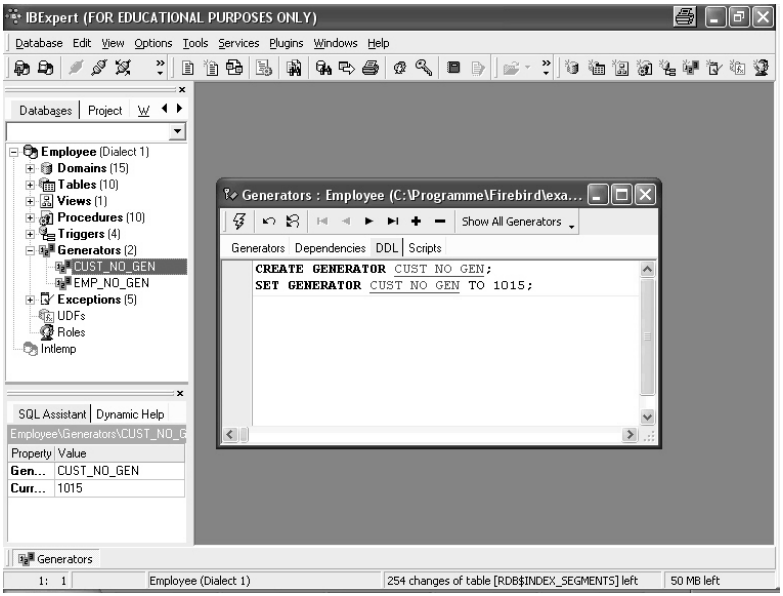
A trigger can only be dropped by the database owner or the SYSDBA.

3.7 Generator

Generators are automatic sequential counters, spanning the whole database. They are necessary because all operations in InterBase/Firebird are subject to transaction control.

A generator is a database object and is part of the database\'s metadata. It is a sequential number, incorporating a whole-numbered 64 bit value integer since InterBase 6/Firebird (in earlier versions a 32 bit value integer), that can automatically be inserted into a column. It is often used to ensure a unique value in an internal primary key.

Generators are the only transaction-independent part of InterBase/Firebird. For each operation a new number is generated, regardless whether this transaction is ultimately committed or rolled back (this consequently leads to \"missing numbers\"). Therefore generators are best suited for automatic internal sequential numbering for internal primary keys.



Generators can be created either directly in the SQL Editor or using the DB Explorer (refer to New Generator for details).

Generally a generator is used to determine unique identification numbers for primary keys. A trigger can be defined for this, which increases the current value using the GEN_ID( ) function, and automatically enters it in the respective table field. Please refer to "create a trigger for a generator" for more information. A generator can also be called from a stored procedure or an application.

A database can contain any number of generators. Although up until the most recent InterBase version the number of generators was limited to one data page. One generator uses 8 bytes, which means approximately 115 generators fit onto one page (at 1K). This limitation has been solved in the most recent InterBase version.

The current generator value of existing generators is not stored in a table but on its own system data pages, as the table contents are subject to transactional changes. The generator value is also secured when backing up.

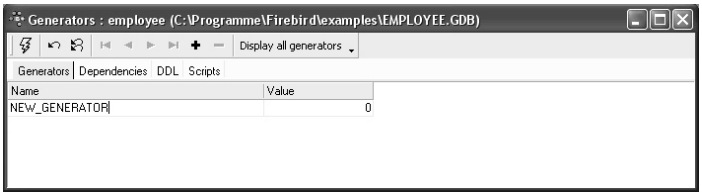
Generators are database objects and are part of the database\'s metadata, and can be created, modified and dropped as all other InterBase/Firebird objects in the IBExpert DB Explorer.

3.7.1 New Generator

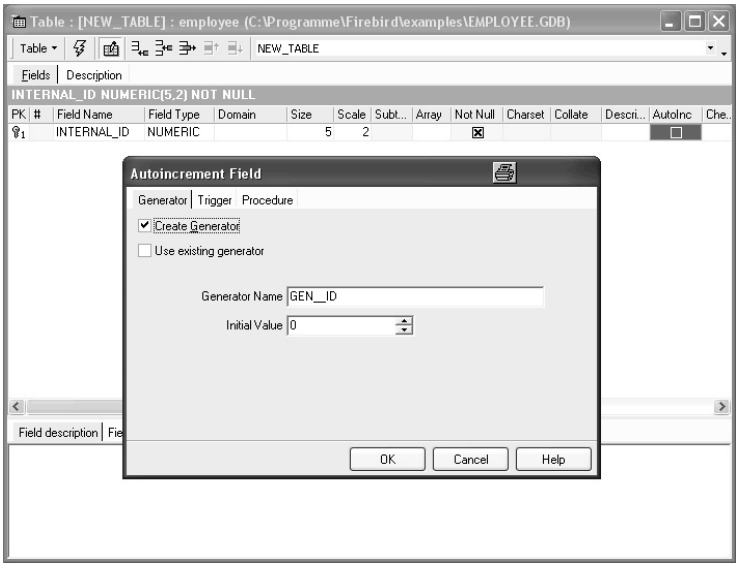
A new generator can be created in a connected database in a number of ways:

1. By using the menu item Database / New Generator, the respective icon in the New Database Object toolbar, or using the DB Explorer right mouse button (or key combi-

nation [Ctrl + N]), when the generator heading of the relevant connected database is highlighted, to start the New Generator Editor:



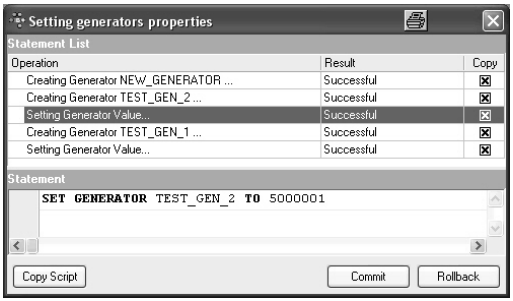
2. Alternatively, a new generator can be created in the DB Explorer Table Editor on the Fields page by double-clicking (or using the space bar when inserting a new field) to check the Autoinc box:



3. Or in the Field Editor under Autoincrement (started by double-clicking on an existing `INTEGER` or `SMALLINT` field in the Table Editor).

4. Or directly in the IBExpert SQL Editor, and then saved as a generator.

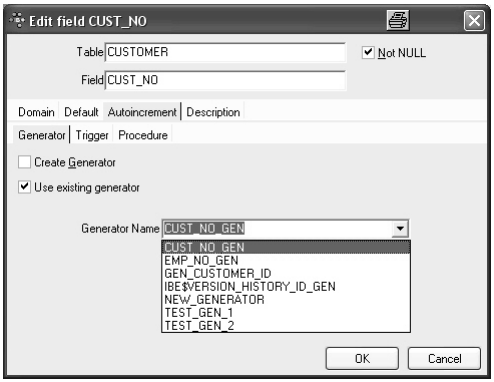
Using the Generator Editor the new generator name simply needs to be specified along with the initial generator value. Several generators can be created in the Generator Editor and compiled simultaneously:



Using the Display all Generators button on the Generator Editor toolbar, all generators for the database can be listed and an existing generator selected. (For internal numbering purposes, the same generator may be used on several fields, for example all internal primary key IDs, within the database.)

Using the Autoinc page in the Table and Field Editors, the Create Generator box simply needs to be checked, and the name and starting value defined.

It is also possible to select an existing generator for the specified field here (simply click Use Existing Generator and select from the pull-down list):



For those preferring direct SQL input, the syntax is as follows:

```
CREATE GENERATOR <Generator_Name>;
```

This statement also sets the initial generator value to zero. To establish a different starting value, use the `SET GENERATOR` statement, for example:

```
SET GENERATOR <Generator_Name> TO n;
```

where n is the initial generator value. `SET GENERATOR` can also be used to reset an existing generator's value. This however requires care, as usually the column(s) that receives the generator value is/are defined to be unique. For example, you would not normally reset customer IDs except under unusual and controlled circumstances.

To increment the generator use the STEP_VALUE parameter (can be positive or negative):

```
GEN_ID(<Generator_Name>, STEP_VALUE)
```

If this parameter is not used, the default STEP_VALUE with an increment of 1 applies.

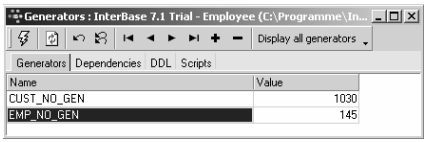
3.7.2 Generator Editor

The Generator Editor can be started using the Database / New Generator menu item; from the DB Explorer, using the right mouse-click menu or double-clicking on an existing generator; or directly from the Field or Table Editor / Autoincrement.

Please refer to New Generator when creating a generator for the first time.

The Generator Editor has its own toolbar (see Generator Editor toolbar) and offers the following options:

- Generators
- Dependencies
- DDL
- Scripts



Generators

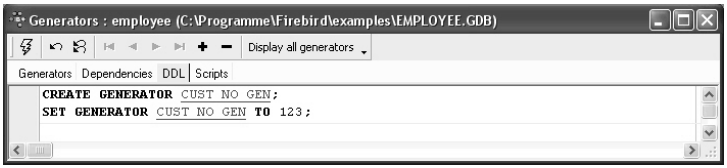
Here it is possible to create new generators, select an existing generator, and alter a generator.

Please refer to New Generator or Alter Generator for details.

Dependencies

Please refer to Table Editor / Dependencies.

DDL



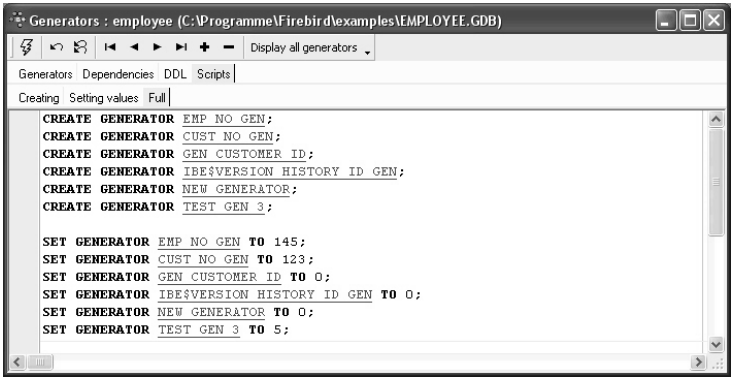
Please refer to Table Editor / DDL.

Scripts

Creating - displays the CREATE GENERATOR statement for the generator selected on the Generators page. If all generators are displayed on the Generator page (Display All Generators button), all corresponding CREATE statements appear on this page.

Setting Values - displays the SET GENERATOR statement for the generator selected on the Generators page. Again, if all generators are displayed on the Generator page (Display All Generators button), all SET statements appear on this page.

Full - displays the full SQL text for the generator selected on the Generators page (or all generators).



Please note that the Scripts page is for display only. It is not possible to make any amendments on this page.

3.7.3 Alter Generator

A generator may be altered to specify a new value. The value of a generator can be changed as often as wished.

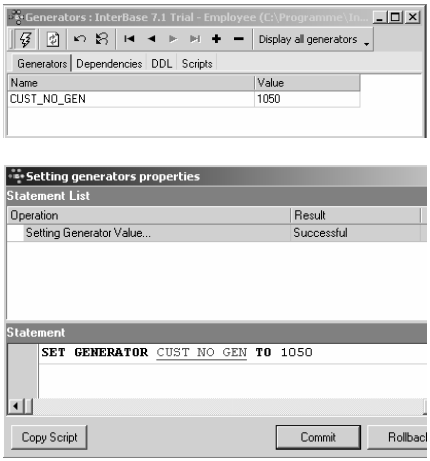
This can be performed in IBExpert using the DB Explorer's Generator Editor, opened either by double-clicking on the generator name, or right-clicking and selecting Edit Generator [Ctrl + O]. Simply enter the new figure in the Value column, compile and commit.

The SQL syntax for altering a generator is as follows:

```
SET GENERATOR <generator_name> TO n
```

where n is the new value. This new value is immediately effective.

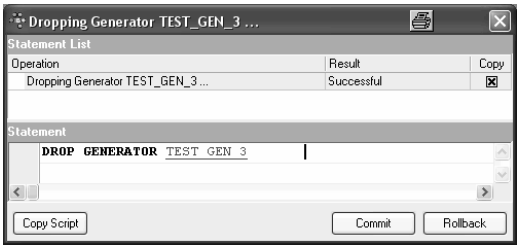
Please refer to the SET GENERATOR statement for further information.



3.7.4 Drop Generator/Delete Generator

In IBExpert, a generator can be dropped from the DB Explorer by selecting the generator to be deleted and using the '-' icon on the Generator Editor toolbar or [Shift + Del].

IBExpert asks for confirmation and displays the SQL statement:



before finally dropping when the statement is committed.

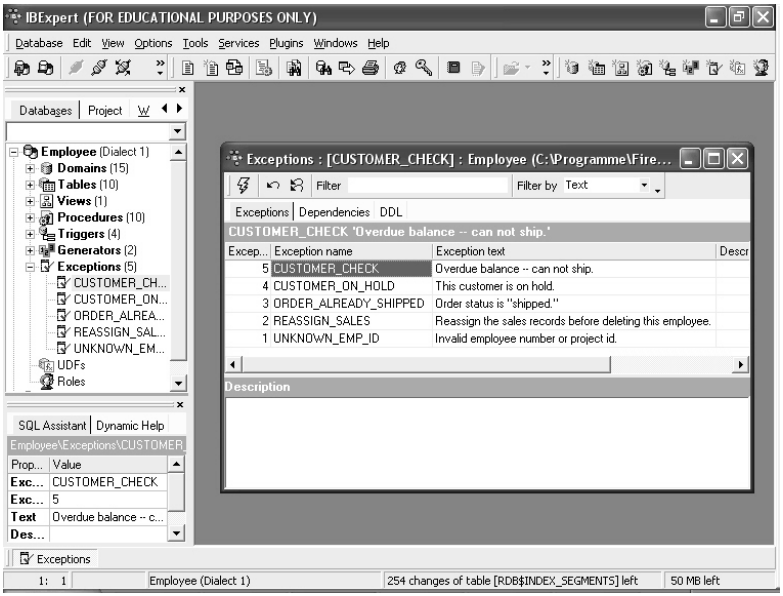
For those preferring to use SQL, the syntax is as follows:

```
DROP GENERATOR <generator_name>;
```

3.8 Exception

Exceptions are user-defined named error messages, written specifically for a database and stored in that database for use in stored procedures and triggers.

If it is ascertained in a trigger that the value in a table is incorrect, the exception is fired. This leads to a rollback of the total transaction that the client application is attempting to commit. Exceptions can be interleaved.



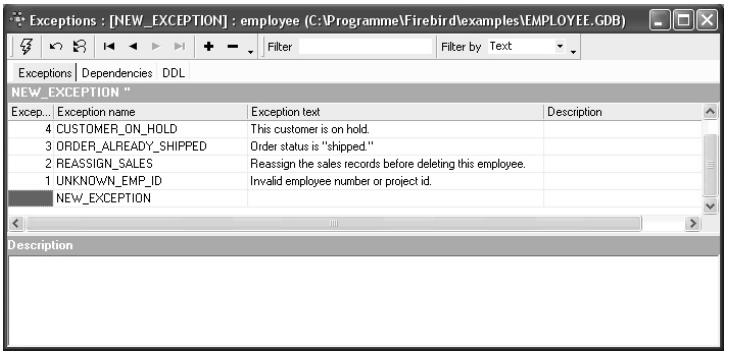
They can be shared among the different modules of an application, and even among different applications sharing a database. They provide a simple way to standardize the handling of preprogrammed input errors.

Exceptions are typically used to implement program logic, for example, you do not wish a user to sell an item in stock, which has already been reserved by another user for their customer.

Exceptions are database objects and are part of the database\'s metadata, and can be created, modified and dropped as all other InterBase/Firebird objects in the IBExpert DB Explorer.

3.8.1 New Exception/Exception Editor

A new exception can be created in a connected database either by using the menu item Database / New Exception, the respective icon in the New Database Object toolbar, or using the DB Explorer right-click menu (or key combination [Ctrl + N]), when the exception heading of the relevant connected database is highlighted. A New Exception dialog appears, with its own toolbar:



Alternatively, a new exception can be created directly in the IBExpert SQL Editor, using the following statement:

```
CREATE EXCEPTION <Exception_Name>
"Exception_Text";
```

The Exception Editor can be opened directly from the DB Explorer by double-clicking on any existing exception name. It can also be started directly from any procedure or trigger containing an exception, simply by double-clicking on the exception name in the SQL text on the Procedure Editor's Edit page, or the Trigger Editor's Triggers page.

Exceptions

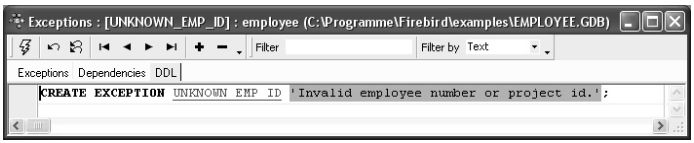
The new exception name can be added to the list displaying all exceptions for the active database, and the exception text message entered. Please be careful when using special characters! Especially when using older versions of InterBase, it is preferable to abstain from using any special characters. With the newer versions, there should not be any problems, provided the correct character set has been specified.

The exception ID is automatically assigned by the database, when the exception is committed.

Dependencies

Please refer to Table Editor / Dependencies.

DDL



Please refer to Table Editor / DDL.

3.8.2 Raising an Exception

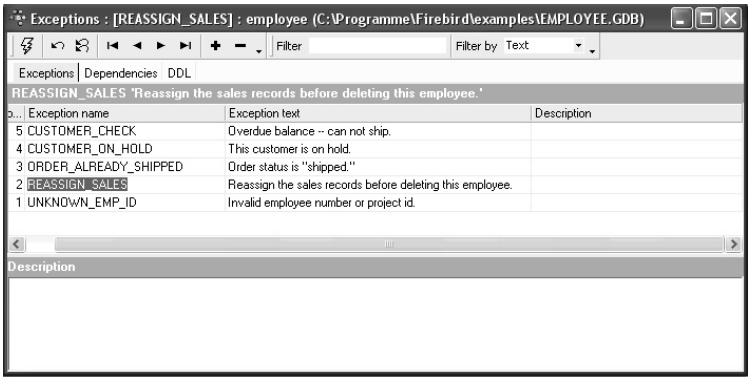
The EXCEPTION statement is used to notify a calling application of an exception. The calling application can be a trigger, a stored procedure, or another program. To raise an exception in a trigger or stored procedure use the EXCEPTION keyword:

```
EXCEPTION <Exception_Name>;
```

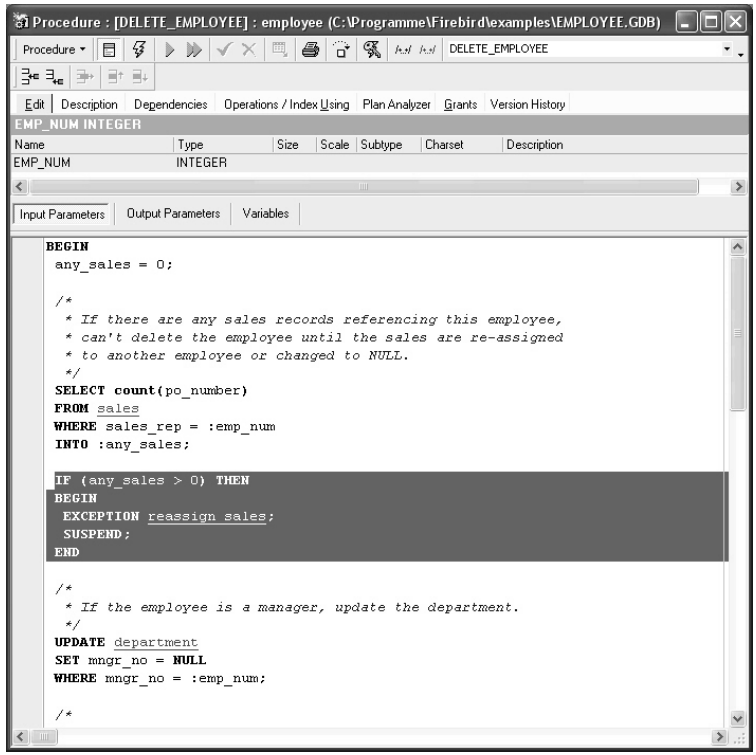
When an exception is raised, the following takes place:

- The exception terminates the trigger or procedure
- Any statements in the trigger or stored procedure that follow the EXCEPTION statement are not executed. In the case of a BEFORE trigger the update that fired the trigger is aborted.
- The trigger or procedure returns an error message to the calling application.

An example of an exception raised in a procedure can be found in the EMPLOYEE database. The exception REASSIGN_SALES was first created:



and then incorporated into the DELETE_EMPLOYEE procedure:



3.8.3 Alter Exception

Exceptions can be altered directly in the Exceptions Editor, started by double-clicking directly on the exception name in the DB Explorer. Alternatively use the DB Explorer's right mouse-click menu item Edit Exception or key combination [Ctrl + O].

The Exception Editor appears, where changes to the exception name and exception text can be made as wished. Changes to exception texts may be made even if other objects depend on them, however not the exception name.

The SQL syntax is:

```
ALTER EXCEPTION <exception_name>
'New Exception Text';
```

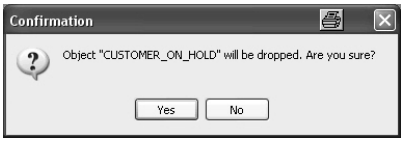
An exception can only be altered by the original creator or by the SYSDBA user.

3.8.4 Drop Exception/Delete Exception

An exception may not be dropped if it is used by other procedures or triggers, until the dependency is removed. Any such dependencies are listed under the Exception Editor / Dependencies page, where they can be directly removed, if wished.

To drop an exception use the DB Explorer right mouse-click menu item Drop Exception... or [Ctrl + Del].

IBExpert asks for confirmation:



before finally dropping the exception. Once dropped, it cannot be retrieved.

Using SQL the syntax is:

```
DROP EXCEPTION <exception_name>;
```

An exception can only be dropped by its creator, the database owner or the SYSDBA.

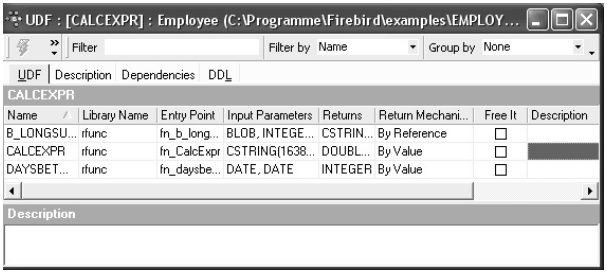
3.9 User–Defined Function (UDF)

A user-defined function (UDF) is used to perform tasks that Firebird/InterBase can't.

It can be described as an external database function written entirely in another language, such as C++ or Pascal, to perform data manipulation tasks not directly supported by InterBase/Firebird.

UDFs can be called from InterBase/Firebird and executed on the server. These functions can exist on their own or be collected into libraries. UDFs offer the possibility to create your own functions (such as SUBSTR) and integrate them in the database itself. Each UDF is arranged as a function, belonging to a DLL (Linux: .SO). Thus one dynamically loaded library consists of at least one function.

UDFs can be incorporated into the database using the IBExpert DB Explorer, IBExpert SQL Editor, or IBExpert Script Executive.



The IBExpert UDF Editor displays those UDFs inserted into the list, by double-clicking on the UDF name in the DB Explorer, or alternatively using the navigation icons in the editor toolbar to insert single or all UDFs. The grid display can also be filtered or grouped if wished. The grid displays key information, including name, library, entry point, input parameters, returns, return mechanism (pull-down list of options),

whether freed (checkbox), and description. Further information is displayed on the Description, Dependencies and DDL pages.

UDF definitions are database dependent and not server dependent, i.e. they need to be registered for each database individually. Since InterBase 6/Firebird, the libraries need to be stored in the InterBase/Firebird UDF folder. This is not critical when working with older InterBase versions.

Please refer to the `DECLARE EXTERNAL FUNCTION` statement for details of incorporating UDFs in InterBase/Firebird.

It is important to note that the majority of UDFs, when used in a `WHERE` condition, prevent indices being used during execution.

An ideal example of a UDF library is RFunc (written in C++) containing over 80 UDFs (although some of these are only applicable for older InterBase versions or for different SQL dialects). It is available for both Windows and Linux platforms in English and Russian and can be downloaded free of charge from www.ibexpert.com/download. FreeudfLib is an example of a UDF library written in Delphi, and can also be downloaded from this link.

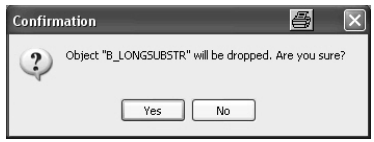
3.9.1 Drop External Function/Drop UDF

The `DROP EXTERNAL FUNCTION` command removes the declaration of the UDF, specified by an additional parameter, from the database.

The dropped function can no longer be reached by the database, as the relevant reference to the UDF library is deleted. However the UDF still exists in the UDF library, so that it can still be used by other databases.

In IBExpert, a UDF can be dropped from the DB Explorer by selecting the UDF to be deleted and using the right-click menu item Drop UDF or [Ctrl + Del].

IBExpert asks for confirmation



before finally dropping.

The SQL syntax is:

```
DROP EXTERNAL FUNCTION <external_function_name>
```

The declaration of a UDF can only be dropped by the database owner or the SYSDBA.

3.9.2 RFunc

RFunc is a UDF library containing over 80 UDFs (although some of these are only applicable for older InterBase versions or for different SQL dialects). It is available for both Windows and Linux platforms in English and Russian. It can be downloaded free of charge from http://www.ibexpert.com/download/udf/. The most up-to-date version of this library can found at http://rfunc.sourceforge.net/.

It represents a set of user\'s (UDF) string, bit, numerical functions, and can also be used for operations with `dates&time` and blobs. Also contains `PARSER`, i.e. calculator of expressions.

InterBase 4.2, 5.x, 6.x, 7.0 (Windows 9x, NT, 2000) and InterBase 5.x, 6.x, 7.0 (Linux) or Firebird are supported. The library is written in C++ and is delivered with source codes.

RFunc installation

The ZIP-file should be selected (Windows or Linux; English or Russian) and downloaded.

Windows Installation:

1. The `RFUNC.DLL` file needs to be copied into a folder:

Variant 1: `IB_path\bin` (for IB6 - `IB_path\UDF`), where `IB_path` is the path to a folder, in which InterBase/Firebird is installed (recommended).

Variant 2: `Windows\System` (for Windows 9x) or `WinNT\System32` (Wpath>`IB_path\bin` (for IB6 - `IB_path\UDF`), where `IB_path` is the path to a folder, in which InterBase/Firebird is installed (recommended).

Variant 2: `Windows\System` (for Windows 9x) or `WinNT\System32` (Windows NT, 2k).

2. only for IB 5.x: copy `ib_util.dll` file from `\Lib` to face="Courier New">path>`\Lib` to `\Bin`.

If several versions of InterBase servers are installed on one computer, it is necessary to use the RFunc library appropriate to the installed client IB (`GDS32.DLL`).

It is recommended before starting the InterBase/Firebird server to substitute `GDS32.DLL` appropriate to the version of the server.

For Linux:

IB 5.x

Variant 1: Copy the RFunc file into directory `/usr/lib`.

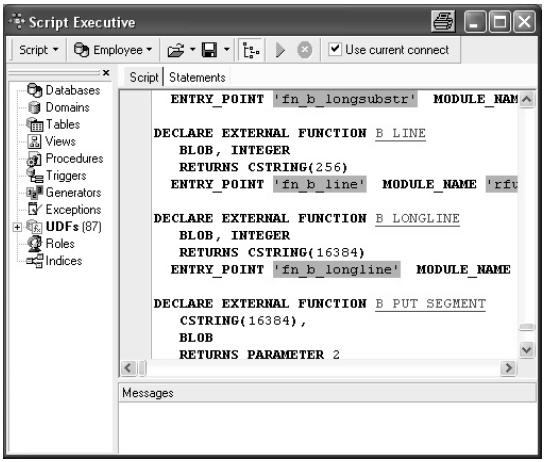
Variant 2: Copy the RFunc file into any directory, for example, `/home/rFunc`. Create the reference to the library by using the `\ln -s /home/rFunc/rfunc`

/usr/lib/rfunc\ command. The user should own the right to create references in the directory /usr/lib.

IB6-7 und Firebird

Copy the RFunc file into directory \UDF.

The rfuncx.sql (x = InterBase version) script should then be copied into the IBExpert Script Executive (found in the Tools menu), and executed [F9]. A database connection must exist, as UDF libraries need to be registered for each database (i.e. they are database dependent and not server dependent).



It is then necessary to disconnect and reconnect to the database so that the full list of RFunc UDFs can be viewed in the DB Explorer under the DB object branch "UDF".

3.9.3 FreeUDFLib

FreeUDFLib is a free UDF library (October 1998) containing many useful UDFs for use with InterBase 4.2 and 5.0 under the Win32 platforms (unfortunately no UNIX support with this). It is written entirely in Delphi and all source code is provided.

It can be downloaded free of charge from http://www.ibexpert.com/download/udf/.

Everything in this release is completely free. However, it's not a PUBLIC DOMAIN. Please refer to the license.txt, included in the ZIP file for more information on licensing.

FreeUDFLib installation

After unzipping FreeUDFLib.zip, copy FreeUDFLib.dll to the InterBase/Firebird bin or udf directory, for example: C:\Program Files\InterBase Corp\InterBase\bin, C:\Program Files\Borland\InterBase\udf\bin or C:\Program Files\Firebird\udf\bin.

The ext_funcs.sql script should then be copied into the IBExpert Script Executive (found in the Tools menu), and executed using [F9]. A database connection must exist, as UDF libraries need to be registered for each database (i.e. they are database-dependent and not server-dependent). If necessary, use the Script Executive menu item Add CONNECT statement to connect to the desired database, before executing.

It is then necessary to disconnect and reconnect to the database so that the full list of FreeUDF external functions can be viewed in the DB Explorer under the DB object branch "UDF".

3.9.4 FreeAdhocUDF

FreeAdhocUDF (copyright (c) 2004 adhoc dataservice GmbH Peter Mandrella) is a free version of FreeUDFLib, which can be used on both Windows and Linux platforms. It comes complete with source code. It is based on FreeUDFLibC (copyright (c) 1999 Gregory Deatz) and was altered in 2001 by ADITO Software GmbH Robert Loipfinger.

FreeAdhocUDF can be downloaded from
http://www.ibexpert.com/download/udf/FreeAdhocUDF.zip

Note: to use with dialect 3 (timestamp-field is TIMESTAMP) length of cstring = 8190 because then you can use one UDF four times in a piped string 4 x 8190 = 32760 < 32762 (32 bit - 6 bit internal use).

Modification Objectives:

1. Compatibility between Windows and Linux, allowing easy database migration by simply performing a backup and restore. This includes the same function names as well as the same entrypoints, and the functions deliver the same results. Under Windows the modul_name is "FreeAdhocUDF.dll", under Linux "FreeAdhocUDF". In the SQL definition the modul_name is always "FreeAdhoc UDF" - Windows searches automatically for the *.dll. Under Linux both the modul_name and the lib name are case-sensitive.

2. Corrections have also been made where Windows produced a wrong result and Linux a correct one. These corrections are, of course, not in the "old" FreeUDFLib.dll, but were made available in the new FreeAdhocUDF.dll.

3. New useful functions that have not previously been available in FreeUDFLib. These become available following compilation under Windows.

4. Optimization of the C code.

Alterations to FreeUDFLibC

F_AGEINWEEKS

The previous function calculated the interval in days, divided by 7 and then rounded up. In certain cases, for example 20.08.2004 to 21.08.2004, this led to a result of 1.

This function has now been modified to produce the same results as under Windows: as a rule, weeks always begin on Sunday, so, if the date is 6/21/97, (a Saturday), and

your date reference is 6/22/97, a Sunday, AgeInWeeks will return a 1. If the two dates fall in the same week, then it returns 0.

New functions:

F_CDOWLONGLANG(date, language)
de = German
uk = English
us = English
fr = French
it = Italian
es = Spanish
F_CDOWSHORTLANG(date, language)
F_CMONTHLONGLANG(date, language)
F_CMONTHSHORTLANG(date, language)

Functions that still behave differently under Linux to Windows (no modifications have been made due to reasons of compatibility):

F_CDOWLONG German language under Windows, English under Linux
F_CDOWSHORT ditto.
F_CMONTHLONG ditto.
F_CMONTHSHORT ditto.

Functions that have been removed from FreeUDFLibC because they either make no sense or are duplicates:

F_STRPOS
F_FINDWORD
F_FINDWORDINDEX

FreeAdhocUDF installation

After unzipping FreeAdhocUDF.zip, copy FreeAdhocUDF.dll to the InterBase/Firebird bin or udf directory, for example: C:\Program Files\InterBase Corp\InterBase\bin, C:\Program Files\Borland\InterBase\udf\bin or C:\Program Files\Firebird\udf\bin.

The ib_declarations8190dialect1.sql or the ib_declarations8190dialect3.sql script should then be copied into the IBExpert Script Executive (found in the Tools menu), and executed using [F9]. A database connection must exist, as UDF libraries need to be registered for each database (i.e. they are database-dependent and not server-dependent). If necessary, use the Script Executive menu item Add CONNECT statement to connect to the desired database, before executing.

It is then necessary to disconnect and reconnect to the database so that the full list of FreeAdhocUDF external functions can be viewed in the DB Explorer under the DB object branch "UDF".

3.10 Blob Filter

Blob filters are routines for blobs. They translate blob data from one type to another, i.e. they allow the contents of blob subtype X to be displayed as subtype Y or vice versa. These filters are ideal tools for certain binary operations such as the compression and translation of blobs, depending upon the application requirements.

A blob filter is technically similar to a UDF (user-defined function). It hangs itself in the background onto the database engine, and is used for example to compress the blob, or to specify the format such GIF or JPG (dependent upon use with Windows or Apple Mac). The blob filter mechanism relies on knowing what the various subtypes are, to provide its functionality.

Blob filters are written in the same way that UDFs are written, and are generally part of standard libraries, just as UDFs are.

3.10.1 Declaring a blob filter

A blob filter needs to be explicitly declared in the database before it is used. This is done using the keyword `DECLARE FILTER`. First it is necessary to connect to the database using the blob filter, and then issue the statement. The syntax of `DECLARE FILTER` is as follows:

```
DECLARE FILTER <IB/FB_Filter_Name>
<Parameter_List>
INPUT TYPE <Type>
OUPUT TYPE <Type>
ENTRY_POINT <External_Function_Name>
MODULE_NAME <Library_Name>;
```

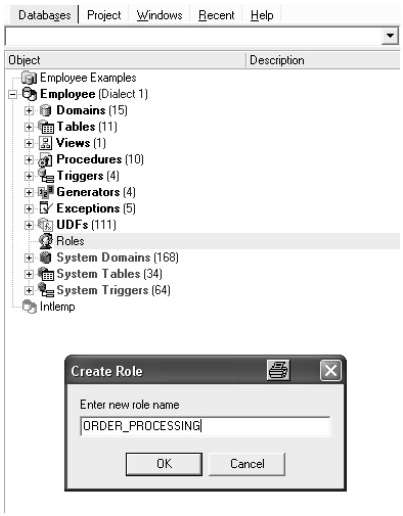
3.10.2 Calling a blob filter

In the same way as UDFs, blob filters can be called from InterBase/Firebird code wherever an InterBase/Firebird built-in function call is used. In order to use the blob filter, invoke the `FILTER` statement when declaring a cursor. Then, whenever InterBase/Firebird uses the cursor, the blob filter is automatically invoked.

3.11 Role

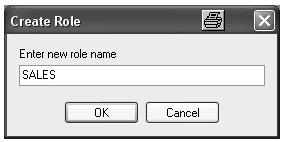
A role is a named group of privileges. It simplifies granting user rights as multiple users can be granted the same role. For example, in a large sales department, all those clerks involved in processing incoming orders could belong to a role "Order Processing".

Should it become necessary to alter the rights of these users, only the role has to be changed.



3.11.1 New Role

A new role can be created in a connected database, either by using the IBExpert menu item Database / New Role, the respective icon in the New Database Object toolbar, or using the DB Explorer right-click menu (or key combination [Ctrl + N]), when the role heading of the relevant connected database is highlighted. A New Role dialog appears:



Simply enter the new role name, and click OK to compile and commit.

Note: when a role with the name SYSDBA is created, no other users (not even the SYSDBA) can access the database.

For those preferring SQL input, the syntax is as follows:

```
CREATE ROLE <Role_Name>;
```

After successfully creating one or more new roles, privileges need to be granted to the role name(s). Please refer to Grant Manager, found in the IBExpert Tools Menu, and GRANT statement for further information.

3.11.2 Alter Role

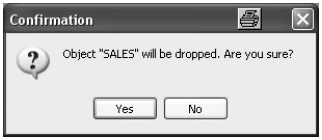
Users and rights may be altered for a role using the IBExpert Grant Manager. This can be started either directly from the DB Explorer using the right-click menu item Edit Role..., or the key combination [Ctrl + O], or using the IBExpert menu item Tools / Grant Manager.

Please refer to Grant Manager for further details.

3.11.3 Drop Role/Delete Role

To drop a role use the DB Explorer right mouse-click menu item Drop Role... (or [Ctrl + Del].

IBExpert asks for confirmation:



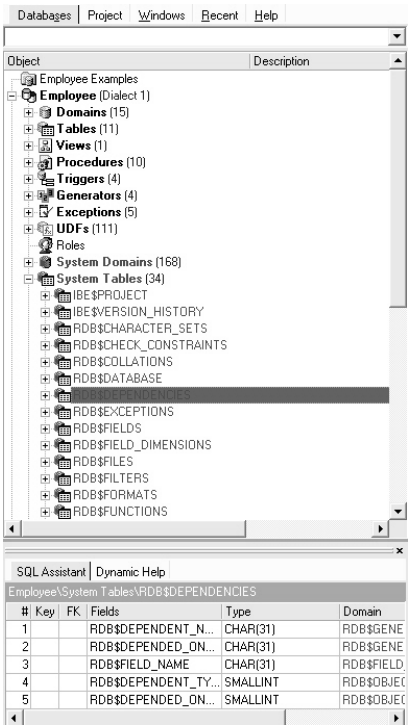
before finally dropping the role. Once dropped, it cannot be retrieved.

Using SQL the syntax is:

```
DROP ROLE <Role_Name>;
```

3.12 System Objects

InterBase/Firebird generates system database objects, and stores its own specific system information about the database objects in system tables. System objects are displayed in the DB Explorer in red, if the system options have been flagged in the Database Registration dialog (called using the right mouse button / Additional / DB Explorer).



System tables and domains contain the prefix RDB$.

A newly created database is almost 0,5 MB large. This is due to the system tables that are automatically generated by InterBase/Firebird when a database is created.

3.13 Text Editor / SQL Code Editor

All Object Editors and SQL Editors include text/SQL input windows. Please refer to the individual subjects, for further information. For example:

- SQL Editor / Edit page,
- Plan Analyzer,
- SQL Editor / Logs,
- Description page,
- Debugger,
- DDL page,
- SQL Monitor,
- Stored Procedure.

Objects may be dragged and dropped from the DB Explorer and SQL Assistant into many of the IBExpert Tools and Services code editor windows, for example, SQL Editor and Query Builder. Since version 2004.2.26.1 this has been greatly improved. When an object node(s) is dragged from the DB Explorer or SQL Assistant, IBExpert will offer various relevant versions of text to be inserted into the Code Editor.

The Text Editor/Code Editor has its own comprehensive right-click context-sensitive menu, the contents of which are described in detail in the SQL Editor / SQL Editor Menu and IBExpert Edit menu.

As with all working areas in IBExpert there are also a number of key combination shortcuts available here in the Text Editor. To view all short cuts or specify your own, use the Localizing Form (a complete list of all shortcuts and operations), opened using [Ctrl + Shift + Alt + L]. For example, a selected block of text can be simply and easily indented using [Ctrl + Shift + U] (decrease indentation using [Ctrl + Shift + I]).

4 IBExpert Edit Menu

The IBExpert Edit menu offers typical manipulation options found in the majority of windows applications. It includes:

- Load and Save to File
- Cut, Copy and Paste
- Find, Search Again and Replace
- Incremental Search
- Print Preview, Print and Page Setup

4.1 Load from File / Save to File

These first two items in the IBExpert Edit menu can also be called using the SQL Editor right-click menu (available in the SQL and Object Editors) or the key combinations [Ctrl + L] or [Ctrl + S] respectively. These items can also be found in the Edit toolbar.

They allow SQL scripts etc. to be loaded or saved to file.

4.2 Cut / Copy / Paste / Select All

These three items can be found in the IBExpert Edit menu and SQL Editor right-click menu (available in the SQL and Object Editors). They can also be executed using the key combinations:

- Cut [Ctrl + X]
- Copy [Ctrl + C]
- Paste [Ctrl + V]

These items can also be found in the Edit toolbar. They allow selected (i.e. marked) text to be cut or copied into the clipboard, and then pasted - either directly in IBExpert or in other applications, such as Windows Editor, Word etc.

The menu item Select All [Ctrl + A] selects a complete text (e.g. SQL script).

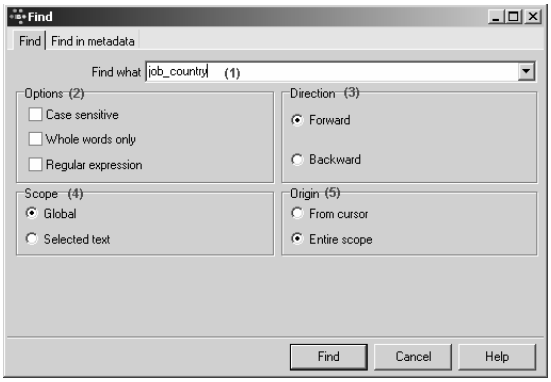
4.3 Find / Search Again / Replace

These three items can be found in the IBExpert Edit menu and SQL Editor right-click menu (available in the SQL and Object Editors). They can be executed using the key combinations:

- Find [Ctrl + F]
- Search Again [F3]
- Replace [Ctrl + R]

or the respective icons in the Edit toolbar.

They are useful for finding individual words/digits or word/digit strings in longer texts or metadata. The Find dialog offers a number of options:



Find page:

(1) Find What: the Find dialog automatically offers the word, where the cursor is currently standing, or a selected text. This can be altered as wished. Previous Find criteria can be selected using the pull-down list.

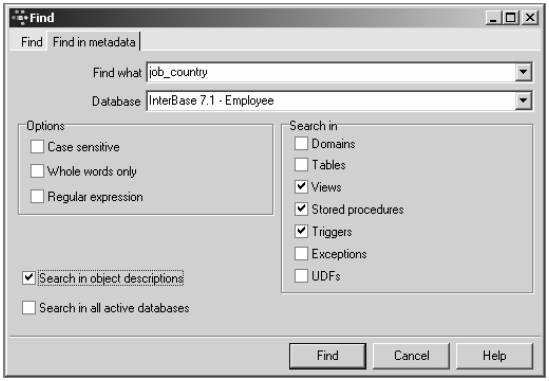
(2) Options: This includes Case Sensitive, Whole Words Only and Regular Expressions (e.g. *,?).

(3) Direction: i.e. forwards or backwards.

(4) Scope: i.e. global or just the selected text.

(5) Origin: From cursor (searches from the cursor position onwards), or entire scope (complete text).

The **Find in Metadata** page offers alternative options:

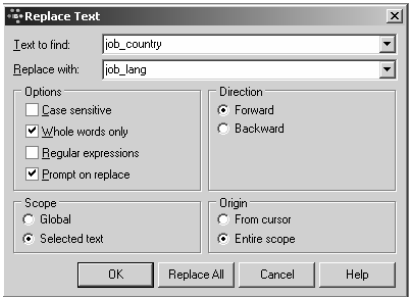


These include database selection (or even a Search in all active databases option using the checkbox at the bottom of the dialog) and, in addition to the options offered on the Find page, a check list of the database object categories to be searched.

New to version 2004.08.05.1 is the checkbox option to search for text or text strings within database object *descriptions*.

Replace

The Replace dialog is similar to the Find page in the Find dialog:



with the following additions:

Replace with: enter the word(s)/number(s) that are to replace the searched for text. Previous Replace entries can be selected using the pull-down list.

The Options check list contains the additional check **Prompt on Replace** (default), allowing the user to check that the found word/number string is correctly replaced.

4.4 Incremental Search

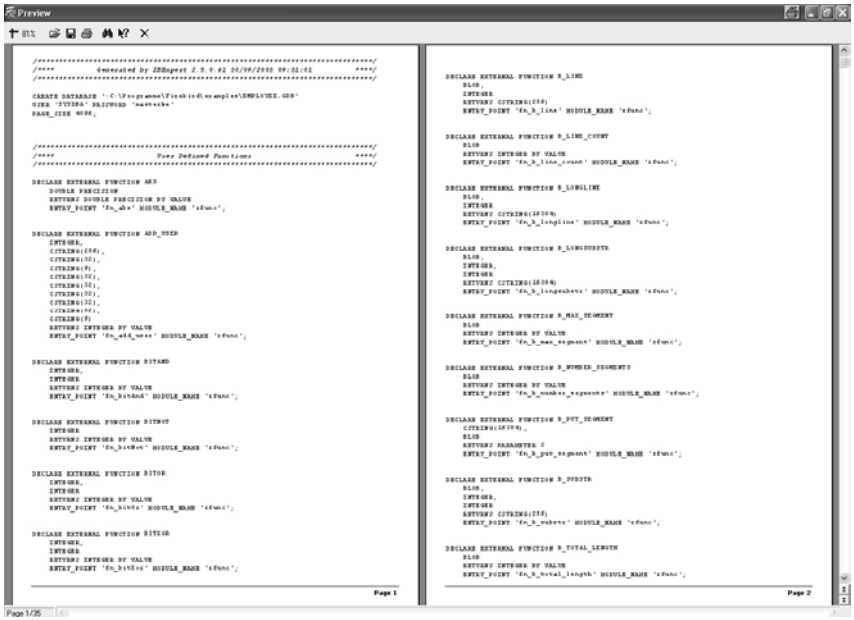
The Incremental Search [Ctrl + F] allows a simple search for individual entries by simply marking the desired column header, clicking the right mouse button menu item Incremental Search [Ctrl + F] and then typing the relevant digits/letters, until the required dataset(s) is/are found. Alternatively, the [Ctrl + Enter] keys can be used to search for the next occurrence of a substring.

This menu item can also be found in the context-sensitive menus in the Table Editor / Data page and in all editors containing an SQL Editor window and right-click SQL Editor Menu.

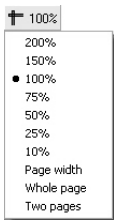
4.5 Print Preview

This item can be found in the IBExpert Edit menu and SQL Editor right-click menu (available in the SQL and Object Editors).

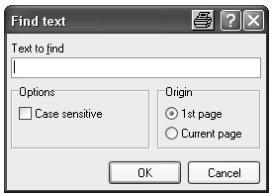
The Print Preview dialog is part of the Fast Report Manager and, when opened, displays the current script/report. It offers a number of options:



It is possible to specify the view scale, using the respective icon or the right-click menu:



Further options include opening a report/script, saving it, printing the report/script previewed, and even searching for text within the script:



The last icon in the Print Preview toolbar allows the Print Preview window to be closed.

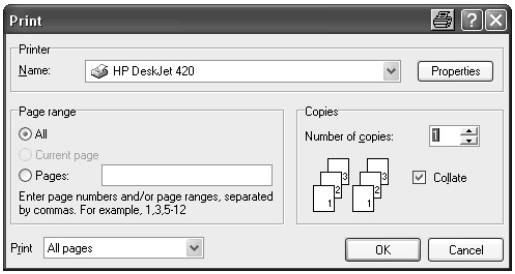
The right-click menu, in addition to scale specification, also offers options to add a page (for example, for a front cover or introduction) or delete one, and also to edit the page previewed, by opening the Report Designer.

The Report Designer (part of the Report Manager) can also be automatically opened by double-clicking on the report, enabling the user to make alterations to the layout as wished.

4.6 Print

This item can be found in the IBExpert Edit menu and SQL Editor right-click menu (available in the SQL and Object Editors), and as an icon on the relevant toolbars, for printing scripts, reports or database object metadata.

It opens a standard Windows Print dialog:



including the usual options such as printer specification (and properties), page range and number of copies.

4.7 Page Setup

This item can be found in the IBExpert Edit menu and SQL Editor right-click menu (available in the SQL and Object Editors).

It opens a standard Windows Page Setup dialog, where the following options can be specified:

- Paper size
- Source (i.e. printer tray specification)
- Portrait or landscape
- Margins

as well as a Printer button to specify the printer.

4.8 Convert Identifiers/Keywords

The menu item, Convert Identifiers/Keywords, can be found in the IBExpert Edit menu or in the right-click Text Editor/Code Editor menu. It offers the following options to alter the appearance of the SQL characters:

Convert keywords: allows all keywords (i.e. statements, commands etc.) in the current SQL script to be converted completely to lower or upper case.

Convert identifiers: allows all identifiers (i.e. object names, field names etc.) in the current SQL script to be converted completely to lower or upper case.

5 IBExpert Grid Menu

The IBExpert Grid menu item is new to version 2003.11.6.1. It includes the following:

- Apply Best Fit
- Save Grid Data as
- Copy Current Record to Clipboard
- Copy All to Clipboard.

It is of course necessary to be in an active grid (e.g. Table Editor / Data page, View Editor / Data page, SQL Editor / Results page etc.) for any of these menu items to be effective!

5.1 Apply Best Fit

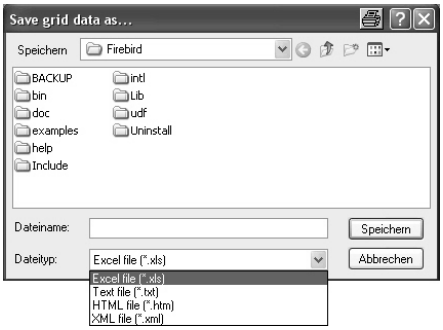
The IBExpert menu item Apply Best Fit is new to IBExpert version 2003.11.6.1 and can be started from the Grid menu, or using the key combination [Ctrl + (NumBlock +)].

This automatically adjusts all grid columns to the ideal width.

5.2 Save Grid Data as

The IBExpert menu item Save Grid Data as is new to IBExpert version 2003.11.6.1 and can be started from the Grid menu, or using the key combination [Shift + Ctrl + S].

It opens the Save Grid Data As... dialog:



It is possible to save grid data into TXT, XLS, HTML or XML formats. This works only with dataset grids (field and index grids in the Table Editor, the parameters/variables grid in the Stored Procedure Editor while working in lazy mode), and doesn't work with SQL Assistant lists, the constraint list in the Table Editor etc.

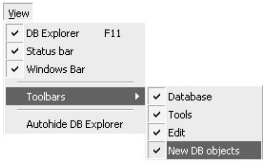
5.3 Copy Current Record to Clipboard/Copy All to Clipboard

The IBExpert menu items Copy Current Record to Clipboard and Copy All to Clipboard are new to IBExpert version 2003.11.6.1 They can be started from the Grid menu, and

used to copy either one selected record or all records (including column captions) in an active grid to clipboard. The values are delimited with the tab character.

6 IBExpert View Menu

The IBExpert View menu allows the developer to specify which, of certain options, he wishes to have displayed on screen. This eliminates superfluous or unnecessary items on screen. The options available can be seen in the following illustration:



The options DB Explorer, status bar and windows bar can be blended in and out simply by clicking on the check box (alternatively using the space bar). The menu item Toolbar is subdivided into the four main standard toolbars: Database toolbar, Tools toolbar, Edit toolbar, and New DB Objects toolbar.

Autohide DB Explorer is a further alternative to quickly blend the DB Explorer in and out as wished (alternatively use the [F11] key). This option namely enables the DB Explorer to disappear automatically when any editor is opened - allowing a larger working area. It is blended back into view simply by holding the mouse over the left-hand side of the IBExpert main window.

7 IBExpert Options Menu

The IBExpert Options menu enables you to organize your IBExpert working environment as you wish. It includes the following options:

- Environment Options
- Editor Options
- Visual Options
- Keyboard Templates
- General Templates
- Object Editor Options

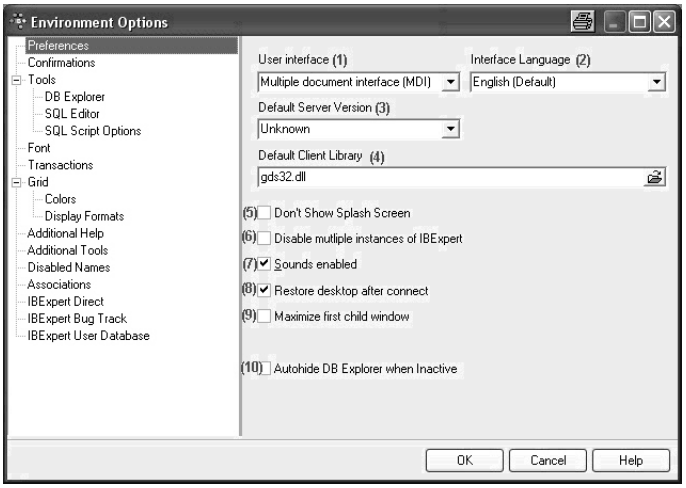
7.1 Environment Options

Environment Options can be found in the IBExpert Options menu. It enables the user to organize his IBExpert working environment as he wishes. It is possible, for example, to set certain defaults for editors and specific menu items, alter colors or the system font, etc. It includes the following options (please click the links above to read more about the individual subjects):

- Preferences,
- Confirmations,
- Tools,
- Font,
- Transactions,
- Grid,
- Additional Help,
- Additional Tools,
- Disabled Names,
- Associations,
- IBExpert Direct,
- IBExpert Bug Track,
- IBExpert User Database.

7.1.1 Preferences

The Preferences window allows the user to specify certain general preferences or defaults.



These include:

(1) User interface: the pull-down list offers the options MDI or SDI (please refer to User Interface for more details). Please note that changes to the user interface only take effect after IBExpert has been restarted.

(2) Interface Language: the default language is English. The pull-down list offers the following alternative languages:

- Czech
- English
- French
- German
- Italian
- Japanese*
- Polish*
- Portuguese
- Russian
- Spanish

* These languages are available since IBExpert version 2004.04.01.1. Other language files have also been updated with this version. In order to install these language files, it is necessary to install the complete version (i.e. `ibec_<version_no=>2004.04.01.1>_full.zip`, and not the `ibec_<version_no>_exe.zip`; since September 2004 these two alternatives have been replace with the `_full` version).

(3) Default Server Version: If the same database version is used for all projects, it is advisable to set a default version here. This saves having to enter the database server version every time a database is registered. The pull-down list offers the following database versions:

- Unknown (default)

- InterBase 5.x
- InterBase 6.0-6.1
- InterBase 6.5
- InterBase 7.0-7.1
- Firebird 1.0
- Firebird 1.5
- Yaffil 1.0

(4) Default Client Library: The `GDS32.DLL` is dependent upon the database server. Firebird has, in addition to this, its own library, `FBCLIENT.DLL`. The `GDS32.DLL` is however also included for compatibility reasons. When working with Firebird, or different InterBase/Firebird server versions, the DLL can be selected here, as wished; simply click the Open File icon to the right of this field, to select the library required.

The following features can be checked or unchecked as wished:

- **(5) Don't Show Splash Screen** - disables the IBExpert Splash Screen displayed whilst IBExpert is being loaded.
- **(6) Disable multiple instances of IBExpert** - when checked this option ensures that IBExpert is only opened once.
- **(7) Sounds enabled** - switches sound on and off.
- **(8) Restore desktop after connect** - if this option is checked, IBExpert will restore all those forms left open as the last connection was ended, when it reconnects to the database.
- **(9) Maximize first child window** - the first Editor/window opened is automatically expanded to fill the maximum screen area. This option is only available in the MDI version.
- **(10) Autohide DB Explorer when inactive** - this option autohides the DB Explorer, if it is not focused. In other words, when the mouse is held over the left area, the DB Explorer appears; when the mouse is removed to begin work in an editor or child window, the DB Explorer is blended out, offering a larger work area.

User Interface

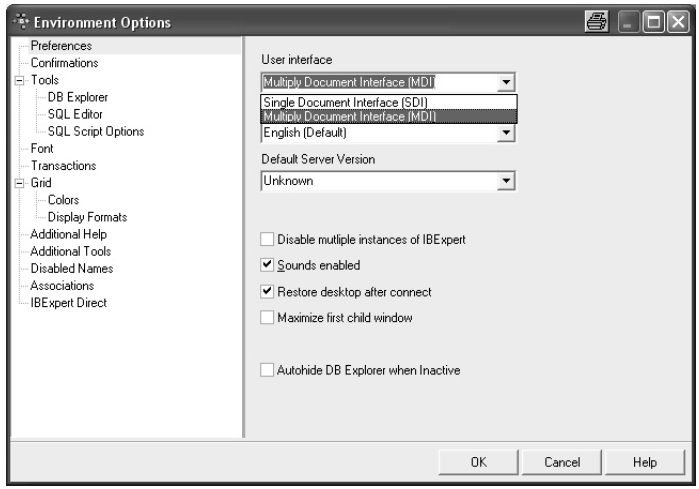
The user interface is the connection between the machine and the user, i.e. the way the software is presented to the user on-screen. The user interface enables the user to use the program and manipulate data.

Under the IBExpert menu item Options / Environment Options, the user interface can be defined as SDI (Single Document Interface) or MDI (Multiple Document Interface).

MDI (Multiple Document Interface)

MDI is the abbreviation for Multiple Document Interface.

It can be specified in the IBExpert menu item Options / Environment Options.



This is the recommended interface, as all windows are contained within one main Window, similar to MS applications. There is one document per window. For all additional objects or documents, the Windows operating system opens an additional window.

The status bar can be seen at the bottom of the screen.

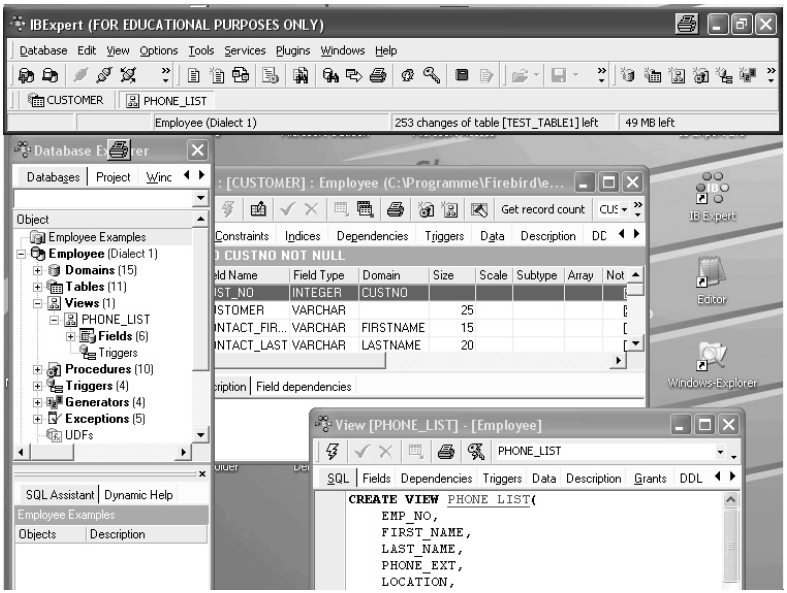
When changing the interface from SDI to MDI and vice versa, IBExpert needs to be restarted for the alterations to take effect.

SDI (Single Document Interface)

SDI

SDI is the abbreviation for Single Document Interface.

It can be specified under the IBExpert menu item Options / Environment Options.



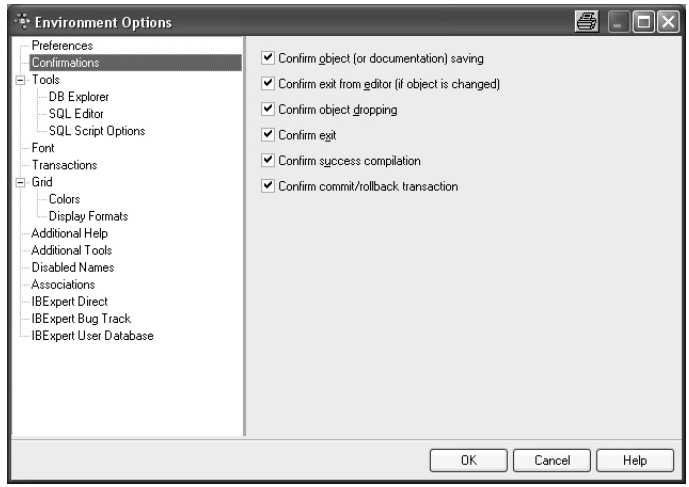
The windows are spread freely and somewhat haphazardly over the screen, similar to Delphi. The status bar is part of the upper menu and toolbar panel.

Careful: it is possible to accidentally move a window totally out of view!

When altering the user interface from SDI to MDI and vice versa, IBExpert needs to be restarted for the change to take effect.

7.1.2 Confirmations

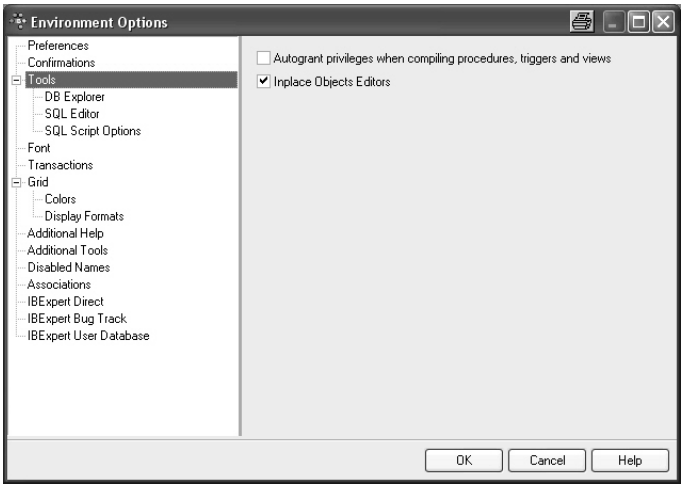
Some users find it annoying to be constantly asked for confirmation, whether or not they really want to carry out an operation. This window allows the user to specify, which confirmations he considers wise.



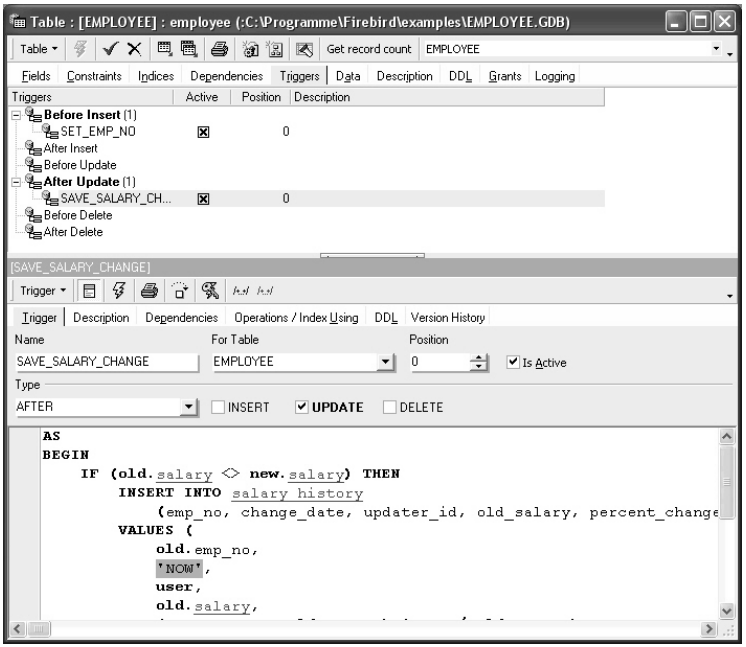
The following options are available:

- **Confirm object (or documentation) saving** - if this options is checked, IBExpert will request confirmation before saving object modifications or descriptions.
- **Confirm exit from editor (if object is changed)** - if this options is checked, IBExpert will request confirmation, if alterations have been made, before exiting from an object editor.
- **Confirm object dropping** (recommended) - if this options is checked, IBExpert will request confirmation before dropping any database object.
- **Confirm exit -** if this options is checked, IBExpert will request confirmation before closing IBExpert.
- **Confirm successful compilation** (recommended) - if this options is checked, IBExpert displays a dialog, showing whether compilation was successful or not..
- **Confirm commit/rollback transaction** (recommended) - this option determines whether a message box appears, asking for confirmation when a user commits or rolls back active transactions in the SQL Editor, Table Editor, View Editor or Stored Procedure Editor.

7.1.3 Tools

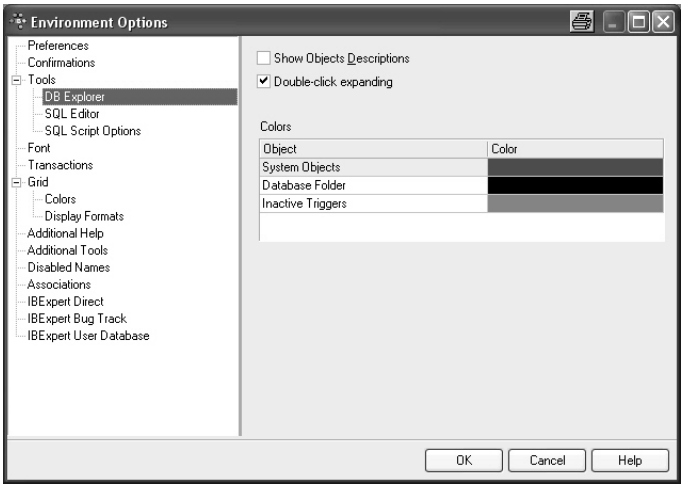


- **Autogrant privileges when compiling procedures, triggers and views** - this saves the repetitive task of autogranting privileges on the Grants page of the object editors each time a new procedure, trigger or view is created, and prevents the problems which inevitably arise, should the assignment of rights be forgotten.
- **Revoke existing privileges** - this option is available since IBExpert version 2005.02.12.1. If it is enabled, existing privileges of an object (stored procedure, trigger, view) will be deleted before granting it new privileges.
- **Inplace Objects Editors** - this item applies to the so-called editors within editors.



For example, the Table Editor is active and a trigger is selected on the Trigger page: if this option is not checked, an SQL Editor window appears automatically in the lower part of the Table Editor, displaying the trigger code, but not allowing any changes to be made. When this option is however checked, a simple click on a trigger automatically opens the Trigger Editor in this lower area, enabling work to be done on this trigger, without having to leave the Table Editor and opening the Trigger Editor.

DB Explorer

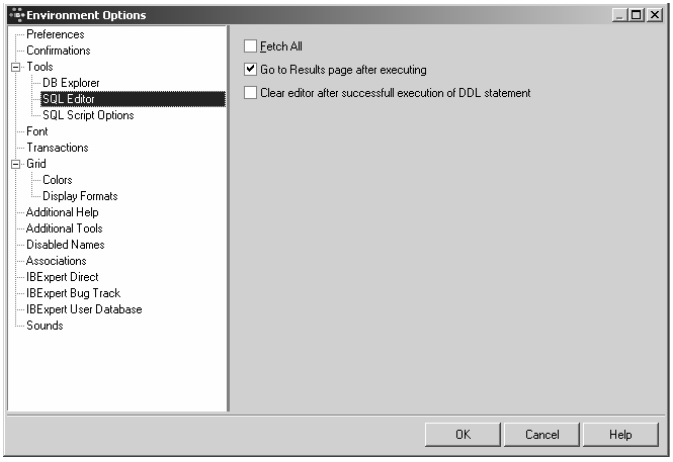


Here it is possible to specify whether database object descriptions should be displayed or not (only makes sense if object descriptions are entered by the user), and whether double-click expanding (for the DB Explorer tree) is desired.

Furthermore, colors may be specified for the following:

- system objects
- database folders
- inactive triggers

SQL Editor

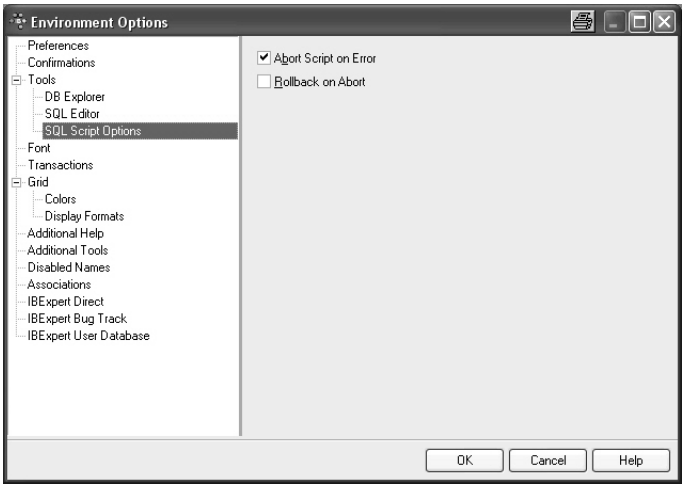


The following options may be user-defined for the SQL Editor:

- **Fetch All** - when this option is checked, all records corresponding to the query will be extracted from the table; as opposed to only those displayed on the Results page, when this option is left unchecked.
- **Go to Results page after executing**
- **Clear editor after successful execution of DDL statement** - new to version 2.5.0.61.

SQL Script Options

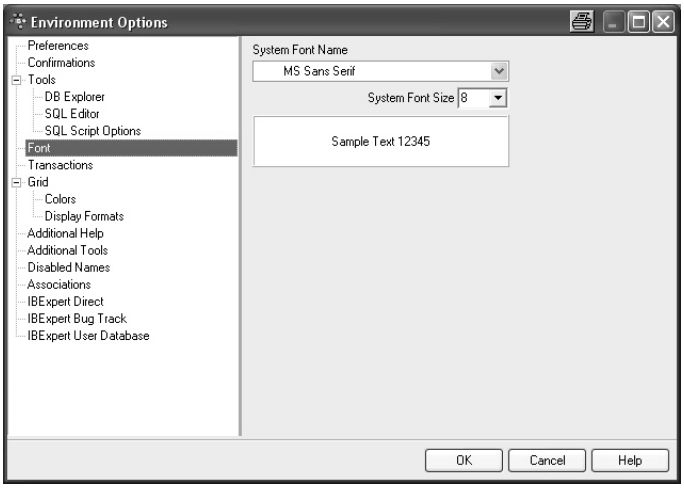
The SQL Script Options page offers the following options which may be specified by the user:



- **Abort Script on Error** - the script execution is halted the moment an error is detected.
- **Rollback on Abort** - the script is automatically rolled back the moment an error is detected in the script. This option is only possible, if the first item, Abort Script on Error, is already selected.

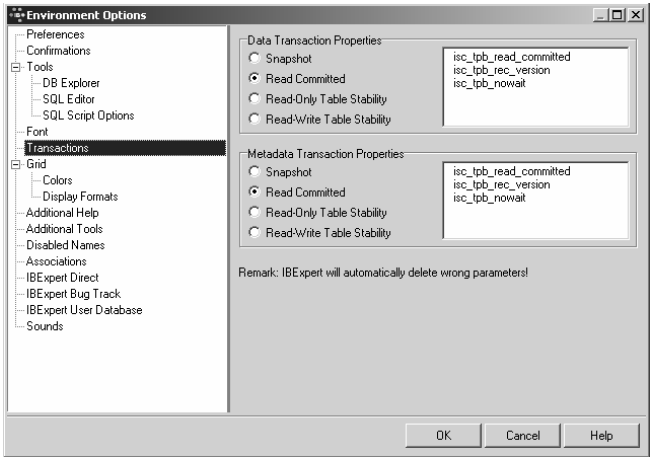
7.1.4 Font

Here it is possible for the user to specify the system (i.e. IBExpert application) font name and size. The `Sample Text 12345` displays the specified font as it will appear in IBExpert.



7.1.5 Transactions

Here certain additional data and metadata transaction properties may be defined for the server connection.



These are all InterBase/Firebird API terms, and may be checked as wished.

Data Transaction Properties:

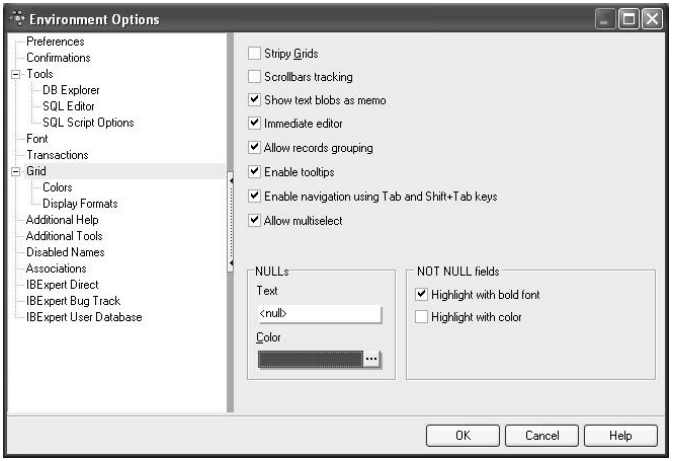
- Snapshot
- Read Committed
- Read-Only Table Stability
- Read-Write Table Stability

Metadata Transaction Properties:

- Snapshot
- Read Committed
- Read-Only Table Stability
- Read-Write Table Stability

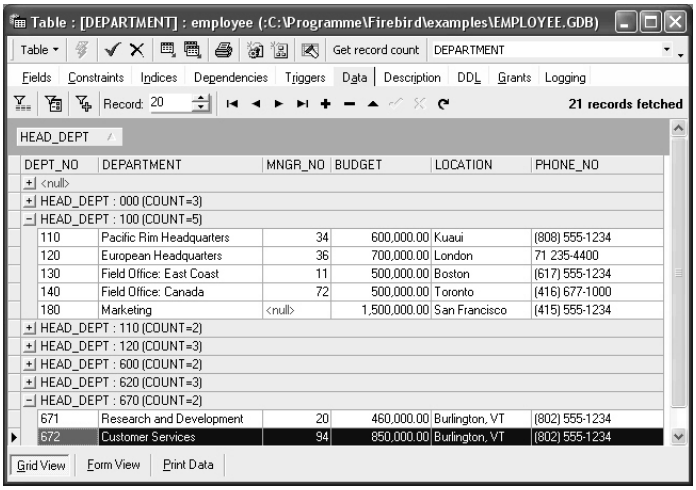
7.1.6 Grid

Here a range of options are available, applicable for all data grids:



Check boxes for the following options:

- **Stripy Grids** - makes reading wide lines of data rows easier.
- **Scrollbars tracking**
- **Show text blobs as memo** - The memo option enables the blob to be easily read by simply focusing the cursor over the blob field.
- **Immediate editor** - Enables immediate editing in the data grid by simply placing the cursor on a field, without having to first double-click on the field, in order to edit it.
- **Allow records grouping** - When this option is checked, an additional gray bar appears above the column headers over the grid. A column header simply needs to be dragged 'n' dropped into this area, to group by the selected column. A reorganized data view appears, where the group contents can be revealed or hidden, by clicking on the '+' or '-' buttons. Please note that this is not the same as the data grid right-click menu item Group Fields/Ungroup Fields.



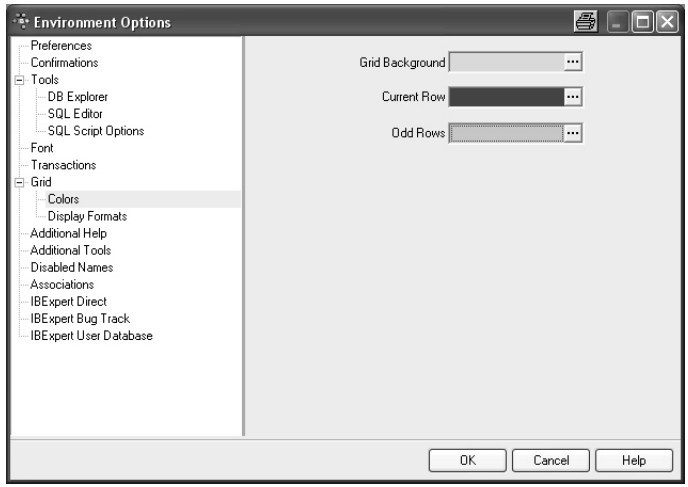
- **Enable tooltips** - when checked, this option displays the full field contents when the cursor is held over a particular field, if the column width is not sufficient to display all information. This is useful, if tables with many columns and long field contents need to be scanned.
- **Enable navigation using [Tab] and [Shift + Tab] keys**
- **Allow multiselect** - allows multiple data sets to be selected for editing (e.g. copying). If this is not checked, it is only possible to select one data set at a time. The change of mode can be recognized by the form/shade of the arrow on the left when pointing at a selected data set.

Furthermore it is possible to specify the exact representation of a Null field. The default value is displayed as <null> (in red). Since version 2004.2.26.1 it is also possible to display NOT NULL fields as bold or to highlight with color.

Colors

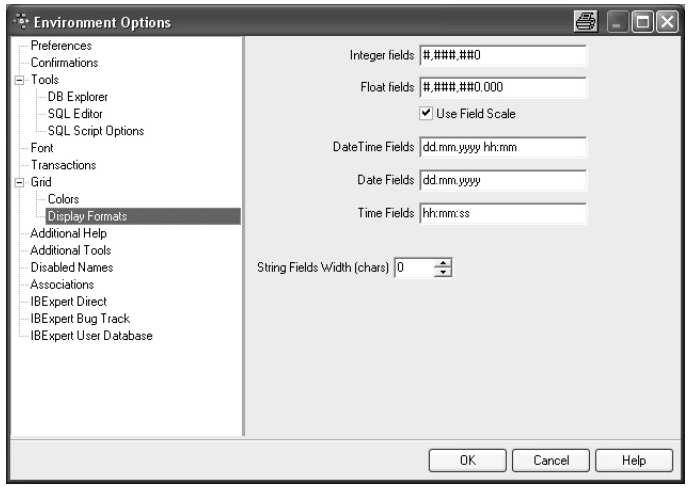
Here the user can specify the colors for different elements in the grids:

- Grid Background
- Current Row
- Odd Rows



Display Formats

These options allow the user to specify the display format in grids for integer, float, date, time and date/time fields.



Further options include a check box option for **Use field scale**, which allows a field definition to override these standard definitions, and an option to specify the **String fields' width** for characters.

For the various date and time formatting options available, please refer to date time formats.

Date Time Formats

The following format allows you to alter the way the date and time is displayed. Please note that this does not alter the way this information is stored, only the way it is displayed.

Controls formatting of dates and times

Description

Date time format strings specify the formatting of date-time values (such as `TDateTime`) when they are converted to strings. date time format strings are passed to formatting methods and procedures (such as `FormatDateTime`), and are also used to set certain global variables (such as `ShortDateFormat`).

Date time format strings are composed from specifiers that represent values to be inserted into the formatted string. Some specifiers (such as "d"), simply format numbers or strings. Other specifiers (such as "/") refer to local-specific strings from global variables.

In the following table specifiers are given in lower case. Case is ignored in formats, except for the "am/pm" and "a/p" specifiers.

Specifier Displays

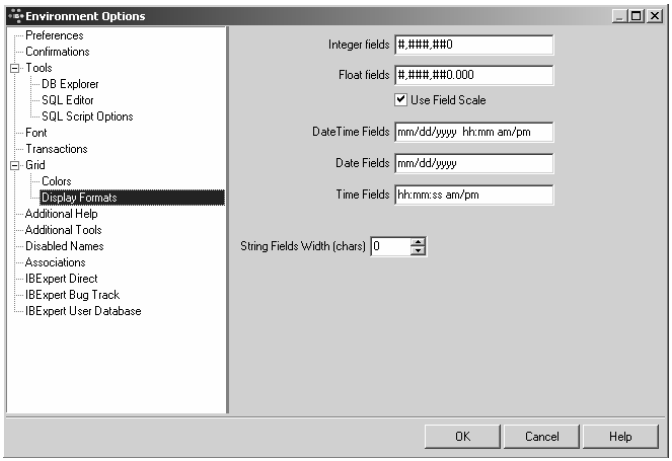
- **c** Displays the date using the format given by the `ShortDateFormat` global variable, followed by the time using the format given by the `LongTimeFormat` global variable. The time is not displayed if the date-time value indicates midnight precisely.
- **d** Displays the day as a number without a leading zero (`1-31`).
- **dd** Displays the day as a number with a leading zero (`01-31`).
- **ddd** Displays the day as an abbreviation (`Sun-Sat`) using the strings given by the `ShortDayNames` global variable.
- **dddd** Displays the day as a full name (`Sunday-Saturday`) using the strings given by the `LongDayNames` global variable.
- **ddddd** Displays the date using the format given by the `ShortDateFormat` global variable.
- **dddddd** Displays the date using the format given by the `LongDateFormat` global variable.
- **e** Displays the year in the current period/era as a number without a leading zero (Japanese, Korean and Taiwanese locales only).
- **ee** Displays the year in the current period/era as a number with a leading zero (Japanese, Korean and Taiwanese locales only).
- **g** Displays the period/era as an abbreviation (Japanese and Taiwanese locales only).
- **gg** Displays the period/era as a full name. (Japanese and Taiwanese locales only).
- **m** Displays the month as a number without a leading zero (`1-12`). If the m specifier immediately follows an `h` or `hh` specifier, the minute rather than the month is displayed.
- **mm** Displays the month as a number with a leading zero (`01-12`). If the mm specifier immediately follows an `h` or `hh` specifier, the minute rather than the month is displayed.

- **mmm** Displays the month as an abbreviation (`Jan-Dec`) using the strings given by the `ShortMonthNames` global variable.
- **mmmm** Displays the month as a full name (`January-December`) using the strings given by the `LongMonthNames` global variable.
- **yy** Displays the year as a two-digit number (`00-99`).
- **yyyy** Displays the year as a four-digit number (`0000-9999`).
- **h** Displays the hour without a leading zero (`0-23`).
- **hh** Displays the hour with a leading zero (`00-23`).
- **n** Displays the minute without a leading zero (`0-59`).
- **nn** Displays the minute with a leading zero (`00-59`).
- **s** Displays the second without a leading zero (`0-59`).
- **ss** Displays the second with a leading zero (`00-59`).
- **z** Displays the millisecond without a leading zero (`0-999`).
- **zzz** Displays the millisecond with a leading zero (`000-999`).
- **t** Displays the time using the format given by the `ShortTimeFormat` global variable.
- **tt** Displays the time using the format given by the `LongTimeFormat` global variable.
- **am/pm** Uses the 12-hour clock for the preceding `h` or `hh` specifier, and displays `'am'` for any hour before noon, and `'pm'` for any hour after noon. The am/pm specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
- **a/p** Uses the 12-hour clock for the preceding `h` or `hh` specifier, and displays 'a' for any hour before noon, and 'p' for any hour after noon. The a/p specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
- **ampm** Uses the 12-hour clock for the preceding `h` or `hh` specifier, and displays the contents of the `TimeAMString` global variable for any hour before noon, and the contents of the `TimePMString` global variable for any hour after noon.
- **/** Displays the date separator character given by the `DateSeparator` global variable.
- **:** Displays the time separator character given by the `TimeSeparator` global variable.
- **'xx'/"xx"** Characters enclosed in single or double quotes are displayed as-is, and do not affect formatting.

Example:

To format the date as month, day, year and the time as am or pm, simply enter the following in Display Formats (IBExpert Options Menu / Environment Options / Grid /face="">To format the date as month, day, year and the time as am or pm, simply enter the following in Display Formats (IBExpert Options Menu / Environment Options / Grid / Display Formats):

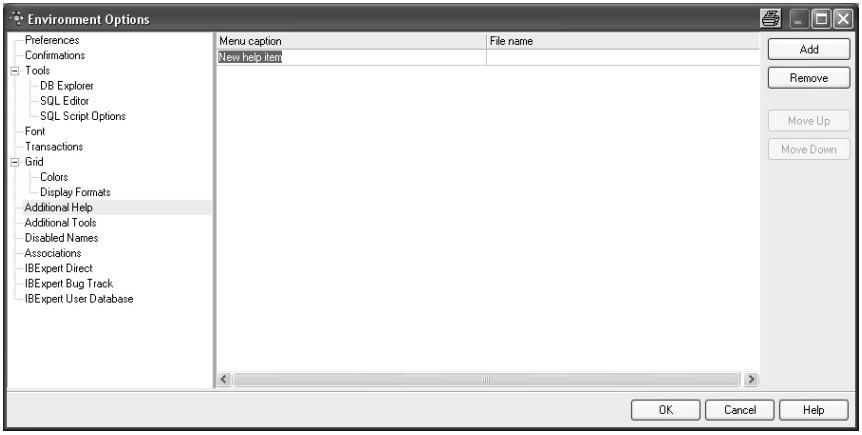
Simply alter DateTime Fields to: **mm/dd/yyyy hh:mm am/pm**and Time Fields to **hh:mm:ss am/pm**



7.1.7 Additional Help

The Additional Help dialog allows the user to add certain additional help files. This is particularly useful for incorporating the help files of third party components, installed in the PlugIn menu.

An additional menu item is automatically inserted in the IBExpert Help menu, for each of these help files.



7.1.8 Additional Tools

The Additional Tools dialog allows the user to add certain additional third party tools.

For more details, please refer to the IBExpert PlugIns Menu.

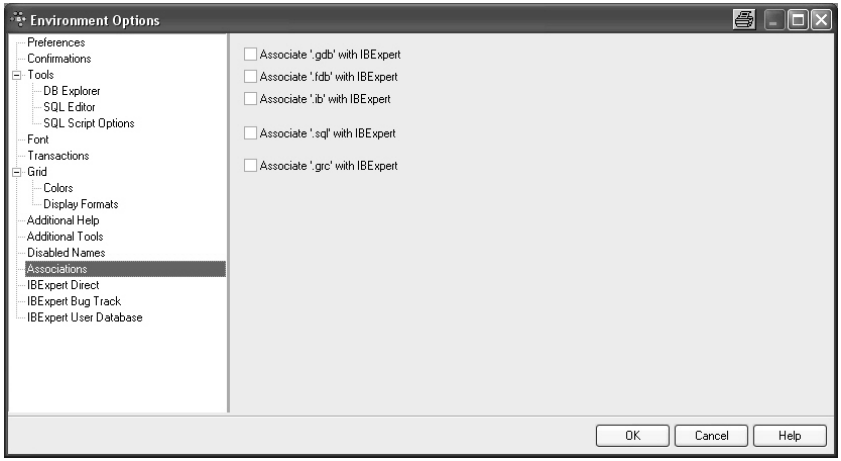
7.1.9 Disabled Names

This page can be used to define a list of disabled object names. IBExpert refers to this list, when new database objects (and fields) are created, and publishes a warning if the new name corresponds to any name in this list.

7.1.10 Associations

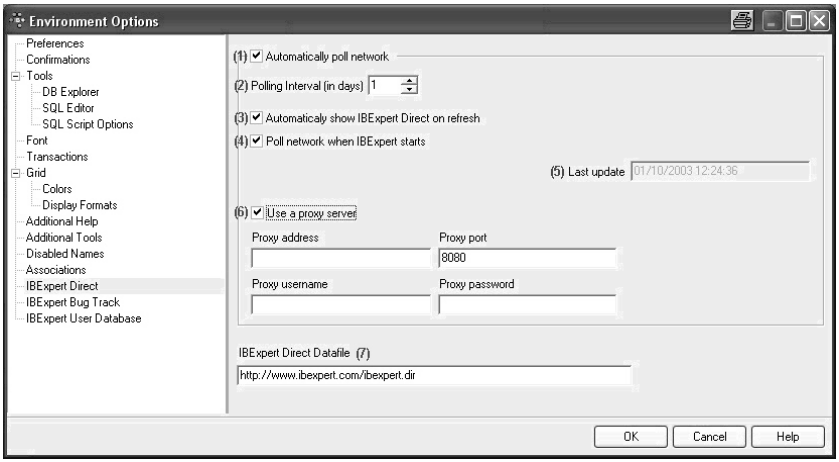
This dialog is important, to specify which file types IBExpert should recognize and associate with the InterBase/Firebird database. The check list includes the following suffixes:

- .GDB
- .FDB
- .IB
- .SQL
- .GRC (new to version 2.5.0.61)



7.1.11 IBExpert Direct

The IBExpert Direct dialog allows the user to specify a number of options, regarding this IBExpert menu item found in the Help menu. This window can be either started from the IBExpert Options menu or alternatively directly from the Help menu item IBExpert Direct using the respective icon:



The options available include the following:

(1) Automatically poll network - this is recommendable, as IBExpert Direct is an important news source, informing all IBExpert users of news concerning IBExpert, such as new versions, documentation, downloads, plugins, newsgroups, as well as contact addresses and a direct link to the IBExpert home page, http://www.ibexpert.com.

(2) The **polling interval** in days can be user-specified.

Check boxes allow the user to specify whether IBExpert Direct should (3) automatically shown on refresh, or whether the network should be polled for new items, (4) each time IBExpert is started.

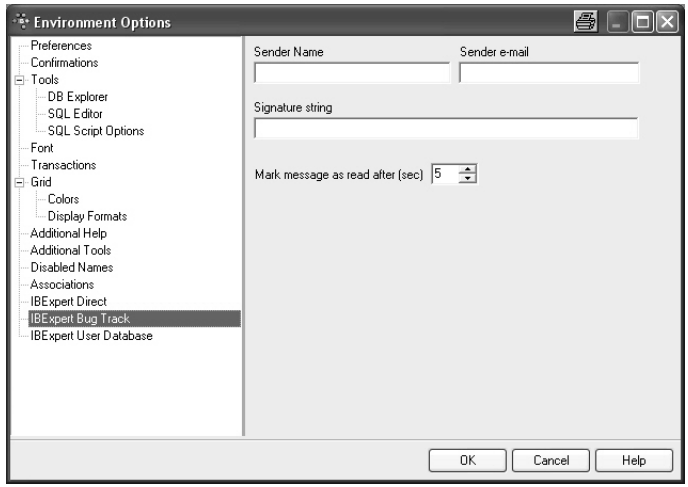
(5) The **Last Update** field is purely a display field, showing the last time the network was polled for new IBExpert Direct news items.

(6) It is also possible to specify a **proxy server** if necessary, with fields for specification of the proxy address, port, user name and password.

(7) The last field displays the IBExpert Direct link address http://www.ibexpert.com/ibexpert.dir

7.1.12 IBExpert Bug Track

This option allows the user to specify his signature before posting bugs in the IBExpert Help menu item, Bug Track System.

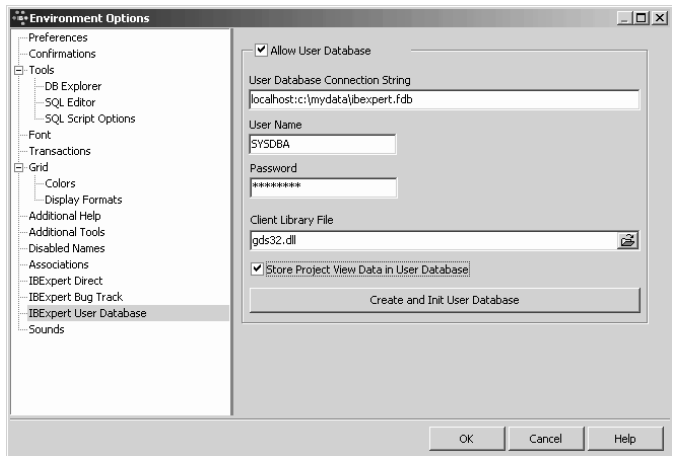


The IBExpert Bug Track System was introduced in IBExpert version 2.5.0.38 (on 28.04.2003). The Bug Track signature requires the following information:

- Sender Name
- Sender email
- Signature string

The option "Mark message as read after (sec)" applies to all bug messages listed in the Bug Track System.

7.1.13 IBExpert User Database



The complete IBExpert configuration and work is stored here in the IBExpert User Database. We recommend using the user database as a main storage for security reasons.

The IBExpert User Database dialog requires the following information, in order to create a new user database. After checking the Allow User Database checkbox the following fields need to be completed:

- User Database Connection String - e.g. If you're using local server the connection string should be as follows: localhost:`c:\mydata\ibexpert.fdb` (for TCP/IP protocol) or just `c:\mydata\ibexpert.fdb` (for a local protocol).
- User Name (default: SYSDBA)
- Password
- Client Library File
- Check box: Store Project View Data in User Database

The user database can then created and initialized using the "Create and Init User Database" button.

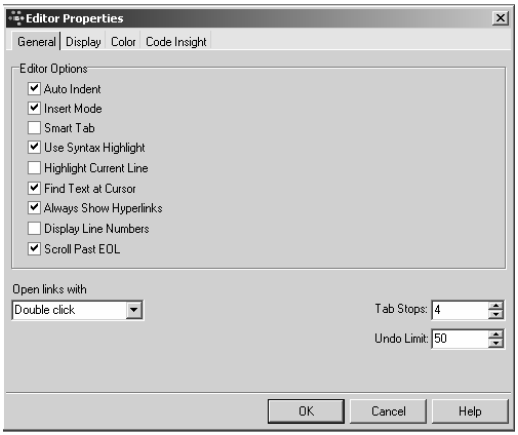
7.2 Editor Options

Editor Options can be found in the IBExpert Options menu. It opens the Editor Properties dialog, which enables the user to organize and customize IBExpert editors as he wishes. It is possible, for example, to set certain defaults, or alter the font or colors, customize code completion etc. It includes the following options:

- Editor Options / General,
- Editor Options / Display,
- Editor Options / Color,
- Editor Options / Code Insight.

7.2.1 General

The first page in the Editor Properties dialog is the General page, which offers the following options:



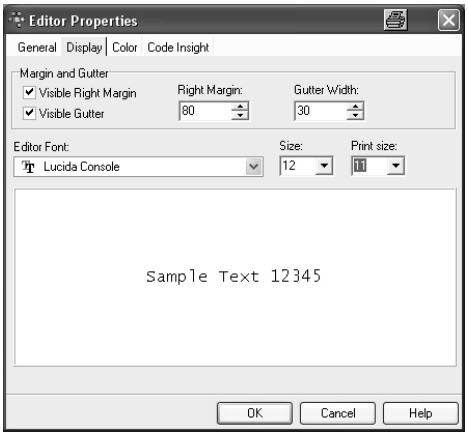
- **Auto Indent** (default) - this automatically indents code when editing SQL script; each new indention identical to the previous. The tab (= tabulator) length can be specified using the lower right *Tab Stops* counter (default = 4 characters).
- **Insert Mode** (default) - inserts text at the cursor without overwriting existing text. When disabled (i.e. when unchecked), the so-called *typeover mode* is activated, i.e. the text at the cursor is overwritten. It is possible to use the [Ins] key to switch the *insert mode* on and off in the code editor, without having to alter the default.
- **Smart Tab** - this automatically limits the tab stop lengths to the length of the previous line.
- **Use Syntax Highlight** (default) - enables highlighted syntax in the object editor window. To set highlighting options, please refer to Editor Options / Color.
- **Highlight Current Line** - useful for orientation in long scripts.
- **Find Text at Cursor** (default) - searches automatically for the word, where the cursor happens to be standing when starting the IBExpert Edit / Find menu item (see also Search - [Strg + F]). This saves having to mark the word first, or type in the text to be searched for each time.
- **Always Show Hyperlinks** (default) - displays hyperlinks in SQL script as green underlined text (unless altered by the user under Editor Options / Color). It can be opened by double-clicking or single-clicking (user-defined; see *Open links with* below).
- **Show Lines Number** - useful when working with long scripts. This option displays line numbers in the gutter in the editor window. A gutter is automatically inserted, even if it has been unchecked on the Display page (please refer to Editor Options / Display).
- **Scroll past end of line** (default) - when this is not checked, the cursor jumps to the beginning of the next line automatically when it has reached the end of the text. If this option is checked, the cursor continues to travel to the right, even after the end of the text has been reached.

Furthermore it is possible to specify the following:

- **Open links with**: double click (default) or single click.
- **Tab Stops** - defines the tab length (see above)
- **Undo limit** - specifies the maximum number of keystrokes, that can be undone (default = 50).

7.2.2 Display

The Display page allows the user to specify certain visual editor properties.

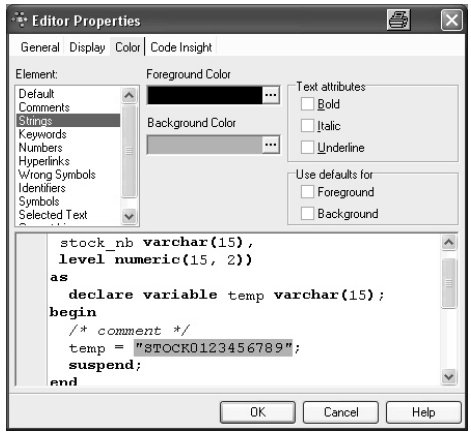


The options available here include:

Margin (= right margin) **and Gutter** (= inner or left margin):
. Visible Right Margin and Gutter (check box option to blend margins in or out)
. User specification of right margin position and gutter width (in characters).
Note: checking the "Show Lines number" box on the General page automatically inserts a gutter, even if it is not checked here.
Editor Font:
User specifications include font, size and print size (with sample text preview). The advantage here is that it is possible to specify a larger or smaller display font size than the print font size.

7.2.3 Color

The Color page allows the user to specify colors and text attributes for a range of elements:



The range of elements includes the following:

- default,
- comments,
- strings,
- keywords,
- numbers,
- hyperlinks,
- wrong symbols,
- identifiers,
- symbols,
- selected text,
- current line,
- double-quoted string (new to IBExpert version 2003.11.6.1),
- conditional directive (new to IBExpert version 2003.11.6.1).
- IBEBlock procedure/function (new to IBExpert version 2005.02.12.1).

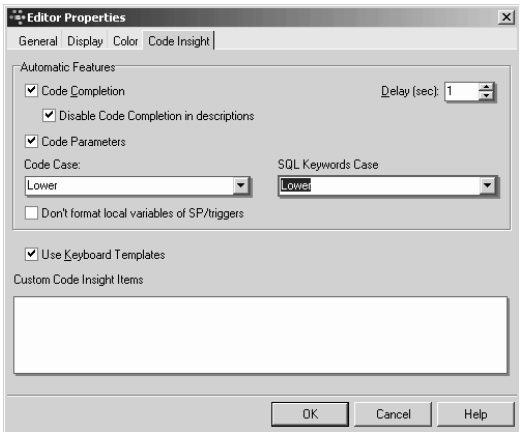
The following properties can be specified for the above elements:

- **Foreground Color**: determines the color of the selected element in the foreground (usually text)
- **Background Color**: determines the color of the selected element in the background (generally used to highlight text)
- **Text attributes** - includes specification of bold, italic and/or underline.
- **Use defaults for** - allows the default to be rapidly specified for both the foreground and background colors for a selected element.

The text preview panel displays the elements as they have been specified, allowing the user to approve or alter his choice, or return to the default settings using the *Use defaults for Foreground/Background* check boxes.

7.2.4 Code Insight

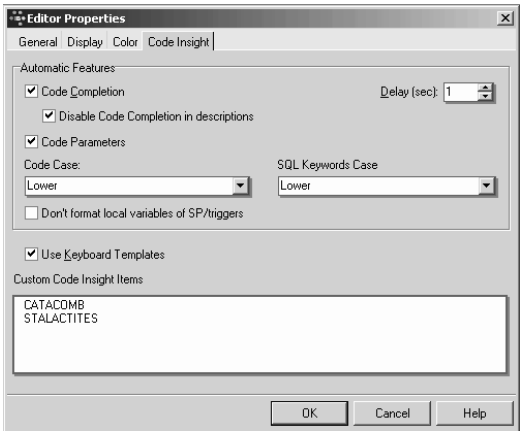
The Code Insight page offers a number of options related to the IBExpert automatic code completion:



These include:

- **Code Completion** - here the user can specify, whether code completion should be active or not.
- **Disable Code Completion in Descriptions** - a new feature in IBExpert version 2005.01.12.1, allowing the user to disable code completion while editing an object description.
- **Code Parameters** - this is a very useful option when active. For example when working with procedures, a list of all necessary input parameters automatically appears, and when one or more parameters have already been specified, the next parameter required appears in bold type. Since IBExpert version 2003.11.6.1 the list of fields to be inserted is now displayed when the `VALUES` part of an `INSERT` statement is typed.
- **Delay in seconds**, before the code completion pop-up list appears with a list of one or more possible suggestions (default value is 1 second).
- **Code Case** - user specification of the words (e.g. object names, field names) inserted automatically by code insight: either lower (default), upper, first upper or name case.
- **Code Case** and **SQL Keywords Case** - user specification of the SQL keywords inserted automatically by code insight: either lower (default), upper, first upper or name case. There is also a check option to disable formatting of local variables when working with stored procedures and triggers.

It is also possible to specify whether keyboard templates (for faster typing of regularly used words or expressions) should be used, and the Custom Code Insight Items display panel displays those items, specified by the user.



7.3 Visual Options

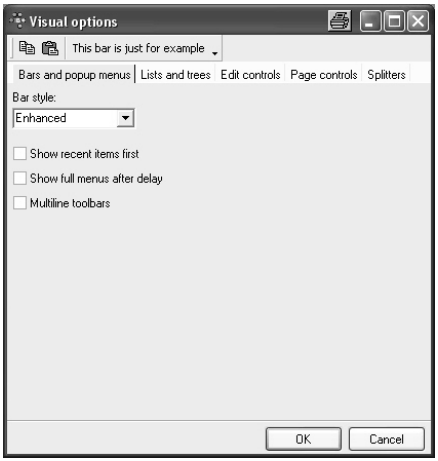
Visual Options can be found in the IBExpert Options menu. It opens the Visual Options Editor, which enables users to customize the IBExpert interface. It is possible, for example, to specify the behavior of pop-up menus, the appearance of border and button styles, and even of splitters.

It includes the following options:

- bars and pop-up menus
- lists and trees
- edit controls
- page controls
- splitters

7.3.1 Bars and Pop–up Menus

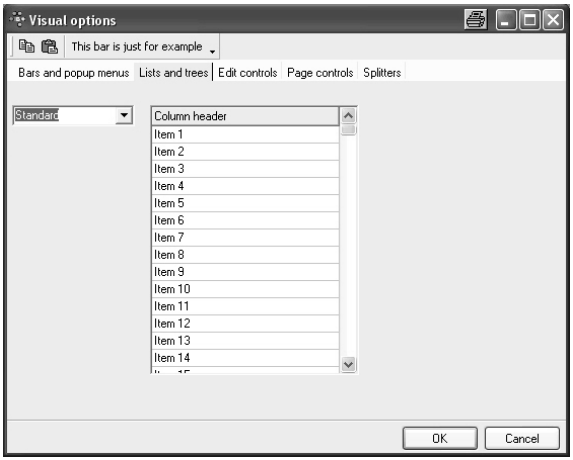
The first tab in the Visual Options Editor is the Bars and Pop-up Menus page, which offers the following options:



- **Bar Style** - the options *Standard*, *Enhanced* or *Flat* may be selected. The visual effects of the selection is immediately visible in the sample toolbar, displayed at the top of the Visual Options dialog.
- **Show recent items first** - reduces those menu items offered in the pull-down list, to those most recently selected by the user.
- **Show full menus after delay** - if one of the most recent menu items is not immediately selected, the full range of menu items is displayed.
- **Multiline toolbars** - allows toolbars to cover more than a single row (which may eventually lead to icons running off the right-hand side of the screen, if too many toolbars are active).

7.3.2 Lists and Trees

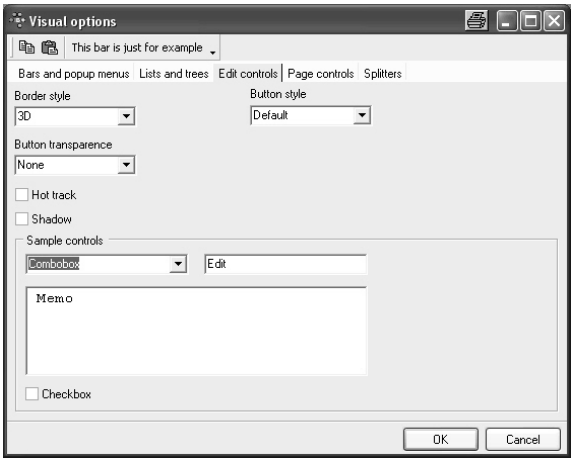
The Lists and Trees page offers the following options:



Lists and trees may be displayed in a *Standard*, *Flat* or *Ultraflat* format. The visual effects of the selection can immediately be seen in the example field grid, displayed to the right of the pull-down list.

7.3.3 Edit Controls

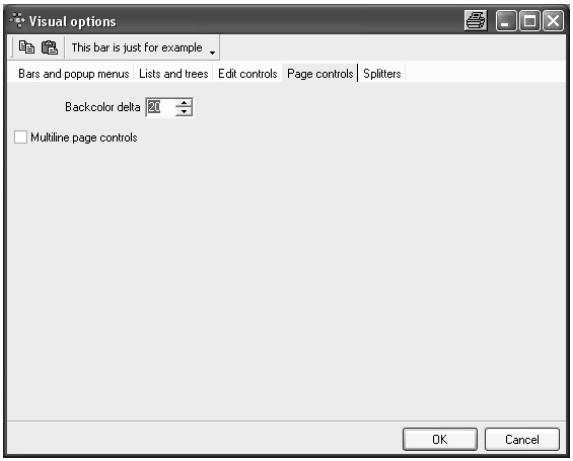
The third tab in the Visual Options Editor is the Edit Controls page, which offers the following options:



- **Border Style** - options offered include *None*, *Single*, *Thick*, *Flat* and *3D*. The visual effects of the selection is immediately visible in the sample controls panel in the lower area of the window.
- **Button Style** - the options offered here include *Default*, *3D*, *Flat Simple*, *HotFlat*. This changes the style of displaying application buttons. The effect can be previewed in the sample controls (observe the combo box and check box).
- **Button Transparence** - here the options include *None*, *Inactive*, *Always Hide* and *Inactive*. This alters the appearance of transparent buttons. The effects can be viewed in the sample controls (observe the combo box).
- **Hot Track** - activating this option causes boxes and buttons to be highlighted with a 3D effect, when the mouse is focused over it. The effect only be previewed on all sample controls, if the *Border Style None* has been selected. Otherwise this effect can only be observed on the combo box.
- **Shadow** - this option places a shadow effect around boxes. The sample controls preview shows the effect of this.

The sample controls panel displays a preview of how a pull-down list (combo box), edit field, memo panel/window and check box appear, as specified by the user.

7.3.4 Page Controls

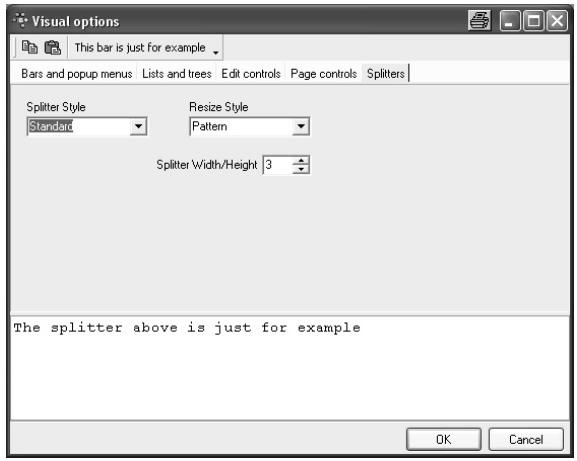


- **Backcolor delta** - this option alters the contrast shade of those page tabs currently in the background. The default value is 20; any changes to this value can be previewed immediately by observing the Visual Options Editor's own page tabs.
- **Multiline page controls** - when checked, this options allows page tabs (or page controls) to be placed over more than one line. This saves the user the necessity of sliding from left to right, in order to find the page he needs. The effect of this option can most easily be viewed in the DB Explorer. Usually the DB Explorer width is limited, in order to allow sufficient space in the main working window. It is therefore often the case that only a small number of the DB Explorer page tabs are visible, and it is necessary to move from left to right before opening, for example, the Windows page. Using this option, the page tabs are displayed over two rows, enabling the user to simply click on the page he needs.

7.3.5 Splitters

A splitter is a moveable line, dividing a child window or editor into two panels.

The Splitters page enables the user to specify the appearance of all IBExpert splitters:

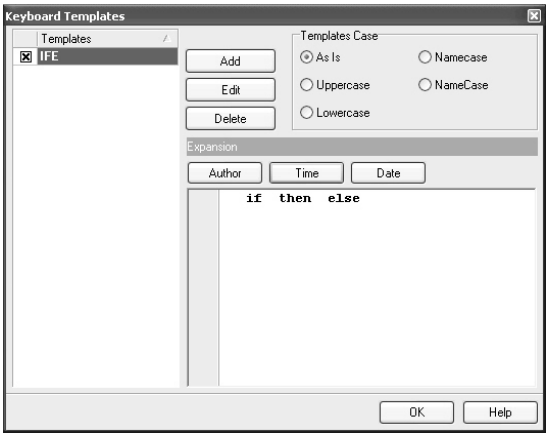


Available options include the following:

- **Splitter Style** - the options offered here include *Standard* and *Netscape*. The Netscape style includes a centered strip (if the splitter width is sufficient, directional arrows are visible). The user simply needs to click on this strip to move the splitter up or down (or left or right if the splitter is vertical), thereby reducing the size of one panel or window and simultaneously increasing the size of the second panel or window. It is also possible to manually adjust the splitter position using drag 'n' drop. When using the standard style the only way to move the splitter is by using drag 'n' drop.
- **Resize Style** - the options offered here include *None*, *Line*, *Update* and *Pattern*. The effects of these options can be viewed by dragging and dropping the sample splitter.
- **Splitter Width/Height** - the effects of any alterations here can be viewed immediately on the sample splitter, displayed in the lower half of this page.

7.4 Keyboard Templates

This can be found under the IBExpert Options menu. It can be used to customize and standardize typing abbreviations for frequently used typical statements, thus increasing efficiency.



For example using the ADD button, enter the shortcut name **IFE**. The full phrase should then be defined in the Expansion panel, in the case of our example, **IF THEN ELSE** (see above).

After confirming the shortcut entry go back to the SQL Editor Edit page, type "ife" and press the space bar. It is automatically expanded to an "if ... then ... else ..." statement.

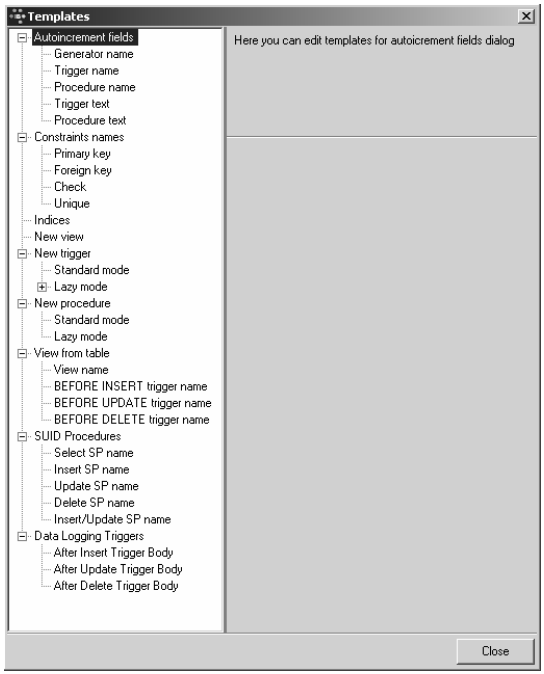
Templates can be added or selected templates edited and deleted as wished. Templates can also be simply deactivated (instead of deleted), by clicking on the flagged checkbox to the left of the template name. To reactivate a deactivated template, simply check the box again.

Further attributes such as Templates Case can also be specified in this editor. Available options include *As Is*, *Uppercase*, *Lowercase*, *Namecase*, and *NameCase*.

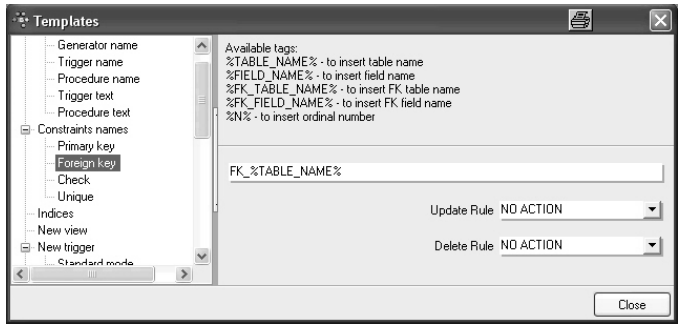
A further feature allows the user to insert author, date and time fields automatically and rapidly, with a simple button click. For example, the abbreviation **ME**, with the expansion **/* #author #date */** results in an simple documentation comment at the beginning of all SQLs listing author and date (i.e. `/* SYSDBA 08/07/2003 */`) simply by typing ME!

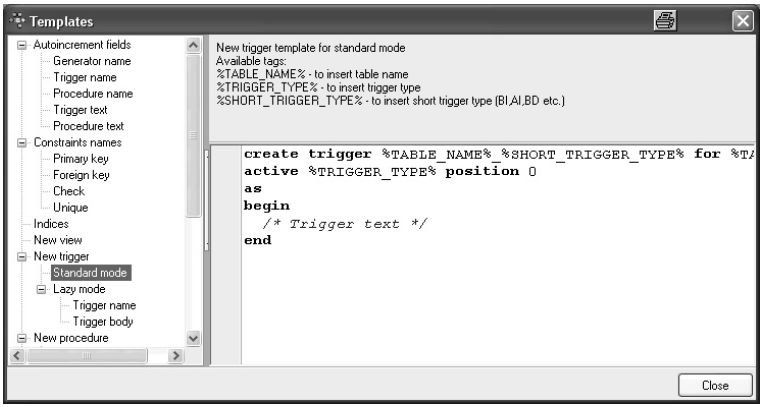
7.5 General Templates

General Templates can be found under the IBExpert menu Options. This can be used to standardize and automate the naming conventions of new database objects, and in some cases, to edit SQL code templates for creating some of these objects.



Below are a couple of illustrations of such templates.





Templates for data logging triggers were added in version 2004.6.17. Please refer to Log Manager for further information.

7.6 Object Editor Options

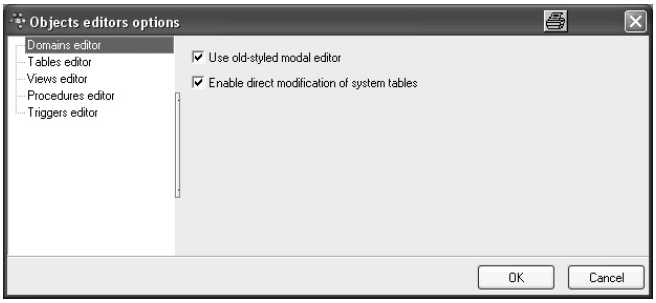
Object Editor Options can be found in the IBExpert Options menu. It opens an Objects Editors Options dialog, which enables users to customize certain database object editors. It is possible, for example, to specify which page should be active, when the Table Editor or View Editor is opened, or specify the standard editor mode in the Procedure Editor or Trigger Editor, and more.

It includes options for the following Editors:

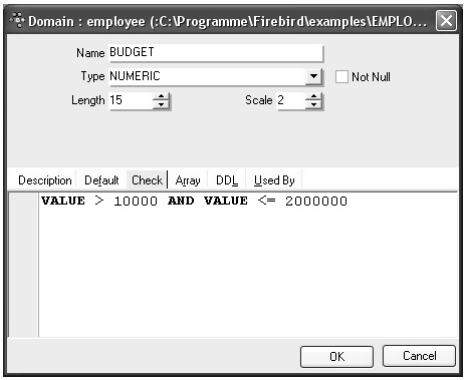
- domains,
- tables,
- views,
- procedures,
- triggers.

7.6.1 Domains Editor Options

The Domains Editor Options page offers the following two options:



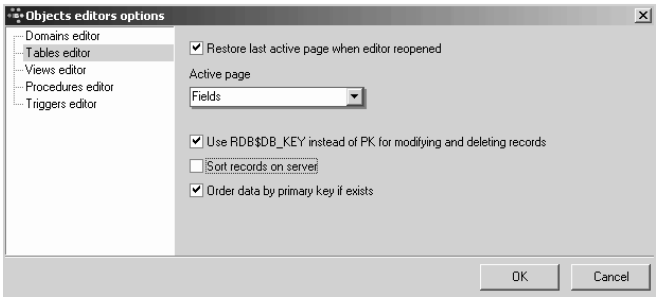
- **Use old-styled modal editor** - when checked, this replaces the current Domain Editor with the old-style editor from earlier versions of IBExpert:



- **Enable direct modification of system tables** - for reasons of security, it is wise not to check this item, unless the SYSDBA, administrator or database owner really need to make changes to any of the system tables.

7.6.2 Tables Editor Options

The Tables Editor Options page offers the following options:



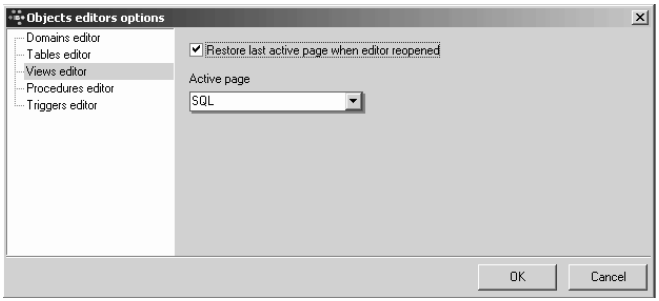
- **Restore last active page when editor reopened** - checking this options results in the last active page remaining the active page, when the editor is reopened.
- **Active page** - offers a choice of all available pages in the Table Editor, i.e. Fields, Constraints, Indices, Dependencies, Triggers, Data, Description, DDL, Grants. This option does not function if the "Restore last active page when editor reopened" option is checked.
- **Use RDB$DB_KEY instead of PK for modifying and deleting records** - RDB$DB_KEY is an internal system field. Every single data set in the database has one of these system keys (a binary column is inserted by InterBase/Firebird for this purpose into each table). It is always unique, and can - in certain cases be very useful. For example, if a developer has created tables in his database, with no primary key, and a particular table column contains the name Miller twice, it is only possible, using SQL, to delete either both data sets or none. RDB$DB_KEY is a possi-

bility to clearly identify individual data sets, and prevent multiple data records accidentally being deleted.

- **Sort records on server** - records may be sorted in the client memory, by simply clicking on a table column header, without running a new SELECT. If the data is to be sorted on the server, a new SELECT statement is required. This is often necessary with large data quantities as the client memory is insufficient.
- **Order data by primary key if exists** - a further sorting option for data.

7.6.3 Views Editor Options

The Views Editor Options page offers the following options:

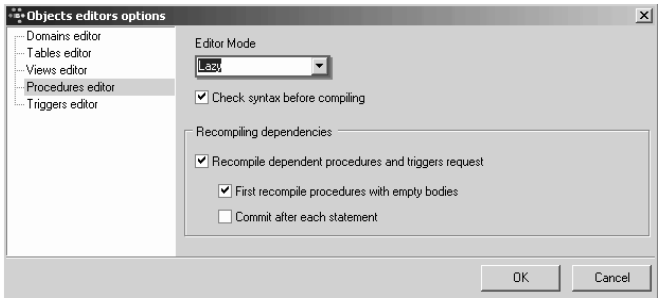


Restore last active page when editor re**opened** - checking this options results in the last active page remaining the active page, when the editor is reopened.

Active page - offers a choice of all available pages in the View Editor, i.e. SQL, Fields, Dependencies, Triggers, Data, Description, Grants, DDL, Version History, Recreate Script, Plan Analyzer. This option does not function if the "Restore active page when editor reopened " option is checked.

7.6.4 Procedures Editor Options

The Procedures Editor Options page offers the following options:



- **Editor Mode** - a default editor mode can be specified here; either *Lazy Mode* or *Standard*.

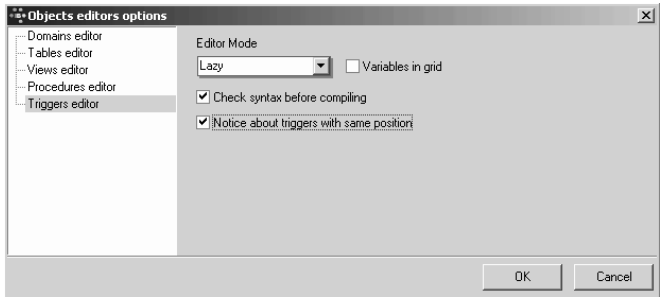
- **Check Syntax before compiling** - here the syntax is first checked locally for any errors, before sending the SQL to the server. This is quicker than sending everything to the server, which will then need to stop and return any eventual errors.

A number of Recompiling Dependencies are also offered:

- **Recompile dependent procedures and triggers request** - this option provides a reminder, asking whether procedures depending upon the amended procedure, should also be recompiled.
- **First recompile procedures with empty bodies** - this option compiles the procedure body source code after the procedure has been compiled, in order to avoid invalid references within the procedures. As soon as one stored procedure has been made dependent on another, procedures are automatically compiled in this way.
- **Commit after each statement** - allows procedures to be compiled step by step, in order to determine where exactly an error lies.

7.6.5 Triggers Editor Options

The Triggers Editor Options page offers the following options:



- **Editor Mode** - a default editor mode can be specified here; either *Lazy Mode* or *Standard*.
- **Variables in grid** - when working in lazy mode, all variables are displayed in a table.
- **Check Syntax before compiling** - here the syntax is first checked locally for any errors, before sending the SQL to the server. This is quicker than sending everything to the server, which will then need to stop and return any eventual errors.
- **Notice about triggers with same position** - if two triggers are both specified the same position, InterBase/Firebird allows this. However InterBase/Firebird chooses which trigger comes first purely by chance. This is therefore a useful warning, just in case two triggers have accidentally been given the same position number.

8 IBExpert Tools Menu

The IBExpert Tools menu offers an extensive range of tools to aid database administration, maintenance and manipulation.

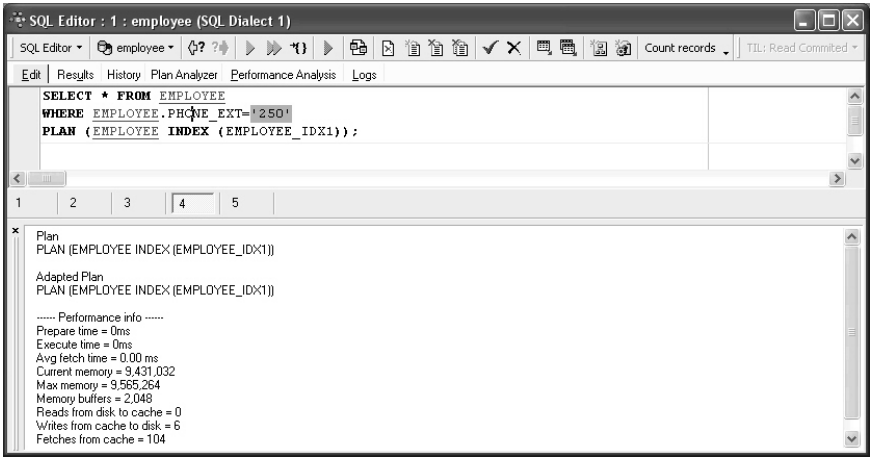
8.1 SQL Editor

The SQL Editor is an IBExpert tool which simplifies the input of SQL commands. It is used to create and execute SQL queries and view and analyze the results.

The SQL Editor is a vital part of IBExpert. As a rule, all work on a database is performed using SQL. The SQL Editor allows you to execute DML and DDL statements, analyze query plans and query performance, move data between databases, export query results into many formats, create views and stored procedures from SELECT etc.

The SQL Editor can be started by selecting the IBExpert Tools menu item, SQL Editor, clicking the respective icon in the Tools toolbar, or using [F12]. This cleans the active SQL window for new input. An additional SQL Editor can be opened using Tools / New SQL Editor or [Shift + F12].

When creating stored procedures or triggers using the DB Explorer menu item New Procedure or New Trigger, an SQL Editor window is also generated. As these editors offer certain additional features (such as lazy mode, debugger), please refer to stored procedure or trigger for specific details.



The SQL Editor can be used together with the DB Explorer (e.g. table fields can be marked and moved from the SQL Assistant into the SQL Editor using drag 'n' drop).

The SQL Editor is intended for the execution of single commands. The Script Executive should be used for more complex scripts.

More than seven tables should not be incorporated into an SQL, as InterBase/Firebird would need too much time to analyze the indices to determine the most efficient solution. Therefore the database server simply starts somewhere, which leads to slow and

lengthy queries. Since Firebird 1.5 the optimizer has been considerable improved when working with several tables.

Ten SQLs can be incorporated into a stored procedure.

A stored procedure or view can be created from the current query directly in the SQL Editor, using the respective icons.

The Tools / SQL Editor option includes the following:

- Edit window (& Results)
- History
- Plan Analyzer
- Performance Analysis
- Logs
- Visual Query Builder

The Edit window is the main input window for all SQL transactions. The History page lists previous queries. The Plan Analyzer provides information in the lower panel in a tree structure with statistics. For details see Plan Analyzer.

The Performance Analysis shows how much effort was required by InterBase/Firebird to carry out this query. For details see Performance Analysis.

For those not yet competent in SQL, the Visual Query Builder is there to make life easier! It is ideal for the beginner, although somewhat limited for more advanced work; more complex queries would need to be performed in the SQL Editor or perhaps even the Script Executive.

To customize the SQL Editor according to your wishes, please refer to Options / Editor Options and Environment Options / SQL Editor.

8.1.1 Query

A query is a qualified search for information held in the data sets stored in the database. The qualification can determine which tables should be searched, which range of values for specified columns should be included, etc. etc.

For an overview of the conditions that are available in SQL, please refer to Comparison Operators.

`SUM` (total), `MIN` (minimum), `MAX` (maximum), `AVG` (average), and `COUNT` are aggregates that can also be used, for example, when the sales department needs to know how many orders are still open or the minimum/maximum or average order value in the past year.

A query on one or more tables produces a set of rows that is itself a table, subject to all the rules for tables in a relational database. This is known as *Closure*. InterBase/Firebird fully supports closure.

Regularly performed queries, such as a list of all unpaid invoices, or a list of all delivery notes that have gone out in the last week, can be stored as procedures.

Queries are optimized by InterBase/Firebird. The optimizer chooses which indices should be used, in order to perform the query as quickly and simply as possible.

8.1.2 SQL Structured Query Language

SQL is the abbreviation for Structured Query Language. It is used to communicate with a relational database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. It serves to define, manipulate, find and fetch data in a database.

InterBase and Firebird conform to the international industrial standards SQL '92.

Furthermore InterBase and Firebird offer a series of additional SQL enhancements, such as generators, triggers and stored procedures, allowing a more extensive modeling and manipulation of data. These enhancements are either based on the ANSI SQL2 Standard or already comply with the outline of the ANSI/ISO SQL3 standards.

8.1.3 SQL Editor Menu

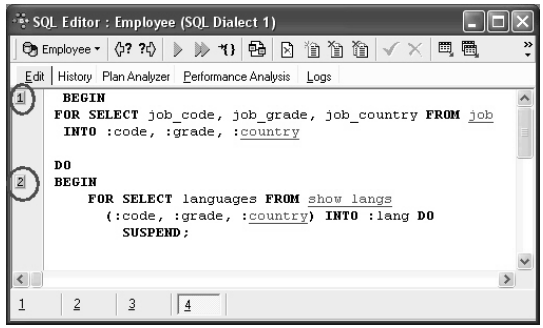
In addition to the icons in the SQL Editor toolbar, the SQL Editor has its own menu, opened using the right mouse button:

The most important menu items are detailed in this section or can be found in the IBExpert Edit menu.

Bookmark

Bookmarks are useful for flagging sections of long SQL scripts. They are purely an aid for the user and have no influence upon the SQL script or database whatsoever.

Bookmarks can be set in the SQL Editor, and in the SQL window in the Stored Procedure and Trigger Editors, using the mouse right-click menu item Toggle Bookmarks. Alternatively they can also be specified using the key combination [Ctrl + Shift + 0-9].



The bookmarks themselves can be seen in the left margin of the SQL script. They can be numbered as wished.

The mouse right-click menu item Go To Bookmarks can be used to spring from bookmark to bookmark. Alternatively the key combination [Ctrl + 0-9] can be used.

Bookmarks can be removed by simply unchecking those bookmarks listed in the Toggle Bookmarks menu.

Copy Text as RTF

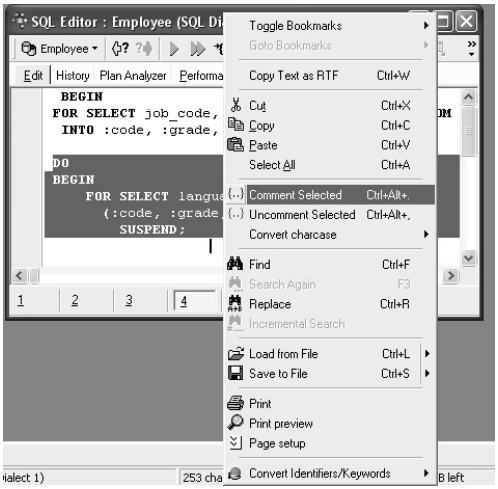
In order to copy a script, including the text formats (color, bold, indent etc.), select the script or script parts to be copied, right-click and choose the menu item Copy Text as RTF (or [Ctrl + W]).

This feature is ideal, for example, for documentation purposes.

Comment Selected/Uncomment Selected

In certain situations it may be necessary to disable certain commands or parts thereof. This can be easily done without it being necessary to delete these commands. Simply select the rows concerned in the SQL Editor, right-click and select the menu item Comment Selected (or using [Ctrl + Alt + .]). This alters command rows to comments.

The commented text can be reinstated as SQL text by using the right mouse key menu item Uncomment Select (or using [Ctrl+ Alt + ,]).



This is particularly useful, when attempting to discover error sources or performing parts of standard selects.

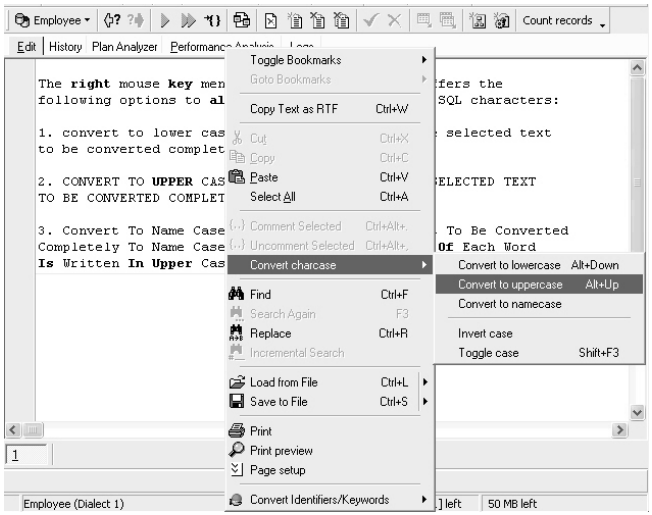
Convert Charcase

The mouse right-click menu item Convert Charcase offers the following options to alter the appearance of the SQL characters:

1. **Convert to lower case** [Alt + Down]: allows the selected text to be converted completely to lower case.

2. **Convert to upper case** [Alt + Up]: allows the selected text to be converted completely to upper case.

3. **Convert to name case**: allows the selected text to be converted completely to name case, i.e. the initial character of each word is written in upper case, the remaining characters in lower case.



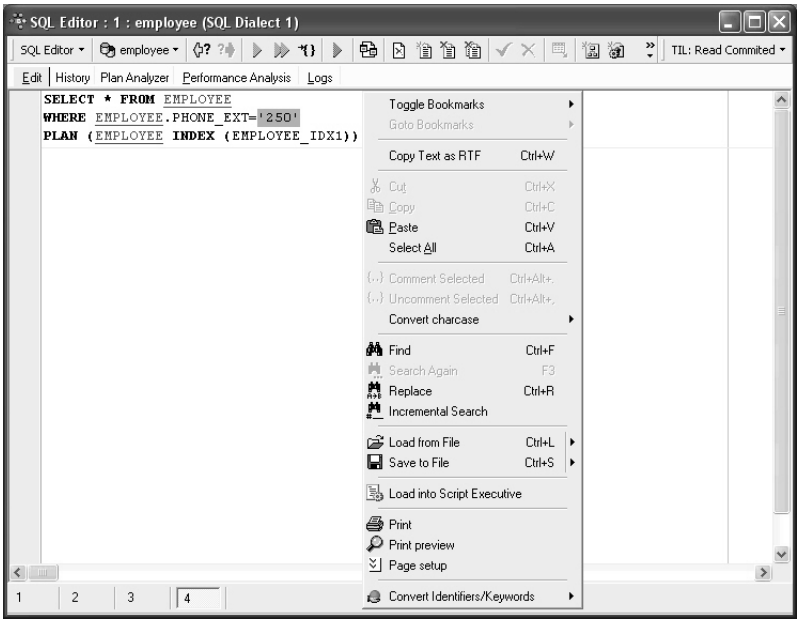
4. **Invert case**: switches between upper and lower case.

5. **Toggle case** [Shift + F3]: switches between upper, lower and name case.

8

8.1.4 (1) Edit

The Edit page appears as the active window when the SQL Editor is opened. It is the main input window for SQL commands.



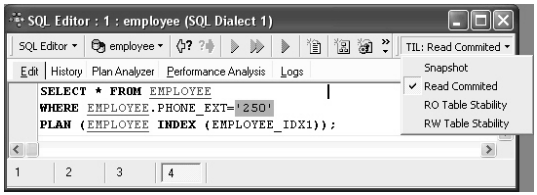
The SQL Editor toolbar and right mouse button menu (SQL Editor menu) offer a wide range of operations.

The lower status bar displays the number of open queries, allowing these to be quickly loaded in the active editing window by clicking on the respective buttons. Alternatively [Ctrl + N] can be used to load the next statement or a new window can be loaded using [Shift + F12] (menu item New SQL Editor).

The SQL Editor allows you to prepare statements and get a statement plan without executing it by using [Ctrl+F9]. It is also possible to prepare only a part of the statement. Just select the corresponding part of the statement and press [Ctrl+F9] or click the Prepare button on the SQL Editor toolbar.

It is also possible to execute a part of statement. Just select the corresponding part of the statement and use [F9] or the corresponding icon

It is so easy to execute and analyze statements (or parts of them) before finally committing. Since version 2.5.0.61 there is the added possibility to quickly change the Transaction Isolation Level (TIL) for a separate SQL Editor. There is a corresponding button on the SQL Editor toolbar which allows you to choose one of the following isolation levels: *Snapshot*, *Read Committed*, *Read-Only Table Stability* and *Read-Write Table Stability*.



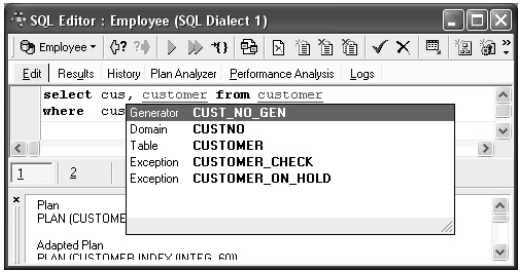
Objects and fields can be simply and quickly dragged and dropped from the DB Explorer and SQL Assistant into the Edit input page. Since version 2004.2.26.1 this has been greatly improved. When an object node(s) is dragged from the DB Explorer or SQL Assistant, IBExpert will offer various versions of text to be inserted into the code editor. It is also now possible to customize the highlighting of variables. Use Options / Editor Options / Colors to choose color and font style for variables.

A Code Insight system is included to simplify command input, i.e. when the first word characters are typed in the SQL text editor, alternatives for word completion are offered in a pop-up list. Database objects are underlined for easy recognition.

There is also a wide range of keyboard shortcuts available in the SQL Editor, e.g. [Ctrl + Alt + R] produces a list of all triggers which can be selected using the mouse or directional keys (insert using the [Tab] key). To view the full list call the Localizing Form using [Ctrl + Shift + Alt + L].

Hyperlinks allow you to quickly reference database objects if necessary.

IBExpert version 2004.04.01.1 includes added support for the EXECUTE BLOCK statement (Firebird 2). And since IBExpert version 2005.02.12.1 there is added support for the INSERTEX command (for importing data from a comma-separated values file).



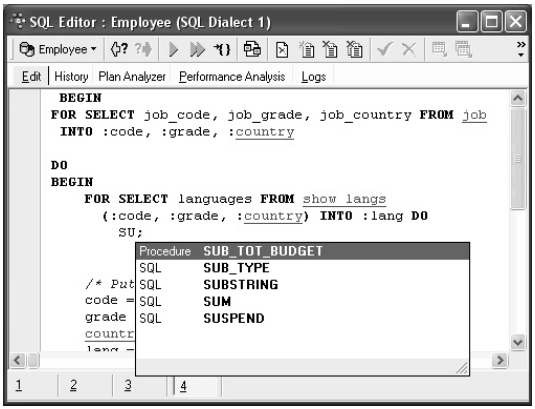
A results page appears after query execution [F9], displaying the returned data.

Code Insight

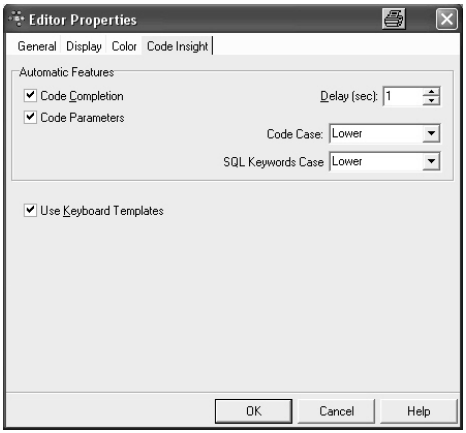
A Code Insight system is included in the IBExpert SQL Editors to simplify command input. When the first word characters are typed in the SQL text editor, alternatives for word completion are offered in a pop-up list. Simply click the required word, or alternatively select the word using the directional keys and insert using the [Tab] key.

Alternatively the key combination [Ctrl + space bar] can be used to explicitly activate the Code Insight dialog. Database objects are underlined for easy recognition. If you

wish to view a list of parameters/variables, use the key combination [Ctrl + Alt + L]. This solution has been offered as it would otherwise be necessary to parse the editor each time before the Code Insight list appears.



Using the IBExpert menu item Options / Editor Options / Code Insight, this can be individually adapted as wished.



Further abbreviations and definitions can be defined by the user if wished, using the IBExpert menu option Options / Keyboard Templates.

Hyperlinks

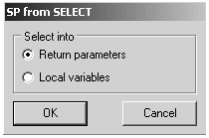
As with all IBExpert editors, the SQL Editor even offers hyperlinks. When an object name is written on the Edit page, the respective object editor can be opened.

A hyperlink is an element in an electronic application or document that links to another place in the same application/editor/text or to an entirely different editor/text. Typically, you click on the hyperlink to follow the link. Hyperlinks are the most essential ingredient of all hypertext systems, including the World Wide Web.

To switch off the automatic hyperlink option, or to change its appearance, please refer to Environment Options / Editor Options.

Create view or procedure from SELECT

If you wish to create a view or procedure from a valid SELECT statement in the SQL Editor, simply use the relevant icon to the right of the toolbar. It is possible to create a view or a procedure from an SQL statement without typing all variables and parameters.



When creating a procedure from a select, it is necesary to specify whether to select into return parameters or local variables.

Create temp tables

If you want to store the result of an SQL statement in a new table, just type

```
Insert into New_Table Select * from Old_Table
```

The new table is automatically created if it does not already exist.

Copy data from one database to another

If you want to copy data from a Source_Alias to a Destination_Alias, simply open an SQL Editor in the Source_Alias and type:

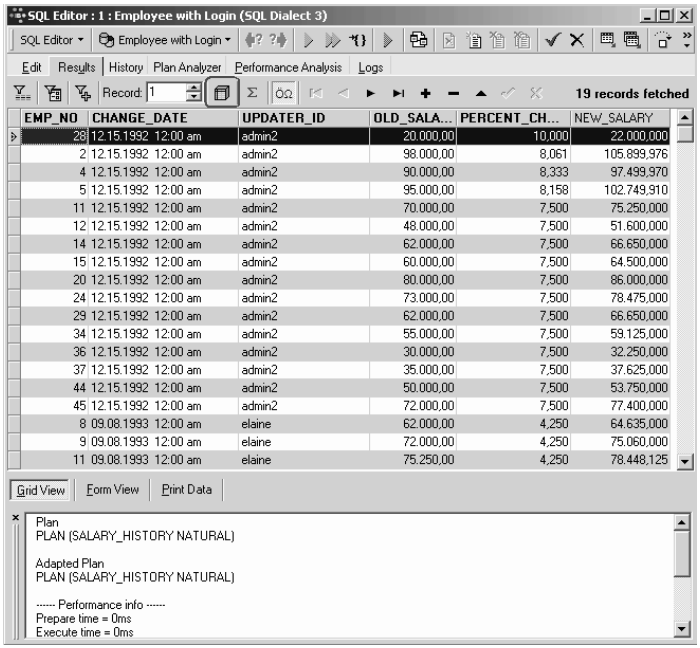
```
Insert into [Destination_Alias].NEW_Table select * from Old_Table
```

8.1.5 (2) Results

The results page is automatically generated as soon as a query is executed. There are three modes of view:

1. Grid View - all data is displayed in a grid (or table form). By clicking on the column header the result set can be sorted (in ascending or descending order) according to that column.

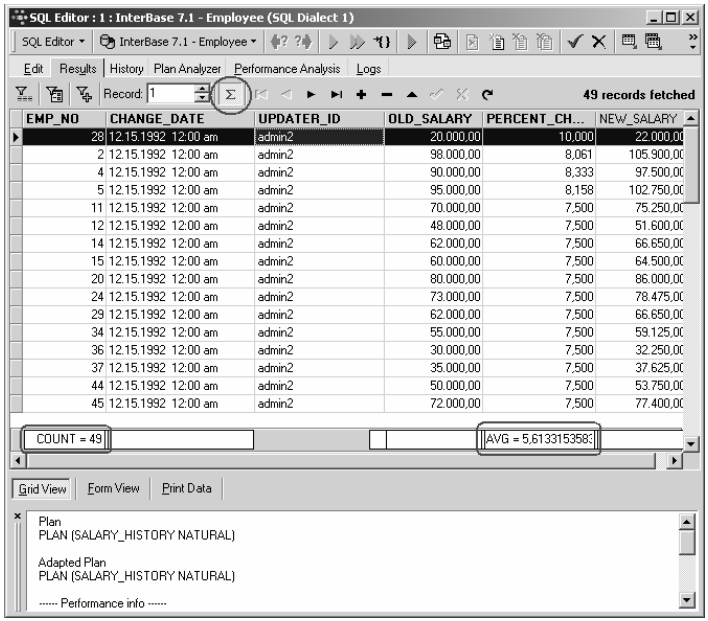
There are many options to be found under Options / Environment Options / Grid, which allow the user to customize this grid view. Under the IBExpert menu item Register Database or Database Registration Info there are additional options, for example, Trim Char Fields in Grids.



Results can only be edited in the Grid View if it is a live result set. Since version 2003.12.18.1 it is possible to copy selected record(s) to clipboard as `UPDATE` statement(s). This will only work if there is a live query with a primary key. Since version 2004.1.22.1 mandatory (`NOT NULL`) fields are now highlighted while working with live queries. Captions of `NOT NULL` fields are in bold.

A new feature in IBExpert version 2004.10.30.1 is the OLAP and data warehouse tool, Data Analysis, opened using the Data Analysis icon (highlighted in red in the above illustration).

Since IBExpert version 2004.8.5.1 there is the added option to calculate aggregate functions (`COUNT, SUM, MIN, MAX, AVG`) on numeric and datetime columns. Simply click "Show summary footer" button on the toolbar of the data view to display the summary footer:

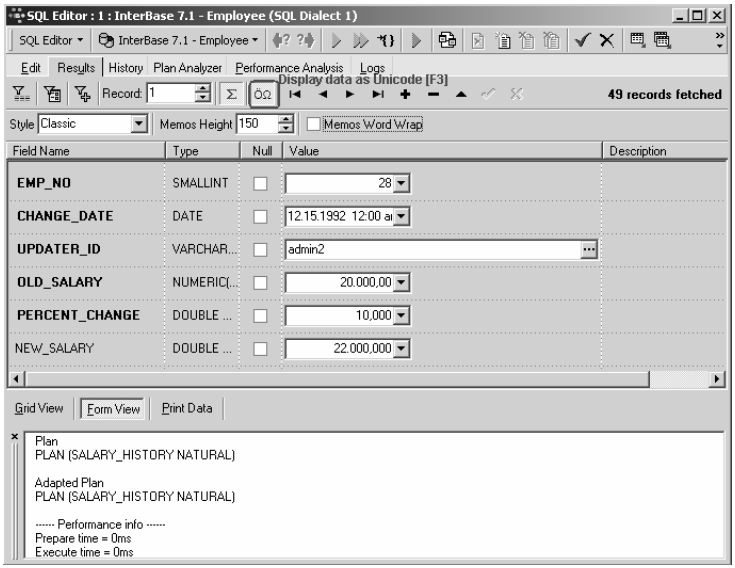


Then simply select the aggregate function from the pull-down list for each numeric/datetime column as required.

IMPORTANT: all calculations are done on the client side so do not use this feature on huge data sets with millions of records because IBExpert will fetch all records from the server to the client in order to calculate aggregates.

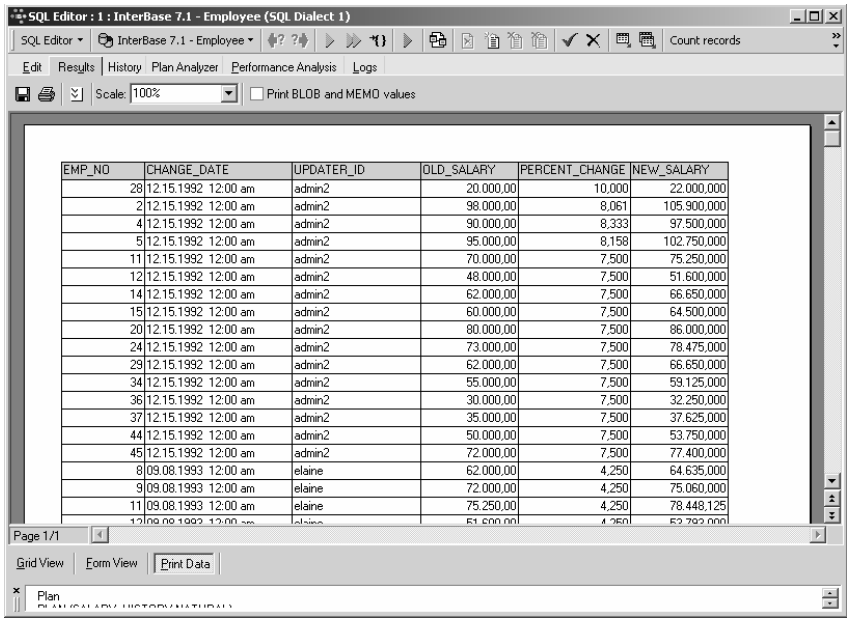
Since IBExpert version 2004.8.26.1 it is also possible to display data as Unicode. Simply click the relevant icon or use [F3] (see illustration below). It is not possible to edit the data directly in the grid. To edit data in unicode, use the Form View or modal editor connected with string cell.

2. Form View - one data set is displayed at a time in a form.



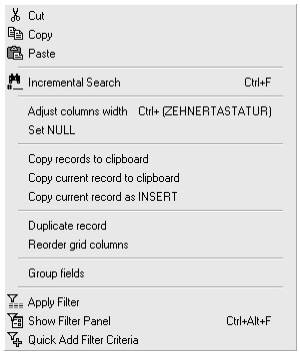
New to version 2004.8.26.1: The Form View has been completely redesigned. It now also displays field descriptions. It is also possible to select alternative layouts (*classic* or *compact*), the compact alternative for those who prefer a more compact and faster interface. Visual options now also include specification of *Memo Height* and *Memo Word Wrap*.

3. Print Data - displays data in WYSIWYG mode, the data can be either saved to file as a simple report or printed.



EMP_NO	CHANGE_DATE	UPDATER_ID	OLD_SALARY	PERCENT_CHANGE	NEW_SALARY
28	12.15.1992 12:00 am	admin2	20.000,00	10,000	22.000,000
2	12.15.1992 12:00 am	admin2	98.000,00	8,061	105.900,000
4	12.15.1992 12:00 am	admin2	90.000,00	8,333	97.500,000
5	12.15.1992 12:00 am	admin2	95.000,00	8,158	102.750,000
11	12.15.1992 12:00 am	admin2	70.000,00	7,500	75.250,000
12	12.15.1992 12:00 am	admin2	48.000,00	7,500	51.600,000
14	12.15.1992 12:00 am	admin2	62.000,00	7,500	66.650,000
15	12.15.1992 12:00 am	admin2	60.000,00	7,500	64.500,000
20	12.15.1992 12:00 am	admin2	80.000,00	7,500	86.000,000
24	12.15.1992 12:00 am	admin2	73.000,00	7,500	78.475,000
29	12.15.1992 12:00 am	admin2	62.000,00	7,500	66.650,000
34	12.15.1992 12:00 am	admin2	55.000,00	7,500	59.125,000
36	12.15.1992 12:00 am	admin2	30.000,00	7,500	32.250,000
37	12.15.1992 12:00 am	admin2	35.000,00	7,500	37.625,000
44	12.15.1992 12:00 am	admin2	50.000,00	7,500	53.750,000
45	12.15.1992 12:00 am	admin2	72.000,00	7,500	77.400,000
8	09.08.1993 12:00 am	elaine	62.000,00	4,250	64.635,000
9	09.08.1993 12:00 am	elaine	72.000,00	4,250	75.060,000
11	09.08.1993 12:00 am	elaine	75.250,00	4,250	78.448,125

The results page also has its own right-click menu, which can be used to perform numerous operations upon the resulting data (please refer to Table Editor / Data for more information).



Filter Panel

In the SQL Editor Results dialog and in the Table Editor Data (grid and form view) dialog it is possible to work with filters, enabling the developer to add/delete criteria and filters directly in the data sets resulting from the executed SQL.

The Filter Panel is opened using the Show Filter Panel icon:

or [Ctrl + Alt + F]. A new two-part window appears. This can be split horizontally or vertically as the user wishes by clicking on the Vertical Layout icon or using the key combination [Shift + Ctrl + L].

New filter criteria can be added by placing the cursor on the field, where a filter is to be inserted and using the +-button or [Ins] key. For the deletion of filters use the - button or [Ctrl + Del] key combination. Select the comparison operator from the pull-down list adjacent to the list of field names and specify the desired value(s).

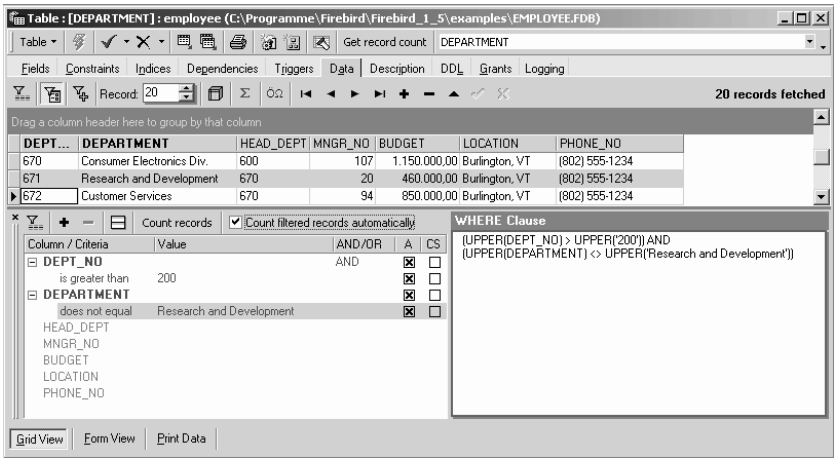
When a second field is marked and a new filter for this field is added, the *AND* column is automatically filled (default is *AND*, may be altered to *OR* if wished, using the space bar or mouse click).

The two right-hand columns provide check box options, to specify whether a filter should be active or not (column A), and to specify whether case-sensitivity is of importance (CS column).

The second panel displays the `WHERE` clause that has just been specified.

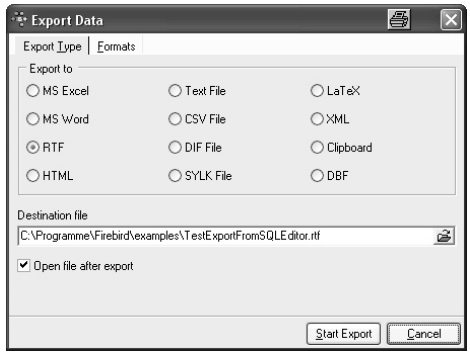
Since IBExpert version 2005.02.12.1 there is now the possibility to recalculate the number of filtered records automatically when the filter condition is changed.

The filter area can be deactivated by re-clicking the Show Filter Panel icon or [Ctrl + Alt + F].



Export Data

The results can be exported using [Ctrl + E] or the respective SQL Editor toolbar icon, which opens the Data Export dialog:



Supported formats are *Excel, MS Word, RTF, HTML, Text, CSV, DIF, SYLK, LaTex, SML, Clipboard* and *DBF*. Depending on the format, further options can be specified on the second or third pages, Formats and Options, specific to the export type. If you want to open the file directly after creating, use a typical file extension such as *.xls, and check the Open File After Export box.

The Format page can be used to specify the following formats:

- currency,
- float,
- integer,
- date time,
- date,
- time.

Using the right-hand icon in the SQL Editor toolbar or Table Editor toolbar (Export data into script) the data can be exported into an insert SQL script (without the blob fields).

Export Data into Script

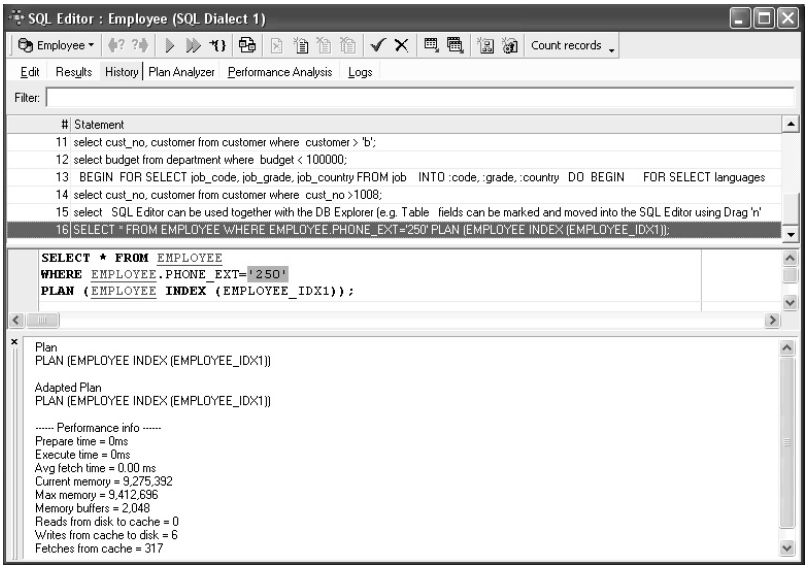
Please refer to Export Data into Script (see Table Editor / Data for further information).

8.1.6 (3) Statements History

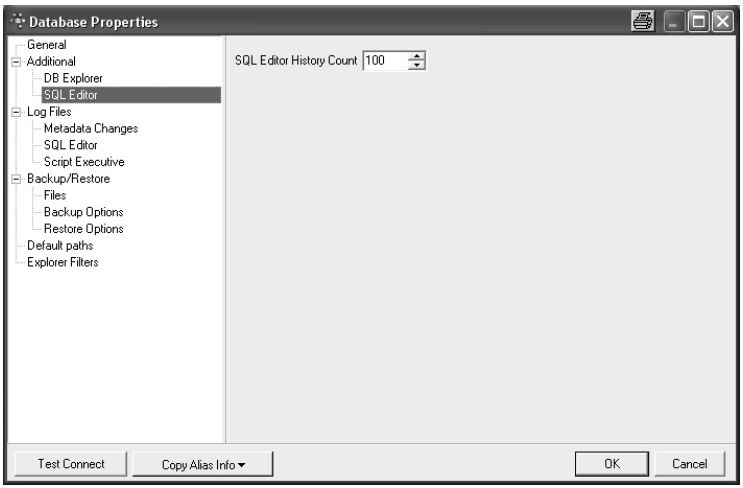
The History page can be found in the SQL Editor, and lists previous SQL queries that have been executed and produced a result (not necessarily committed), along with their performance statistics. This saves having to reenter recurring commands, and offers a concise overview of the individual SQL performances for comparison. All statements are only visible when the same alias is in use.

The middle panel displays the script and the lower panel the SQL plan or error messages.

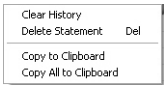
The filter (directly above the statement list) can be used to display only those objects containing the character string entered in the filter, e.g. Find all SQLs containing a `SELECT` or all SQLs containing the `EMPLOYEE` table.



The SQL History lists a record of the last 100 statements. This default quantity of 100 stored statements can be altered by using the IBExpert menu item Database or the DB Explorer right mouse button menu: Database Registration Info / Additional / SQL Editor, where the SQL Editor History Count can be specified as wished.



The SQL History list can be streamlined, as and when required, by deleting individual list entries, using the right mouse button.

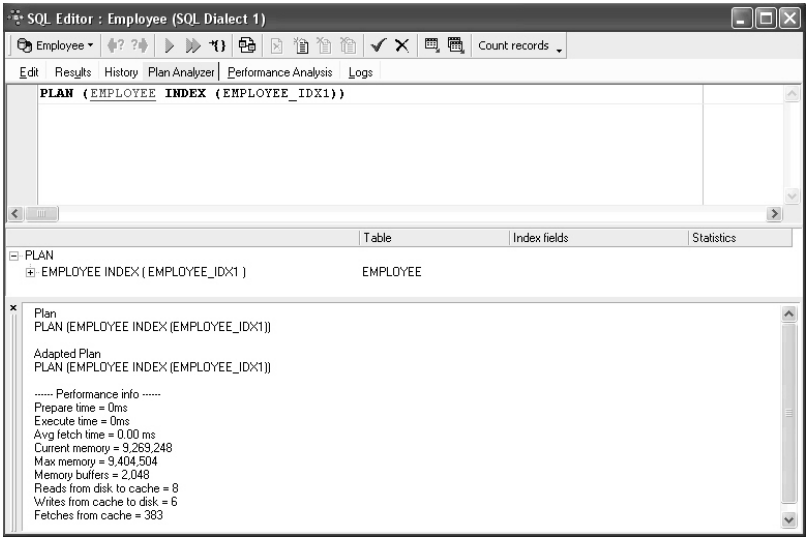


This menu also allows single statements (or all) to be copied to clipboard.

8.1.7 (4) Plan Analyzer

The SQL Editor Plan Analyzer (also a part of the Procedure Editor and Trigger Editor) shows how Firebird/InterBase approaches a query, e.g. with SORTS, JOINS etc, which tables and indices are used. This information is displayed in a tree structure: firstly what and which data quantities, and secondly what is carried out with this data and how.

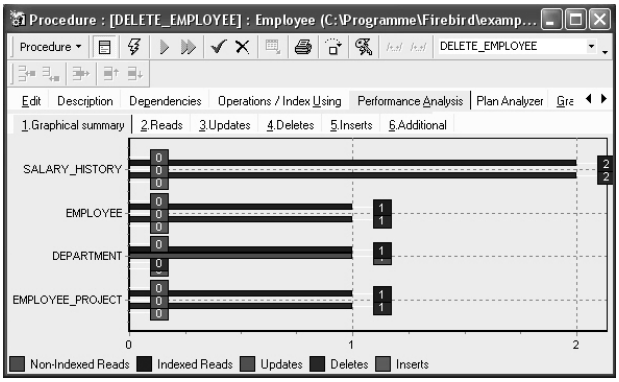
The plan is an InterBase/Firebird description, showing how the optimizer uses tables and indices to obtain the result set. If the word SORT is displayed, you should check whether improvements upon the query or the indices are possible.



The Plan Analyzer provides information in the lower panel in a tree structure with statistics.

8.1.8 (5) Performance Analysis

The Performance Analysis is part of the SQL Builder, Visual Query Builder and Stored Procedure Editors. It displays information showing how much effort was required by InterBase/Firebird to carry out an executed query or procedure. The analysis is performed after a SELECT statement is opened or a stored procedure started.



It is possible to deactivate the Performance Analysis, by checking the Disable Performance Analysis option, found under Database / Register Database *or* Database Registration Info / Additional. This may be desirable, when working remotely with a slow modem connection.

It is however often interesting to know what exactly a procedure or query does and how; and all this can be viewed in the Performance Analysis.

The main advantage here is, of course, the possibility to compare the performance of different queries and procedures.

The performance can be viewed in 6 different ways:

1. Graphical summary
   i) indexed reads
   ii) non-indexed reads
   iii) updates
   iv) deletes
   v) inserts
2. Reads (graphical representation)
3. Updates (graphical representation)
4. Deletes (graphical representation)
5. Inserts (graphical representation)
6. Additional
   i) Enhanced Info
   ii) Query Time
   iii) Memory
   iv) Operations

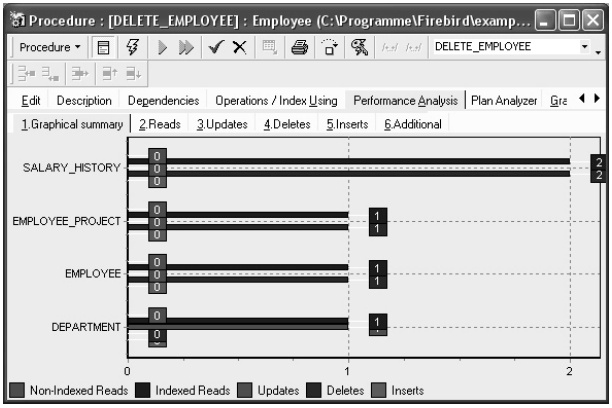
SELECT statements will only have a result on the Reads page, but some stored procedures will have results on all pages.

In the SQL Editor the lower panel displays the query plan, along with a summary of the performance information included under 6. Additional. For further information regarding the query plan, please refer to the Plan Analyzer.

The analysis displayed in 6. Additional can also be documented using the Copy Analysis to Clipboard button.

Graphical Summary

This provides a graphical overview, broken down by the tables involved, of the number of operations performed by the query/procedure, including reads (indexed and non-indexed), updates, deletes and inserts. It shows whether indices have been used indicating the efficiency of the database's indices. The figures displayed refer to the number of data sets.

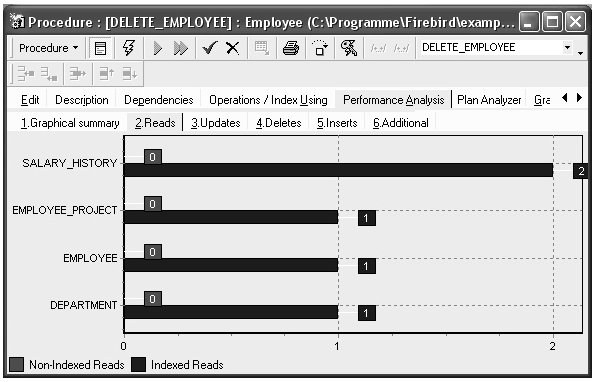


The x-axis lists the names of the tables consulted by the query/procedure, with the number of operations displayed graphically. The color key can be seen below the graphic. The operation types are as follows:

- Non-indexed reads
- Indexed reads
- Updates: The number and type of updating operations.
- Deletes: The number and type of deleting operations.
- Inserts: The number and type of inserting operations.

The graphical information displayed here can also be viewed in tabular format under 6. Additional.

Reads



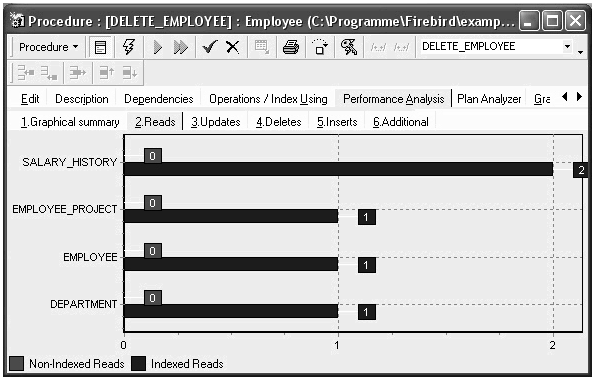
This displays the number and type of reading operations in an executed query/procedure. The figures displayed refer to the number of data sets and are broken down by table into the categories *indexed* and *non-indexed* reads.

Those database indices used to perform an SQL query can be viewed in the SQL Editor in the Performance Analysis query plan.

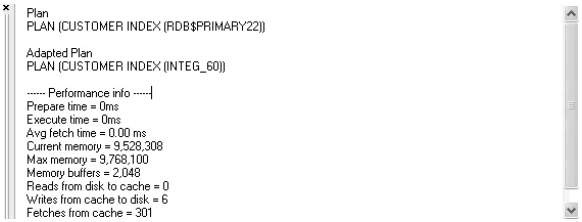
This information can be used to evaluate the efficiency of the database\'s indices.

Indexed Read

Indexed reads are displayed in the Performance Analysis, which can be found in the SQL Editor, Visual Query Builder and Stored Procedure Editors.

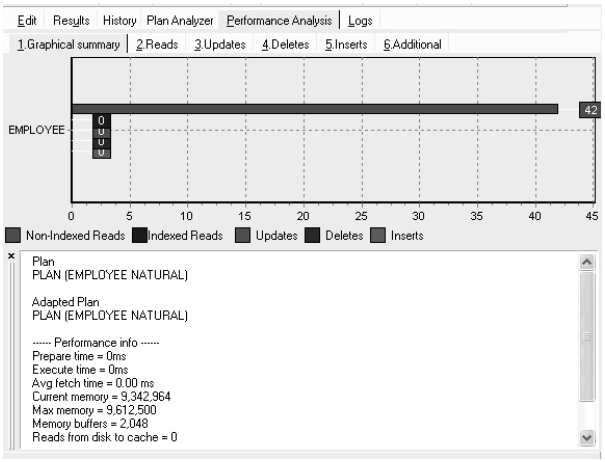


An indexed read indicates that the data was selected by the InterBase/Firebird server using one or more indices (named in the SQL Editor query plan in the lower panel). This results in many cases in a significantly lower number of data sets being consulted than with a non-indexed read, saving both time and memory.



Non–Indexed Read

Non-indexed reads are displayed in the Performance Analysis, which can be found in the SQL Editor, Visual Query Builder and Stored Procedure Editors.

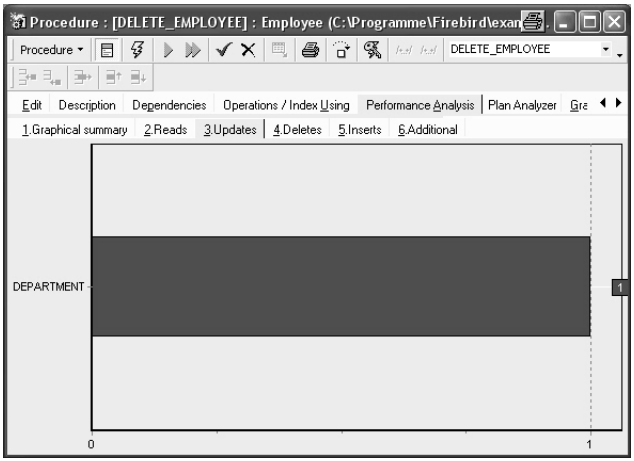


A non-indexed reads indicates that the data was read without the aid of an index. In most situations this can be both time- and memory-consuming. Non-indexed reads always include a large number of data sets, as the server needs to search through the whole table(s) to find the relevant information. All data pages from the corresponding table(s) need to be loaded.

The SQL Editor's query plan shows which tables were read without an index using the term NATURAL.

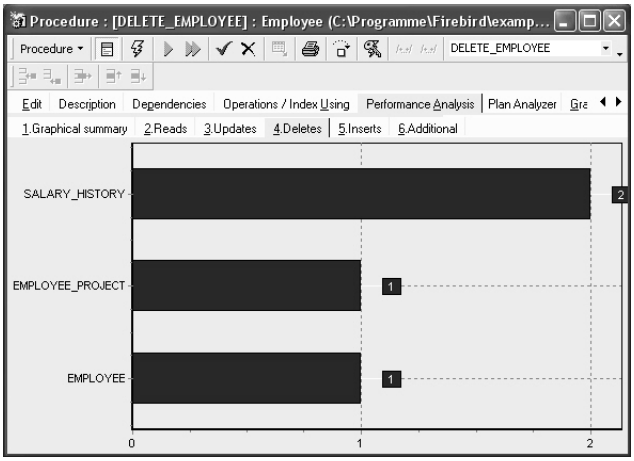
For further information regarding the use of indices, please refer to index.

Updates



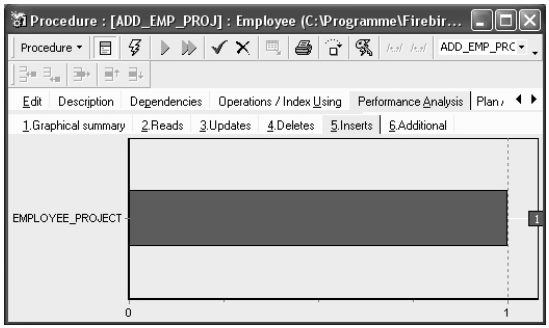
This displays the number and type of updating operations in an executed query/procedure. The figures displayed refer to the number of data sets, broken down by table.

Deletes



This displays the number and type of deleting operations in an executed query/procedure. The figures displayed refer to the number of data sets, broken down by table.

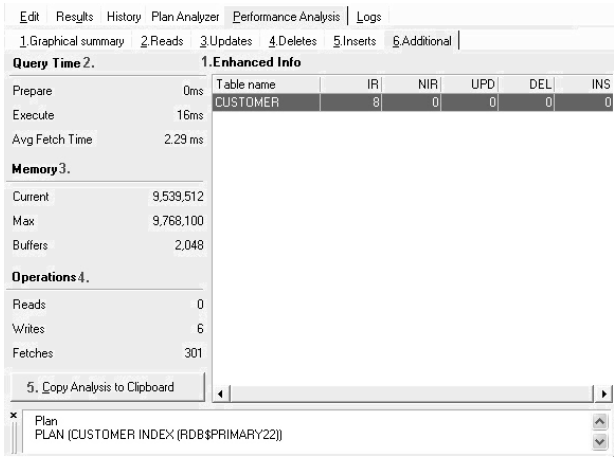
Inserts



This displays the number and type of inserting operations in an executed query/procedure. The figures displayed refer to the number of data sets, broken down by table.

Additional

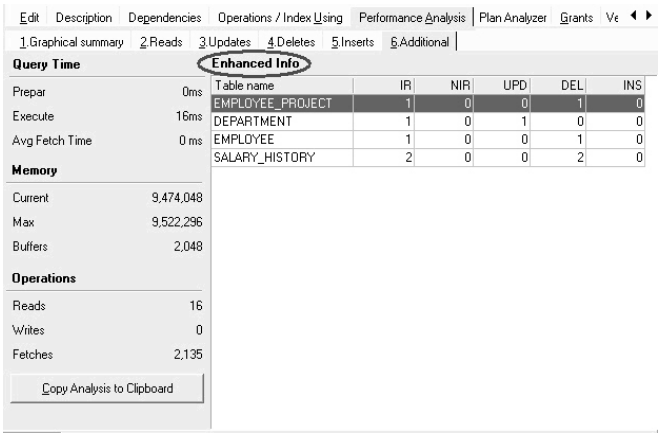
This displays a statistical report. The *Enhanced Info* displays a statistical summary of the information shown in 1. Graphical Summary. Certain additional information, such as query time, memory and operations, is also included in this section.



There is furthermore a Copy Analysis to Clipboard button, to document the statistics if wished.

Enhanced Info

The Enhanced Info displays a statistical summary of the information shown in 1. Graphical summary.



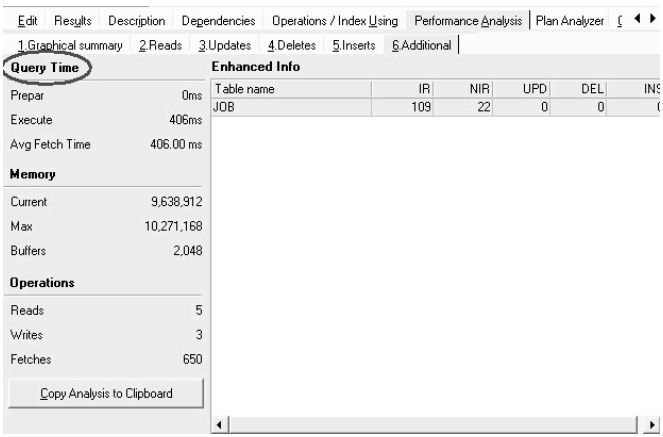
The names of tables consulted during execution of the query/procedure are listed in the first column, with the number of data sets listed according to the following criteria:

- IR = Indexed Read
- NIR = Non-Indexed Read
- UPD = Updates
- DEL = Deletes
- INS = Inserts

The information can be copied to clipboard, if wished, using the Copy Analysis to Clipboard button.

Query Time

Query time shows the time needed to prepare for the execution of the query/procedure, along with the execution time and average fetch time.



Prepare:

This measures the preparation time required by InterBase/Firebird to plan and prepare the query/procedure execution, i.e. from the moment when the source text is sent to the server and is compiled on the server in binary form (decides which indices, tables etc. need to be used to perform the query/procedure).

When a query/procedure is executed a second time, the query time is usually 0 ms, as it has already been prepared.

Execute:

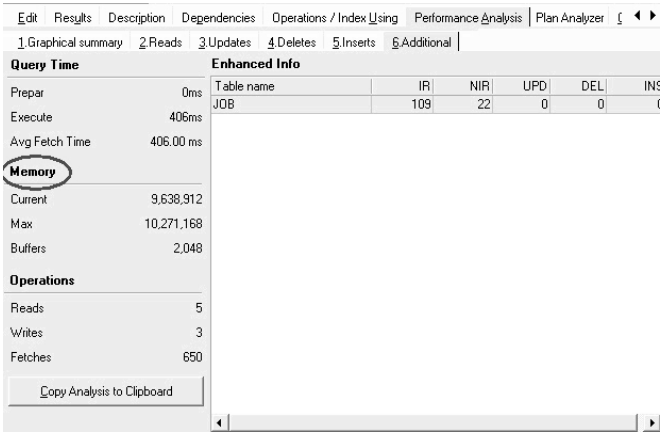
This measures the direct execution time of the command.

Avg fetch time:

This shows the average fetch time pro data set. This figure is calculated based only on those data sets that can be seen in the returns and does not include those that are not yet visible. An optimal analysis can be attained when the query/procedure is executed using [Shift + F9] = Execute and Fetch all.

Memory

This shows the memory development during and following execution of the procedure/query.



Current:

This displays the current memory used by the server.

Max.:

This displays the maximum memory used by the server during execution of the query/procedure.

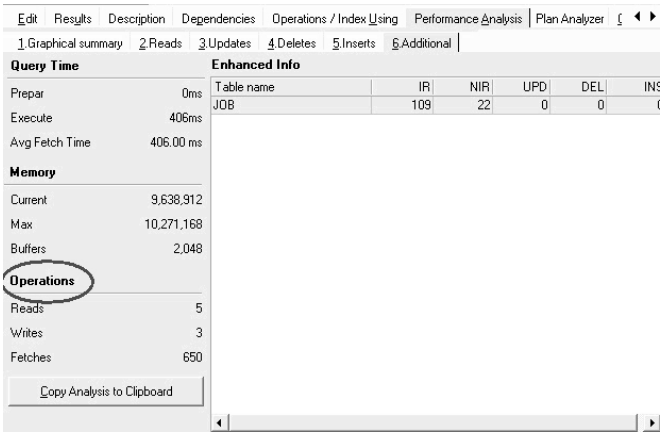
Buffers:

This displays the number of data pages that are being held as cache on the server (from InterBase 6 onwards the standard is 2,048). This can be found in the corresponding configuration file: since Firebird 1.5 it is called `FIREBIRDCONFIG`; in older Firebird versions or InterBase, it is called `IBCONFIG`, found in the main InterBase folder.

This can be altered for the current database if wished, using the IBExpert menu item Services / Database Properties / Buffers. The total KB is calculated according to the current database page size. For an alteration to become effective, it is therefore necessary for all users to disconnect from the database and then reconnect. Buffers are only reserved if they are really necessary for pages loaded from the database file.

Operations

Operations displays the number of data pages that were read from the database file to the memory, written and fetched, while executing the query/procedure.



Reads:

This displays the number of pages read for the executed query/procedure. This is necessary when data sets have to be loaded, that are not already in the memory.

Writes:

This displays the number of pages written while executing the query/procedure. If the total cache buffers are too small to load subsequent pages, it may be necessary for the server to save altered pages to the hard drive, in order to make room for further pages to be loaded. If these values are very high, it may be wise to increase the buffers, providing of course, that physical memory is sufficient.

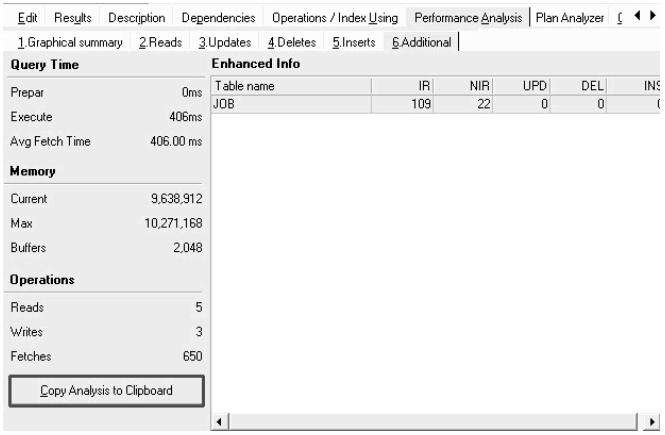
Fetches:

When a query/procedure is started, the command (or series of commands) is sent to the database server. To obtain results, numerous data sets/pages need to be referred to (= fetch), in order to perform the operation. Fetches are, in other words, internal operations performed by InterBase/Firebird in order to successfully execute a query/procedure. This indicates, for example, if deleted data sets in a SELECT are recognized as deleted, they will still appear here in the number of fetches, as the server also searches through those data sets that have been marked as deleted. This can however offer an advantage over the number of indexed and non-indexed reads, as these only display operations on undeleted data sets. If the query is executed again, the result is quicker if the garbage collection is running simultaneously.

Using the Performance Analysis, the number of fetches in data pages could possibly indicate why one query is quicker than another with an equal number of data sets and the same index plan.

Copy Analysis to Clipboard

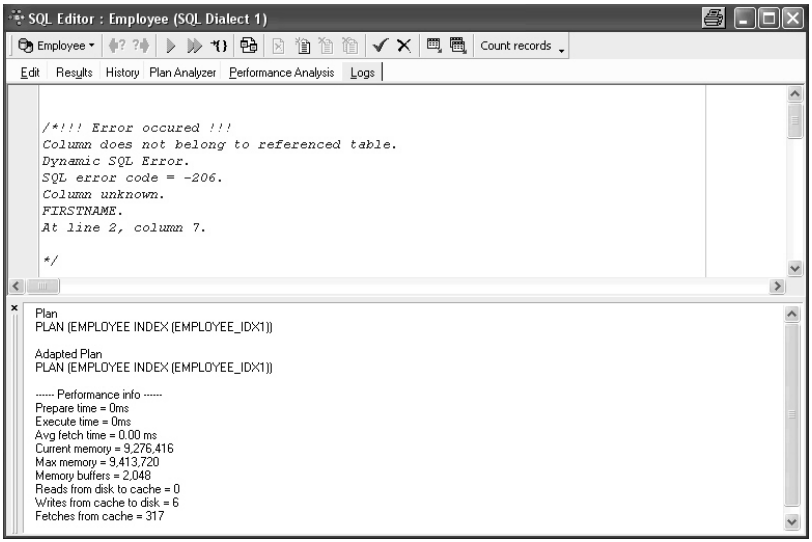
The Copy Analysis to Clipboard button copies all information included in the Additional page, including both the grid contents (= Enhanced Info) and the statistics listed in the left-hand panel (= query time, memory and operations).



The Copy Analysis to Clipboard button can be found in the bottom left corner of the 6. Additional dialog in the Performance Analysis. Should this not be visible, it is probably because the windows in IBExpert are set to Cascading. This can be easily solved by clicking the SQL/Procedure Editor dialog window to full-size (right-hand blue icon in the dialog title bar).

8.1.9 (6) Logs

The Log page can be found in the SQL Editor and displays a list of qualified error messages etc. It shows what Firebird/InterBase did and when in each respective SQL window.



8.1.10 Optimizing an SQL statement

If a lot of non-indexed reads (the red ones) appear in the Performance Analysis, it is often helpful to create some indices, reopen the query and check if it has been of help.

Analyze the reads, writes and fetches! Reads and writes are typically 0 when InterBase/Firebird can operate in the cache. Fetches are the internal operations in InterBase/Firebird, so when one query is slower than the other, it may not be visible directly in the graphical view, for example when InterBase/Firebird creates external temporary sort files.

Use the Plan Analyzer to analyze how the optimizer uses tables and indices to obtain the result set. If the word `SORT` is displayed, you should check whether improvements to the query or the indices are possible.

8.1.11 Special features

The IBExpert SQL Editor has two special features that allow you to:

- Create a table from query results and populate it with data.
- Move data between two registered databases.

Creating a table from query results

As everyone knows it is possible to insert data into any table by executing the `INSERT` statement:

```
INSERT INTO TARGET_TABLE
   SELECT FIELD_1, FIELD_2 FROM SOURCE_TABLE
   WHERE SOMETHING_FIELD <> 5
```

However this will only work if the table TARGET_TABLE already exists in the database.

IBExpert enables execution of this kind of statement even if the TARGET_TABLE does not exist in the database. First IBExpert notifies the user that TARGET_TABLE doesn't exist in the database and offers to create this table using query structure. If confirmed, IBExpert creates the TARGET_TABLE and then populates it with data from SELECT.

A small example illustrates how this works, based on a SOURCE_TABLE with the following structure:

```
CREATE TABLE SOURCE_TABLE (
   ID INTEGER,
   SOME_TEXT VARCHAR(50),
   SOME_PRICE NUMERIC(15,4),
   SOME_DATE DATE);
```

When the following statement is executed:

```
INSERT INTO TARGET_TABLE
   SELECT * FROM SOURCE_TABLE
```

and there is no TARGET_TABLE in the database, IBExpert will create TARGET_TABLE as:


```
CREATE TABLE TARGET_TABLE (
   ID INTEGER,
   SOME_TEXT VARCHAR(50),
   SOME_PRICE NUMERIC(15,4),
   SOME_DATE DATE);
```

and after that inserts into this table records retrieved with the SELECT part.

Of course, it is possible to write different INSERT statements. For example:

```
INSERT INTO [TARGET_DATABASE].TARGET_TABLE
   SELECT ID, SOME_DATE FROM TEST_TABLE
```

In this case IBExpert will create table TARGET_TABLE as

```
CREATE TABLE TARGET_TABLE (
   ID INTEGER,
   SOME_DATE DATE);
```

Moving data between databases

IBExpert allows you to move data from one database to another by executing special statement in SQL Editor.

Syntax:

```
INSERT INTO <database_alias>.<table_name>
   [(<columns_list>)]
```

```
<select_statement>
```

Argument	*Description*
database_alias	Alias of a registered database. This must be enclosed in square brackets. This argument is case-insensitive so aliases "My alias" and "MY ALIAS" are equivalent.
table_name	Name of the table to be populated with data..
columns_list	List of columns in target table. This argument is not obligatory.
select_statement	Any SELECT statement.

Examples

The following statement moves data from SOURCE_TABLE of the current database into TARGET_TABLE of the database with the alias "My test DB":

```
INSERT INTO [My test DB].TARGET_TABLE
   SELECT * FROM SOURCE_TABLE
```

If the table TARGET_TABLE doesn't exist in the target database, IBExpert will create it after your confirmation with the structure of the SOURCE_TABLE.

8.2 New SQL Editor

An additional SQL Editor can be opened using Tools / New SQL Editor, the respective icon in the Tools toolbar, or [Shift + F12].

The use of multiple SQL Editor windows does not affect the list of previous SQLs found on the History page, as this list is database dependent and not window dependent.

8.3 Query Builder

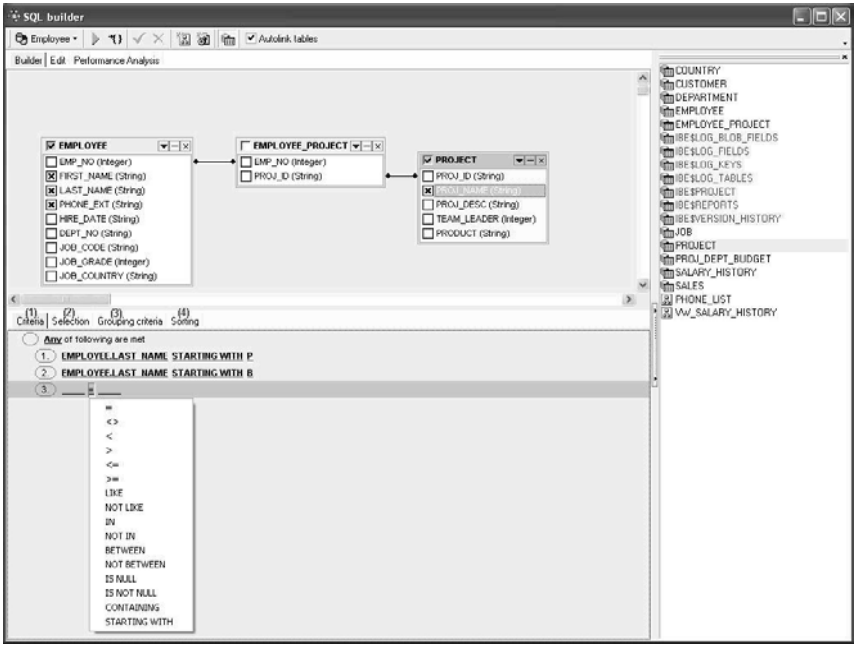
For those not yet competent in SQL, the Visual Query Builder is there to make life easier! It allows you to create and edit queries with multiple tables without previous knowledge of SQL, as well as prepare and execute queries, and view the results. This feature is unfortunately not included in the Personal Edition.

The IBExpert Query Builder is started using the menu item Tools / Query Builder. It can also be started directly from the SQL Editor using [Ctrl + Shift + Alt + B] or the

icon.

A query can be built by simply moving the database objects (e.g. by dragging the desired table) from the right panel over to the left editing area. Objects may also be dragged and dropped from the DB Explorer and SQL Assistant into the code editor window. Since version 2004.2.26.1 this has been greatly improved. When an object node(s) is dragged from the DB Explorer or SQL Assistant, IBExpert will offer various

versions of text to be inserted into the code editor. It is also now possible to customize the highlighting of variables. Use Options / Editor Options / Colors to choose color and font style for variables.



The required fields can be selected using the mouse. By clicking on the circle to the left of the table name, all fields are automatically highlighted. Tables can be linked, e.g. by key relationships, joins etc., using the mouse (click on the desired field in the first table and drag it across to the desired field in the second table). This creates a `JOIN`.

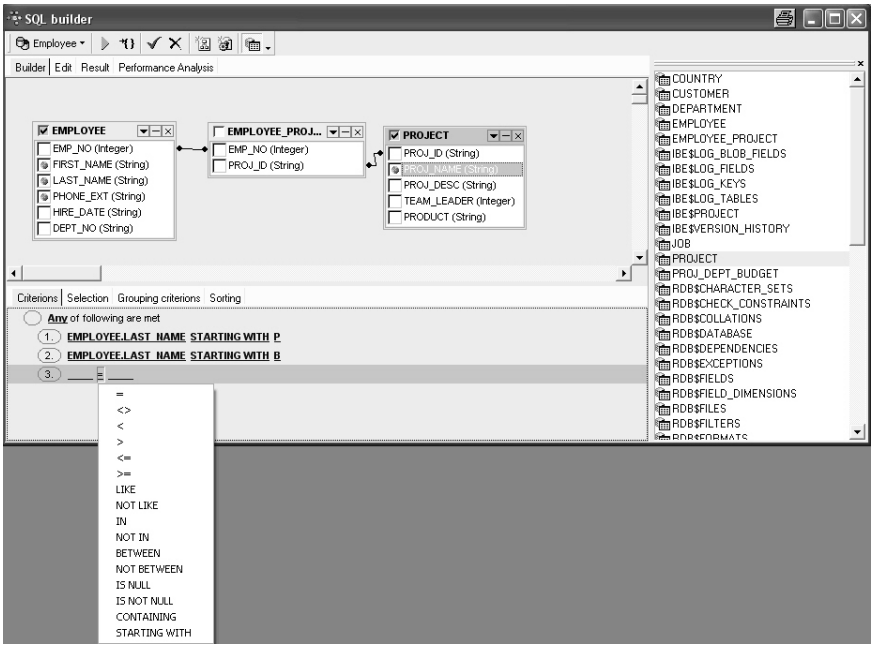
By double-clicking on the lines connecting two tables the option *Link Properties*, appears, and the developer can specify from which table all of the information should be fetched (see `JOIN` for more information about joins).

Alternatively, a small context-sensitive menu appears when right-clicking on a line, offering not only the above mentioned option, but also the option to insert or delete point or to delete the link.

Check every field which is important for the result set and use [F9] or the respective icon to execute and view the results. For information regarding the Results page, please refer to SQL Editor / Results.

Conditions can be specified in the lower part of the Query Builder dialog using the options listed under the following tabs:

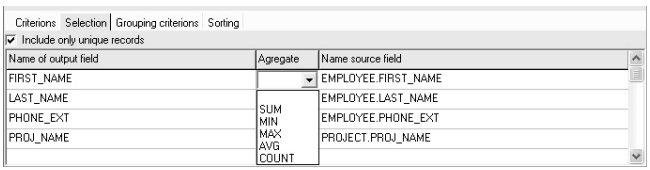
(1) Criteria



A simple condition string contains three fields: an argument, a condition and a second argument - if required for the condition. By clicking on the word *ALL* of *All of following are met*, it is possible to change this condition to *ALL, ANY, NONE*, or *NOT ALL*. By clicking on the ring to the left of *All of following are met*, it is possible to add a condition. Using [Shift + Enter] or right-clicking, fields can be selected from the specified tables. Alternatively a value can be manually entered. By clicking on the '=' sign a list of available conditions appears.

If you wish to view the SQL statement at any time, simply switch to the Edit page.

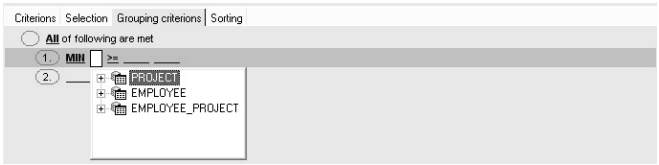
(2) Selection



An aggregate (`SUM`, `MIN`, `MAX`, `AVG` and `COUNT`) can be specified for individual fields if wished. For example, if a minimum or maximum order value needs to be determined; or the number of unpaid invoices. By double-clicking on a field in the builder area, the field source is automatically inserted. An output field name may be specified by double-clicking (or using the [Enter] key) on the first input field. The *Aggregate* pull-down list can be viewed by double-clicking or using the [Enter] key and downward arrow key, and an option selected.

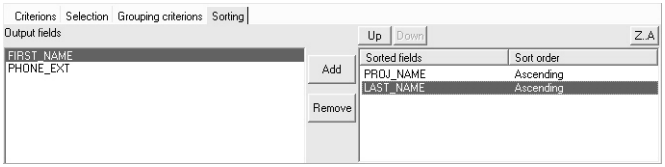
The *Include only unique records* checkbox eliminates duplicate records when checked.

(3) Grouping criteria



Again ALL, ANY, NONE, or NOT ALL of the specified conditions can be met. Here combined criteria can be determined, i.e. aggregate and comparative selection criteria.

(4) Sorting



Here the results can be sorted in ascending or descending order by one or more fields in order of priority. Simply move the field(s) to be used as the sorting criteria from the list on the left to the right-hand window, by selecting and clicking the Add button or using drag 'n' drop. Use the A.Z -Z.A button to specify ascending or descending order, and use the *Up* and *Down* buttons (when sorting by multiple fields) to specify sorting priority (i.e. which field should be sorted first).

When the query preparation is complete, it can be prepared [Ctrl + F9] and analyzed, and/or executed [F9] before finally committing.

In addition to the main Builder window, there is also an Edit page, displaying the query, resulting from the drag 'n' drop and condition specification in the main builder window, as SQL text. This is, in effect, the same as the SQL Editor's main Edit window. It can be edited directly, if wished, and all changes are displayed on the other Query Builder pages.

A **Results page** appears following query execution, displaying the returned data resulting from the query. A filter panel can also be blended into the dialog to aid data navigation and allow further filtering. For more information, please refer to SQL Editor / Edit and Filter Panel.

The **Plan Analyzer** is displayed following query execution and shows how Firebird/InterBase approaches a query, e.g. with `SORTS`, `JOINS` etc, which tables and indices are used. The information is shown in the lower panel in a tree structure with statistics.

The **Performance Analysis** displays information showing much effort was required by InterBase/Firebird to carry out an executed query or procedure. For more information please refer to SQL Editor / Performance Analysis.

Visual Query Builder is ideal for the beginner, although somewhat limited for more advanced work; complex queries should be performed in the SQL Editor.

8.4 Data Analysis

The IBExpert Tools menu item, Data Analysis, is new to IBExpert version 2004.10.30.1.

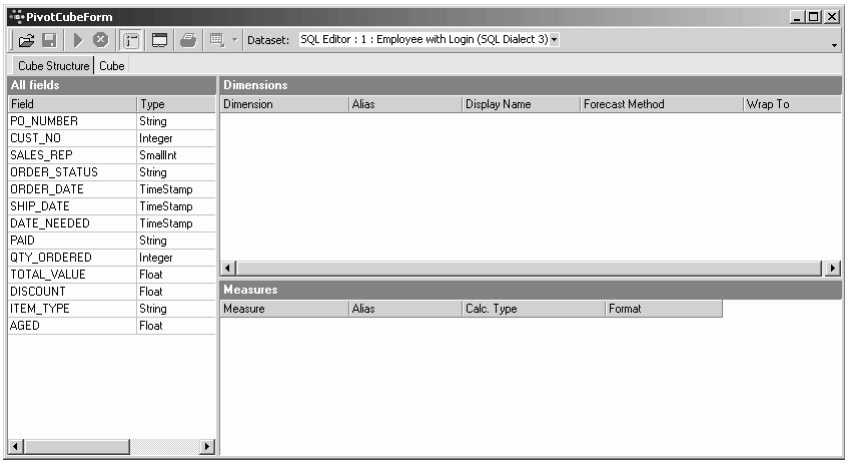
It is an ideal OLAP and data warehouse component, for analyzing data in the database quickly and easily. This sophisticated module can be used to build cubes, manage dimensions and measures, the technology being based on the building of multidimensional data sets - so-called OLAP cubes. It includes a powerful filtering system, enabling not only dimensions but also measures to be filtered.

The PivotCubeForm can be opened using the IBExpert Tools menu, or started directly from the SQL Editor / Results page, the Table Editor / Data page or the View Editor / Data page, using the Data Analysis icon:

We will illustrate the functionalities and options available in the Pivot Cube, using the following simple SELECT command, executed in the SQL Editor:

```
SELECT * FROM SALES;
```

By clicking the Data Analysis icon on the SQL Editor / Results page, the PivotCubeForm is opened:



The PivotCubeForm has its own toolbar (please refer to Data Analysis toolbar for further information), and contains two pages: Cube Structure and Cube.

Cube Structure:

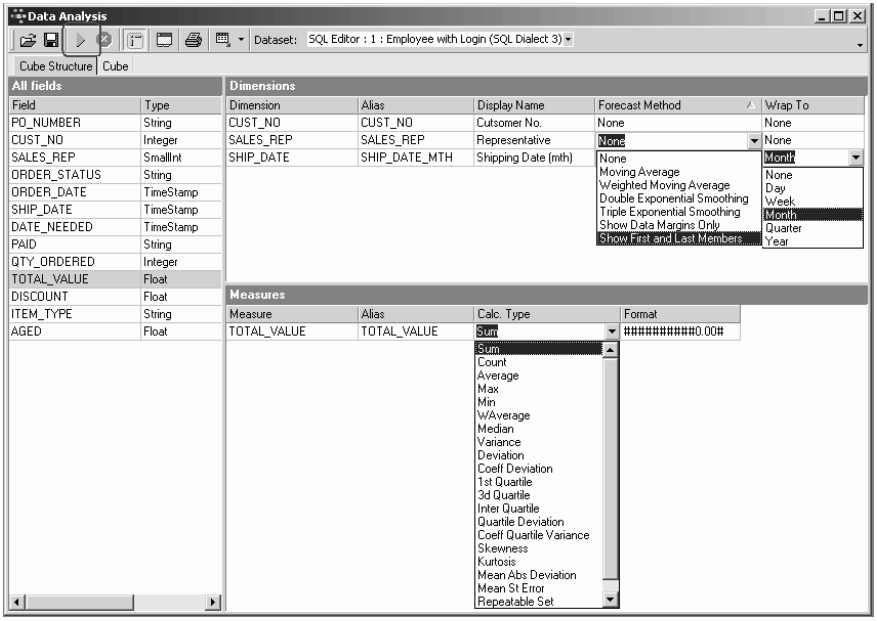
The first page has three main areas:

- **All Fields** - This automatically displays all data set fields displayed on the SQL Editor's Results page.
- **Dimensions** - what is to be analyzed and displayed. The field order is at this stage irrelevant.
- **Measures** - which values are to be analyzed and displayed. IBExpert Data Analysis permits use of any data types as measures; the only restriction being that non-numeric data types can only use the `ctCount` aggregate type.

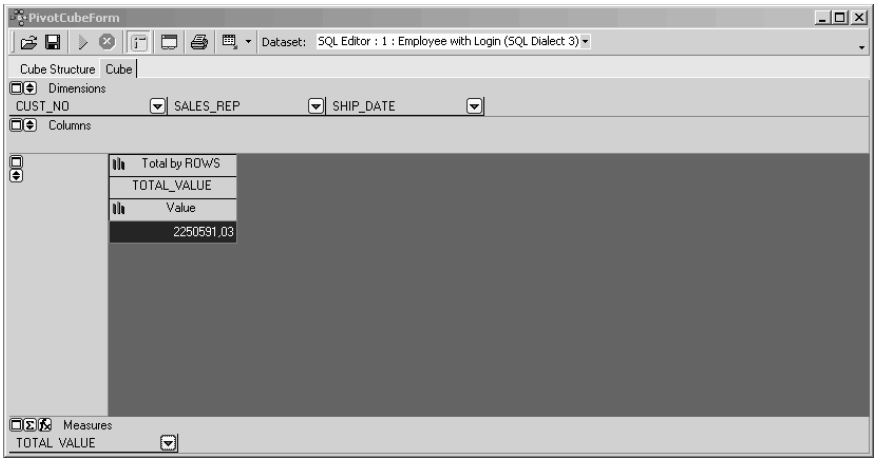
As with all IBExpert grids, columns can be sorted in ascending and descending order by simply clicking on the column headers.

Fields can be selected from the **All Fields** panel and dragged 'n' dropped into the **Dimensions** panel. For example, `CUST_NO`, `SALES_REP` and `SHIP_DATE`, the shipping date also being grouped by month. The *Alias* names and *Display Names* can be manually altered as wished, and the *Forecast Method* and *Wrap To* periods can be selected from the pull-down lists. (Simply click on the field where a selection is to be made, and click the black downward arrow on the right of the field to open the list of available options.)

The `TOTAL_VALUE` field can be dragged 'n' dropped from the **All Fields** panel into the **Measures** area. Again select *Calculation Type* from the options offered in the pull-down list; the numeric Format can be manually altered if desired:



And then the cube can be generated using the Build Cube icon or [F9] (see illustration above) and displayed on the Cube Page:



Cube:

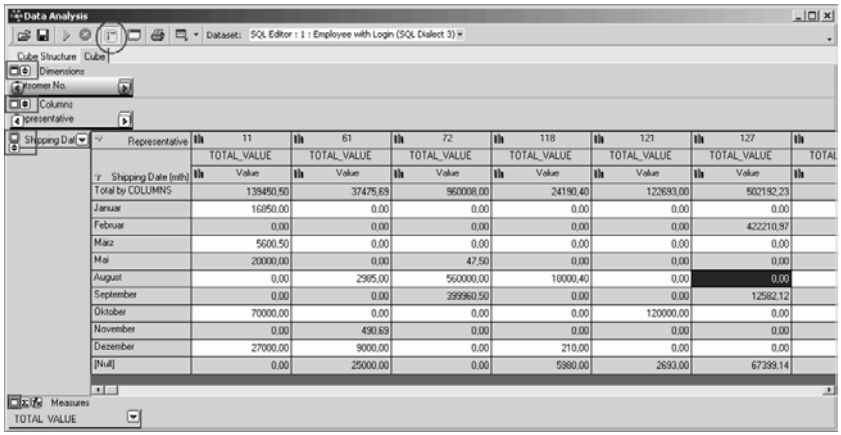
The second page in the PivotCube Form displays the cube itself in the third of four areas, so-called toolbars:

- Dimensions
- Columns
- Main display area
- Measures - the order of the items here determines how the data is displayed in the pivot grid.

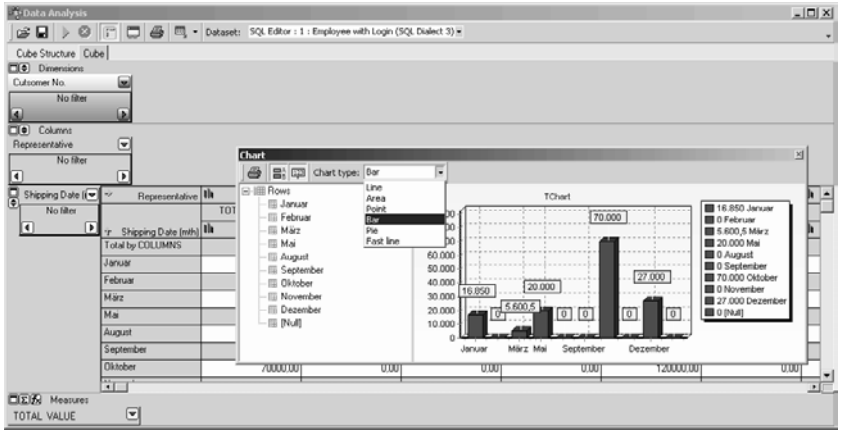
These areas can all be opened or closed, by clicking on the small square buttons in the upper left-hand corner of each area (see rectangular marked symbols in the illustration below). The arrow buttons can be used to adjust the size of the expanded areas, and display/hide the filter, which allows values to be searched and viewed for individual data sets.

The toggle toolbars on/off icon (see circled icon below) can be used to remove these areas completely leaving just the main blue display area, or blending them in again.

It is now possible to generate a summary, for example, which customer or which sales representative has generated which sales revenue. Or even which representative (column) has generated which revenue in which month:



The data can be displayed graphically with a simple mouse click. Simply click on the desired graphics icon to the left of the Measures (here: *Representative* or *Shipping Date (mth)*):



The Graphics window has its own mini toolbar, with the following options:

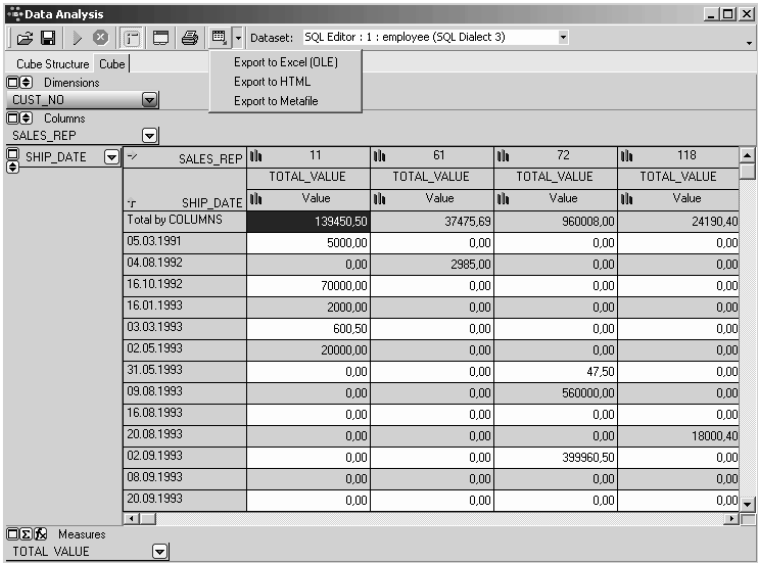
allowing the graph type to be altered, the legend and notes to be blended in or out, and enabling the graph to be printed.

There are numerous options to add functional values and formulae. Please refer to:

- Cube Manager,
- Calculated Measures Manager,

for further information.

The data and analyses generated, can be saved as *.CUB files, or exported to Excel (OLE), HTML or metafile. Simply click the small black arrow directly to the right of the Export icon, and select from the list:

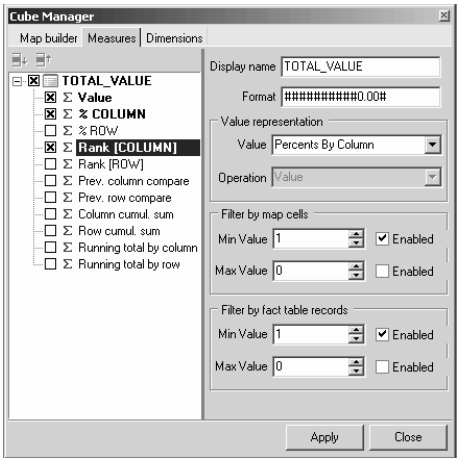


They can even be quickly and easily printed - simply click the printer icon (or [Ctrl + P]), to go to the Print Preview, where the page layout and appearance may be modified before finally printing.

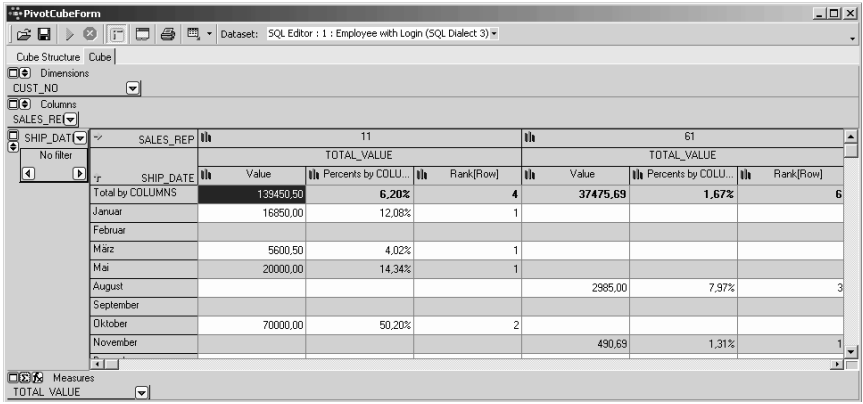
In fact, IBExpert's Data Analysis offers innumerable possibilities to define reports quickly and easily, or to simply collate the data material. And in order to pass these analyses on to others, a free runtime version is currently being planned, enabling the *.cub files (cube files) to be viewed without IBExpert having to be installed, so that other users can view those dimensions stored in the file, and can distribute them across rows or columns. It will also be possible to export the information to Excel etc. with this runtime version or print out the analysis.

8.4.1 Data Analysis Cube Manager

The Cube Manager can be opened using the PivotCube Form icon, or by clicking the Sum button in the bottom left hand corner of the Measures toolbar on the Cube page. This can be used to include certain alternative additional values. For example, alter the view to percentage column values



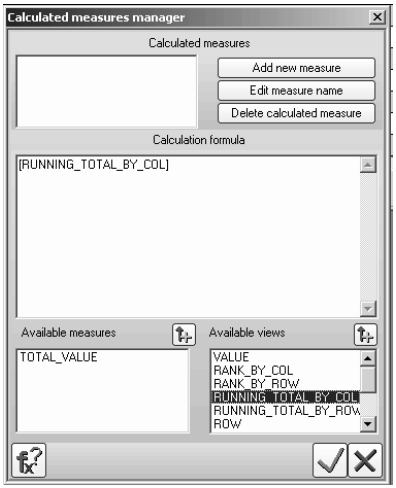
Click the Apply icon to view the results:



Depending on what you wish to see, it is possible to specify an ascending or descending order by simply clicking on the column headers.

8.4.2 Data Analysis Calculated Measures Manager

It is possible to integrate certain function values by clicking on the Function button in the bottom left hand corner of the Measures toolbar on the Cube page, to open the Calculated Measures Manager.



You can add new measures and edit or delete existing measures.

A new measure name can be added by clicking the Add New Measure button and inserting a name. A template automatically appears in the *Calculation Formula* input area. This can be completed manually, the *Available Measures* (bottom left-hand list) and *Available Views* (bottom right-hand list) can be inserted simply by double-clicking on the measure name, or clicking the [upward arrow +] button to the right of the Available Measures or Available Views headings.

When you are satisfied with your specifications, simply click the

button. You will now see both the original evaluation and the new calculated measure name displayed in the status bar. By clicking the black arrow to the right of these names, the Cube Manager is automatically opened, displaying the specifications made for the selected measure.

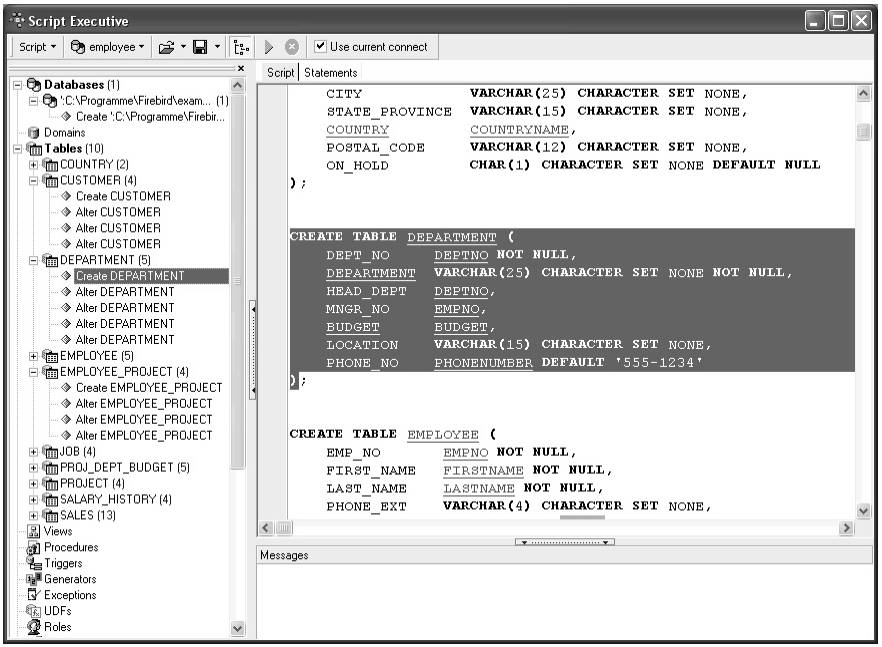
Simply re-click the Function button to reopen the Calculated Measures Manager, to make additional alterations, insertions or deletions as required.

8.5 Script Executive

The Script Executive can be used to view, edit and execute SQL scripts. It can be started from the IBExpert Tools menu, using the respective icon in the Tools toolbar or using [Ctrl + F12]. It is used for SQLs covering several rows. The Script Executive can both read and execute scripts.

Although InterBase/Firebird can also process such procedure definitions in the SQL Editor, it is recommended to use the Script Executive for more complex work, as it can do much more than the SQL Editor.

The main advantage of the Script Executive is that it displays all DDL and DML scripts of a connected database.



The Script Explorer (the left-hand panel) displays all database objects used in the current script in a tree structure. It even allows you to find a script part rapidly by clicking on the object in the tree. The Script Explorer can be blended in and out using the respective icon on the Script Executive toolbar. SQL scripts can be loaded from and saved to file if wished.

Objects may be dragged and dropped from the DB Explorer and SQL Assistant into the code editor window. And since version 2004.2.26.1 this has been greatly improved. When an object node(s) is dragged from the DB Explorer or SQL Assistant, IBExpert will offer various versions of text to be inserted into the code editor. It is also now possible to customize the highlighting of variables. Use Options / Editor Options / Colors to choose color and font style for variables.

Complete scripts can be transferred from the SQL Editor or extracted directly from the Extract Metadata Editor into the Script Executive using the relevant menu items (please refer directly to these subjects for further details).

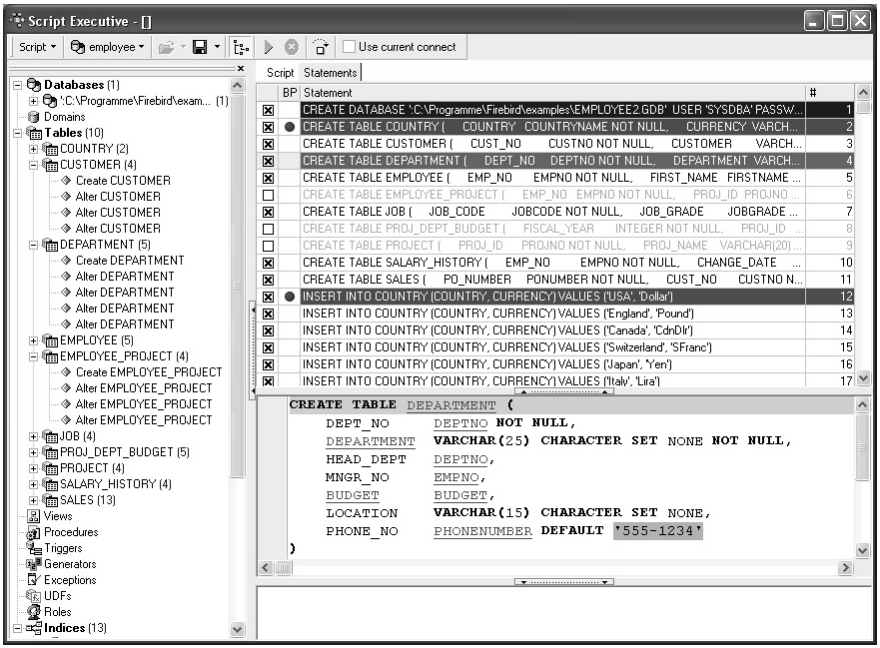
The Script Type may be selected from the Script Executive toolbar pull-down list (options include InterBase/Firebird or MySQL).

The Script page includes other features, such as code completion (please refer to Code Insight for details) - familiar from the SQL Editor. The SQL Editor menu can be called by right-clicking in the script area. Following statement execution, the Script page displays any errors highlighted in red. Using the

icon, the script can be executed step by step.

Any errors appearing in the lower Messages box may be saved to file if wished, using the right-click menu item Save Messages Log ...

The Statements page displays a list of individual statements in grid form:



These statements may be removed from the script simply by unchecking the left-hand boxes. One, several or all statements may be checked or unchecked using the right-click menu. Breakpoints can be specified or removed simply by clicking (or using the space bar) to the left of the selected statement in the BP column.

IBExpert version 2004.04.01.1 includes added support for the `EXECUTE BLOCK` statement (Firebird 2).

The following features were introduced in IBExpert version 2005.03.12.1:

- Executing of `INSERT/UPDATE/EXECUTE PROCEDURE` statements `WITHOUT` parameters is up to 10 times faster now.
- Added support for the following Firebird 2 features:
- `CREATE SEQUENCE`
- `DROP SEQUENCE`
- `ALTER SEQUENCE`

- Extended syntax of OUTPUT command. Please refer to OUTPUT for futher information and examples.

8.5.1 Executing multiple scripts from a single script

Simply use the following syntax:

```
connect 'server:c:\my_db.gdb' ...;

input 'c:\my_scripts\f2.sql';
input 'c:\my_scripts\f1.sql';
input 'c:\my_scripts\f3.sql';
```

8.5.2 Create multiple CSV files from a script

The following is an example illustrating the creation of multiple csv files from a script:

```
shell del C:\list.dat nowait;    --deleting the old file
shell del C:\*.csv nowait;    --deleting the old csv files

connect 'localhost:C:\employee.fdb' user 'SYSDBA' password 'masterke';  -
--connect to employee example database
output 'C:\list.dat';    --record the following result as a simple text
file, based on each unique employee, we create a new output ...;select
... ;output; line in the dat file

SELECT distinct
'OUTPUT C:\'||EMPLOYEE.last_name||'.csv delimiter '';'';'||
'SELECT distinct EMPLOYEE.last_name, customer.customer,customer.phone_no
'||
'FROM SALES INNER JOIN CUSTOMER ON (SALES.CUST_NO = CUSTOMER.CUST_NO) '||
'INNER JOIN EMPLOYEE ON (SALES.SALES_REP = EMPLOYEE.EMP_NO) where EM-
PLOYEE.last_name='''||EMPLOYEE.last_name||''';'||
'OUTPUT;'
FROM SALES INNER JOIN CUSTOMER ON (SALES.CUST_NO = CUSTOMER.CUST_NO) IN-
NER JOIN EMPLOYEE ON

(SALES.SALES_REP = EMPLOYEE.EMP_NO);

output;      --close the dat file
input 'C:\list.dat';  --execute them
```

The dat file is created automatically.

The outer query gets one record for each employee, in the inner select, all phone numbers for the employees' customers are selected.

8.5.3 Script Language Extensions

Script language extensions are unique to IBExpert, and offer the developer a number of additional language options. These include, among others, conditional directives, DESCRIBE database objects, as well as SET, SHELL, INSERTEX, OUTPUT and RECONNECT.

Conditional Directives

Conditional directives control conditional execution of parts of the script. Four types of conditional directives are supported:

- $IFEXISTS,
- $IFNOTEXISTS (or $IFNEXISTS),
- $ELSE,
- $ENDIF.

$IFEXISTS

This tests the existence of the specified database object or data and executes the following block of the script if the object or data do exist in the database.

Syntax:

1.
```
{$IFEXISTS DOMAIN|TABLE|VIEW|TRIGGER|PROCEDURE|
           EXCEPTION|GENERATOR|UDF|ROLE object_name}
```

2.
```
{$IFEXISTS select_statement}
```

Example:

The following script drops the exception `InvalidUserID` if it exists in the database:

```
{$IFEXISTS EXCEPTION "InvalidUserID"}

   DROP EXCEPTION "InvalidUserID";

{$ENDIF}
```

The next script alters a procedure:

```
{$IFEXISTS SELECT RDB$PROCEDURE_NAME
           FROM RDB$PROCEDURES
           WHERE RDB$PROCEDURE_NAME = 'GETDBVER'}

   ALTER PROCEDURE GETDBVER
   RETURNS (
       VER INTEGER)
   AS
   begin
     ver = 2;
     suspend;
   end;

{$ENDIF}
```

$IFNOTEXISTS ($IFNEXISTS)

This tests the existence of the specified database object or data and executes the following block of the script if the object or data does not exist in the database.

Syntax:

```
1. {$IFNOTEXISTS DOMAIN|TABLE|VIEW|TRIGGER|PROCEDURE|
                  EXCEPTION|GENERATOR|UDF|ROLE object_name}

2. {$IFNOTEXISTS select_statement}
```

Example:

The following script creates a table CUSTOMERS if there is no such table in the database:

```
{$IFNOTEXISTS TABLE CUSTOMERS}

   CREATE TABLE CUSTOMERS (
     ID          INTEGER NOT NULL PRIMARY KEY,
     FIRST_NAME  VARCHAR(30),
     MIDDLE_NAME VARCHAR(30),
     LAST_NAME   VARCHAR(30));

{$ENDIF}
```

The next script creates an exception:

```
{$IFNOTEXISTS SELECT RDB$EXCEPTION_NAME
              FROM RDB$EXCEPTIONS
              WHERE RDB$EXCEPTION_NAME = 'InvalidUserID'}

   CREATE EXCEPTION "InvalidUserID" 'Invalid User Identifier!';

{$ENDIF}
```

$ELSE

Switches between executing and ignoring the script part are delimited by the previous {$IFEXISTS} or {$IFNOTEXISTS} and the next {$ENDIF}.

Syntax:

```
{$ELSE}
```

Example:

The following script tests the existence of domain DOM_BOOL in the database. If domain DOM_BOOL cannot be found in the database it will be created. If domain DOM_BOOL already exists in the database it will be altered.

```
{$IFEXISTS DOMAIN DOM_BOOL}

  ALTER DOMAIN DOM_BOOL
  ADD CHECK (VALUE IN (0,1));

{$ELSE}

  CREATE DOMAIN DOM_BOOL AS SMALLINT
  DEFAULT 0 CHECK (VALUE IN (0,1));

{$ENDIF}
```

$ENDIF

Ends the conditional execution initiated by the last {$IFEXISTS} or {$IFNOTEXISTS} directive.

Syntax:

```
{$ENDIF}
```

Example:

The following script creates a generator:

```
{$IFNOTEXISTS GENERATOR "GenUserID"}

    CREATE GENERATOR "GenUserID";

{$ENDIF}
```

Conditional Directives – the complete example

This example illustrates the use of conditional directives for upgrading databases. Let's assume there is an initial version of your database (version 1):

```
CREATE TABLE FIRST_TABLE (
   ID    INTEGER NOT NULL,
   DATA  VARCHAR(100));

CREATE PROCEDURE GETDBVER
RETURNS (
  VER INTEGER)
AS
begin
  ver = 1;
  suspend;
end;
```

The next script will upgrade a database of any version < 4 to version 4.

```
/***** Upgrade to version 2 *****/
{$IfNotExists select ver from GetDBVer where ver > 1}

ALTER TABLE FIRST_TABLE
ADD CONSTRAINT PK_FIRST_TABLE
PRIMARY KEY (ID);

ALTER PROCEDURE GETDBVER
RETURNS (
  VER INTEGER)
AS
begin
  ver = 2;
  suspend;
end;

{$endif}


/***** Upgrade to version 3 *****/
{$IfNotExists select ver from GetDBVer where ver > 2}

CREATE GENERATOR GEN_FIRST_TABLE_ID;

CREATE TRIGGER FIRST_TABLE_BI0 FOR FIRST_TABLE
ACTIVE BEFORE INSERT POSITION 0
AS
begin
  new.id = gen_id(gen_first_table_id, 1);
end;

ALTER PROCEDURE GETDBVER
RETURNS (
    VER INTEGER)
AS
begin
  ver = 3;
  suspend;
end;

{$endif}


/***** Upgrade to version 4 *****/
{$IfNotExists select ver from GetDBVer where ver > 3}

CREATE EXCEPTION DELETION_NOT_ALLOWED 'You cannot delete records!';

CREATE TRIGGER FIRST_TABLE_BD0 FOR FIRST_TABLE
ACTIVE BEFORE DELETE POSITION 0
AS
begin
```

```
  exception deletion_not_allowed;
end;

ALTER PROCEDURE GETDBVER
RETURNS (
    VER INTEGER)
AS
begin
  ver = 4;
  suspend;
end;

{$endif}
```

DESCRIBE DOMAIN

This changes a domain description.

Syntax:

```
DESCRIBE DOMAIN domain_name 'description';
```

Argument	*Description*
domain_name	Name of an existing domain
'description'	Quoted string containing a domain description

DESCRIBE DOMAIN changes the description of an existing domain domain_name. When the IBExpert Script Executive executes this statement it modifies the value of the RDB$DESCRIPTION column in RDB$FIELDS connected with the specified domain name.

Actually the following statement is executed:

```
UPDATE RDB$FIELDS
SET RDB$DESCRIPTION = :DESC
WHERE RDB$FIELD_NAME = 'domain_name'
```

where DESC parameter is filled with the description.

Example:

```
DESCRIBE DOMAIN DOM_BOOL
'Boolean value:
 0 - FALSE
 1 - TRUE';
```

DESCRIBE EXCEPTION

This changes an exception's description.

Syntax:

```
DESCRIBE EXCEPTION exception_name 'description';
```

Argument	*Description*
exception_name	Name of an existing exception
'description'	Quoted string containing a new description of specified exception

Description:

DESCRIBE EXCEPTION changes the description of an existing exception exception_name. When the IBExpert Script Executive executes this statement it modifies the value of the RDB$DESCRIPTION column in RDB$EXCEPTIONS connected with the specified exception. Actually the following statement is executed:

```
UPDATE RDB$EXCEPTIONS
SET RDB$DESCRIPTION = :DESC
WHERE RDB$EXCEPTION_NAME = 'exception_name'
```


where DESC parameter is filled with the description.

Example:

```
DESCRIBE EXCEPTION MISSING_USER
'There is no such user!';
```

DESCRIBE FIELD

This changes a column description.

Syntax:

```
DESCRIBE FIELD column_name TABLE table_name 'description';
```

Argument	*Description*
column_name	Name of an existing column of table table_name
table	Name of an existing table
'description'	Quoted string containing a column description

Description:

DESCRIBE FIELD changes the description of an existing column column_name of table table_name. When the IBExpert Script Executive executes this statement it modifies the value of the RDB$DESCRIPTION column in RDB$RELATION_FIELDS connected with the specified column and table names. Actually the following statement is executed:

```
UPDATE RDB$RELATION_FIELDS
 SET RDB$DESCRIPTION = :DESC
 WHERE (RDB$RELATION_NAME = 'table_name') AND
       (RDB$FIELD_NAME = 'column_name')
```

where the DESC parameter is filled with the description.

Example:

```
DESCRIBE FIELD FULL_USER_NAME TABLE USERS
'Full user name.
 Computed, concatenation of FIRST_NAME, MIDDLE_NAME and LAST_NAME';
```

DESCRIBE FUNCTION

This changes an UDF description.

Syntax:

```
DESCRIBE FUNCTION function_name 'description';
```

Argument	*Description*
function_name	Name of an existing user-defined function
'description'	Quoted string containing an UDF description

DESCRIBE FUNCTION changes the description of an existing user-defined function function_name. When the IBExpert Script Executive executes this statement it modifies the value of the RDB$DESCRIPTION column in RDB$FUNCTIONS connected with the specified function. Actually the following statement is executed:

```
UPDATE RDB$FUNCTIONS
 SET RDB$DESCRIPTION = :DESC
 WHERE RDB$FUNCTION_NAME = 'function_name'
```

where the DESC parameter is filled with the description.

Example:

```
DESCRIBE FUNCTION COMPARE_BLOBS
'Compares two blob values and returns 1
if both values are equal. In other case returns 0';
```

DESCRIBE PARAMETER

This changes a procedure parameter description.

Syntax:

```
DESCRIBE PARAMETER parameter_name PROCEDURE procedure_name 'description';
```

Argument	*Description*
parameter_name	Name of an existing parameter of stored procedure
procedure_name	Name of an existing stored procedure
'description'	Quoted string containing a parameter description

Description:

DESCRIBE PARAMETER changes the description of an existing parameter parameter_name of a specified stored procedure procedure_name. When the IBExpert Script Executive executes this statement it modifies the value of the RDB$DESCRIPTION column in RDB$PROCEDURE_PARAMETERS connected with the specified parameter and procedure names. Actually the following statement is executed:

```
UPDATE RDB$PROCEDURE_PARAMETERS
 SET RDB$DESCRIPTION = :DESC
 WHERE (RDB$PROCEDURE_NAME = 'procedure_name') AND
       (RDB$PARAMETER_NAME = 'parameter_name')
```


where the DESC parameter is filled with the description.

Example:

```
DESCRIBE PARAMETER USER_ID PROCEDURE CALC_TRAFFIC
'User ID';
```

DESCRIBE PROCEDURE

This changes a stored procedure description.

Syntax:

```
DESCRIBE PROCEDURE procedure_name 'description';
```

Argument	*Description*
procedure_name	Name of an existing stored procedure
'description'	Quoted string containing a procedure description

Description:

DESCRIBE PROCEDURE changes the description of an existing stored procedure procedure_name. When the IBExpert Script Executive executes this statement it modifies the value of the RDB$DESCRIPTION column in RDB$PROCEDURES connected with the specified procedure. Actually the following statement is executed:

```
UPDATE RDB$PROCEDURES
SET RDB$DESCRIPTION = :DESC
WHERE RDB$PROCEDURE_NAME = 'procedure_name'
```

where the DESC parameter is filled with the description.

Example:

```
DESCRIBE PROCEDURE CALC_TRAFFIC
 'Calculates the summary traffic';
```

DESCRIBE TABLE

This changes a table description

Syntax:

```
DESCRIBE TABLE table_name 'description';
```

Argument	*Description*
table_name	Name of an existing table
'description'	Quoted string containing a table description

Description:

DESCRIBE TABLE changes the description of an existing table table_name. When the IBExpert Script Executive executes this statement it modifies the value of the RDB$DESCRIPTION column in RDB$RELATIONS connected with the specified table. Actually following statement is executed:

```
UPDATE RDB$RELATIONS
 SET RDB$DESCRIPTION = :DESC
 WHERE RDB$RELATION_NAME = 'table_name'
```

where the DESC parameter is filled with the description.

Example:

```
DESCRIBE TABLE CUSTOMERS
 'Customers of our excellent application';
```

DESCRIBE TRIGGER

This changes a trigger description

Syntax:

```
DESCRIBE TRIGGER trigger_name 'description';
```

Argument	*Description*
trigger_name	Name of an existing trigger
'description'	Quoted string containing a trigger description

Description:

DESCRIBE TRIGGER changes the description of an existing trigger trigger_name. When the IBExpert Script Executive executes this statement it modifies the value of the RDB$DESCRIPTION column of RDB$TRIGGERS connected with the specified table. Actually the following statement is executed:

```
UPDATE RDB$TRIGGERS
 SET RDB$DESCRIPTION = :DESC
 WHERE RDB$TRIGGER_NAME = 'trigger_name'
```

where the DESC parameter is filled with the description.

Example:

```
DESCRIBE TRIGGER USERS_BI
 'Generates an unique identifier';
```


DESCRIBE VIEW

This changes a view description

Syntax:

```
DESCRIBE VIEW view_name 'description';
```

Argument	*Description*
view_name	Name of an existing view
'description'	Quoted string containing a view description

Description:

DESCRIBE VIEW changes the description of an existing view view_name. When the IBExpert Script Executive executes this statement it modifies the value of the RDB$DESCRIPTION column of RDB$RELATIONS connected with the specified view. Actually the following statement is executed:

```
UPDATE RDB$RELATIONS
 SET RDB$DESCRIPTION = :DESC
 WHERE RDB$RELATION_NAME = 'view_name'
```

where the DESC parameter is filled with the description.

Example:

```
DESCRIBE VIEW ALL_USERS
'Just all users...:)';
```

INSERTEX (CSV file import)

This imports data from a CSV-file into a database table.

Syntax:

```
INSERTEX INTO table_name [(columns_list)]
   FROM CSV file_name
   [SKIP n]
   [DELIMITER delimiter_char]
```

Argument	*Description*
`table_name`	Name of a table into which to insert data
`columns_list`	List of columns into which to insert data
`file_name`	Name of CSV-file from which to import data
`SKIP n`	Allows the first n lines of CSV-file to be skipped while importing data
`DELIMITER delimiter_char`	Allows a delimiter to be specified, which will be used for parsing data values. If this argument isn't specified IBExpert will use a colon as a delimiter.

Description:

`INSERTEX` imports data from a CSV-file into a database table. Values within the CSV-file must be separated with a colon `CHAR` or any other char. In the latter case it is necessary to specify a delimiter `CHAR` using the `DELIMITER` argument. It is also possible to specify non-print characters as a delimiter. For example, if values are separated with tab char (ASCII value $09) it may be specified as `DELIMITER #9` or `DELIMITER $9`.

If a table `table_name` is missing in the database, it will be created automatically. In this case the number of columns in the newly created table will be equal to the number of values in the first line of the CSV-file. Columns will be named F_1, F_2 etc. The data type of each column is `VARCHAR(255)`.

If the `columns_list` isn't specified IBExpert will insert data from the very first column. Otherwise data will only be inserted into specified columns.

It is possible to skip the first several lines of the CSV-file using the `SKIP` argument. This may be useful if the first line contains column captions or is empty.

Since IBExpert version 2005.02.12.1 it is possible to use the `INSERTEX` command in the SQL Editor.

Examples:

Let's consider the use of INSERTEX in the following examples. Assume there is a CSV-file with the following data, delimited with a colon:

```
C:\Mydata.csv
 ======================================================
 ID:FIRST_NAME:LAST_NAME:SEX
 1:John:Doe:M
 2:Bill:Gates:M
 3:Sharon:Stone:F
 4:Stephen:King:M
 ======================================================
```

The following INSERTEX statement creates a table PEOPLE (if it doesn't already exist) and fills it with data from C:\Mydata.csv:

```
INSERTEX INTO PEOPLE FROM CSV 'C:\Mydata.csv' DELIMITER ':';
```

The structure and contents of PEOPLE after the data import are shown below:

F_1 (VAR-CHAR(255))	*F_2 (VAR-CHAR(255))*	*F_3 (VAR-CHAR(255))*	*F_4 (VAR-CHAR(255))*
ID	FIRST_NAME	LAST_NAME	SEX
1	John	Doe	M
2	Bill	Gates	M
3	Sharon	Stone	F
4	Stephen	King	M

The following INSERTEX statement is almost identical to the one above, but here the first line of the CSV-file has been skipped:

```
INSERTEX INTO PEOPLE FROM CSV 'C:\Mydata.csv' DELIMITER ':' SKIP 1;
```

The structure and content of the PEOPLE table after import is shown below:

F_1 (VAR-CHAR(255))	*F_2 (VAR-CHAR(255))*	*F_3 (VAR-CHAR(255))*	*F_4 (VAR-CHAR(255))*
1	John	Doe	M
2	Bill	Gates	M
3	Sharon	Stone	F
4	Stephen	King	M

In the next example the PEOPLE table is created first, and then subsequently populated with the data from C:\Mydata.csv:

```
CREATE TABLE PEOPLE (
    ID         INTEGER NOT NULL,
```

```
    FIRST_NAME VARCHAR(30),
    LAST_NAME  VARCHAR(30),
    SEX        CHAR(1));

INSERTEX INTO PEOPLE FROM CSV 'C:\Mydata.csv' DELIMITER ':' SKIP 1;
```

Below the structure and content of the PEOPLE table after import:

ID (INTEGER)	*FIRST_NAME (VARCHAR(30))*	*LAST_NAME (VARCHAR(30))*	*SEX (CHAR(1))*
1	John	Doe	M
2	Bill	Gates	M
3	Sharon	Stone	F
4	Stephen	King	M

In the next example only three columns (ID, FIRST_NAME and LAST_NAME) are affected:

```
CREATE TABLE PEOPLE (
    ID         INTEGER NOT NULL,
    FIRST_NAME VARCHAR(30),
    LAST_NAME  VARCHAR(30),
    SEX        CHAR(1));

INSERTEX INTO PEOPLE (ID, FIRST_NAME, LAST_NAME)
    FROM CSV 'C:\Mydata.csv'
    DELIMITER ':' SKIP 1;
```

The structure and content of the PEOPLE table after import can be seen below:

ID (INTEGER)	*FIRST_NAME (VARCHAR(30))*	*LAST_NAME (VARCHAR(30))*	*SEX (CHAR(1))*
1	John	Doe	NULL
2	Bill	Gates	NULL
3	Sharon	Stone	NULL
4	Stephen	King	NULL

OUTPUT

This redirects the output of SELECT statements to a named file.

Syntax:

```
OUTPUT [filename [DELIMITER delim_char]
                 [QUOTECHAR 'quote_char']
                 [TIMEFORMAT 'time_format']
                 [DATEFORMAT 'date_format']
```

```
[DECIMALSEPARATOR 'dec_sep']
[NULLS]
[FIELDNAMES]
[ASINSERT [INTO table]]]
```

Argument	*Description*
`filename`	Name of the file in which to save output.
`DELIMITER delim_char`	Determines a delimiter character which is used for separating field values. If the delimiter is not specified, or the empty string is specified as a delimiter, outswapping of the data will be carried out in the format with the fixed positions of fields. It is also possible to specify a delimiter character as a decimal or hexadecimal value of the character code. For example, to set the tab character (ASCII value $09) as a delimiter, simply specify `DELIMITER #9` or `DELIMITER $9`.
`QUOTECHAR 'quote_char'`	Defines the character which will be used for quoting string values. If this argument is not specified or an empty string is specified, string values will not be quoted.
`TIMEFORMAT 'time_format'`	Defines the string which will be used for formatting the values of time fields and the time slice of datetime values. If the argument is not defined, time values will be unloaded in the native InterBase format (for example, 17:15:45).
`DATEFORMAT 'date_format'`	Defines the string which will be used for formatting values of date fields and the date part of datetime values. If the argument is not defined, date values will be unloaded in the native InterBase format (for example, 17-FEB-2001).
`DECIMALSEPARATOR 'dec_sep'`	Defines the decimal separator which is used when outswapping the data. If this argument is not defined, the system decimal separator is used.
`NULLS`	Defines how `NULL` values will be output. If the argument is not specified, `NULL`s are output as an empty string. Otherwise `NULL`s will be unloaded as the string "<null>".
`FIELDNAMES`	If this argument is specified, the first line in the resulting file will be a line with names of `SELECT` columns.
`ASINSERT`	This argument allows data to be unloaded as a set of `INSERT` operators, i.e. to get a usual SQL script.
`INTO table`	It is used together with `ASINSERT` for redefining the name of the table in `INSERT` operators. If the argument is not given, the name of the first table in the record set will be used.

Description:

The `OUTPUT` operator is intended for redirecting the output of `SELECT` statements in an external file. With the help of the given operator it is possible to export the data easily into a file with separators or with a fixed column position.

OUTPUT without parameters closes the file which was opened with the previous OUTPUT command, and resets all export customizations to default.

If ASINSERT is not specified, blob fields are ignored when outswapping the data. Using ASINSERT even blob values are exported, i.e. an additional file with the extension "lob" is created, in which all blob fields are stored.

While outputting into SQL script (ASINSERT is specified) DELIMITER, QUOTECHAR, NULLS and FIELDNAMES arguments are ignored.

Examples:

The following script creates a MyData.txt file in the current directory and outputs the data of the SELECT into it, with a fixed column position format. If MyData.txt file already exists in the current directory, the data will be appended to it.

```
OUTPUT MyData.txt;

SELECT * FROM MY_TABLE;
OUTPUT;
```

In the next example the data will be exported in the comma-separated values (CSV) format:

```
OUTPUT 'C:\MyData\MyData.csv' DELIMITER ';'

                              FIELDNAMES
                              QUOTECHAR '"'
                              DECIMALSEPARATOR '.';
SELECT * FROM MY_TABLE;
OUTPUT;
```

In the following script the data will be exported into SQL script as a set of INSERT operators:

```
OUTPUT 'C:\MyScripts\Data.sql' ASINSERT INTO "MyTable";

SELECT * FROM MY_TABLE;
OUTPUT;
```

The next example illustrates usage of the OUTPUT statement together with SHELL.

```
/* First create a folder C:\MyData*/
SHELL MKDIR C:\MyData;

/* Try to delete mydata.csv */
SHELL DEL C:\MyData\mydata.csv;

/* Redirect output of SELECTs into mydata.csv */
OUTPUT C:\MyData\mydata.csv DELIMITER ';'
                           DATEFORMAT 'MMMM-dd-yyyy'
                           TIMEFORMAT 'hh:nn:ss.zzz'
                           QUOTECHAR '"';

SELECT * FROM MY_TABLE;
```

```
/* Close C:\MyData\mydata.csv */
OUTPUT;

/* Try to open just created CSV-file with Windows Notepad */
SHELL notepad.exe C:\MyData\mydata.csv NOWAIT;

/* Try to open C:\MyData\mydata.csv with the application
   associated with CSV files */
SHELL C:\MyData\mydata.csv NOWAIT;
```

New in IBExpert version 2.5.0.61:

1. The NOFIELDNAMES option is obsolete now. This means that there will be no column captions in the output file by default. If you wish to include column captions use FIELDNAMES option.

2. Added possibility to customize delimiter char for INSERTEX command (DELIMITER option). If the DELIMITER option is missing a comma will be used as the delimiter char.

New in IBExpert version 2005.03.12:

Extended syntax of OUTPUT command:


```
1.
output 'E:\data.sql'
  as insert into mytable commit after 1000;
  select * from IBE$$TEST_DATA where F_INTEGER < 3000;
  output;

2.
output 'E:\data.sql'
  as reinsert into mytable
  commit after 2000;
  select * from IBE$$TEST_DATA where F_INTEGER < 3000;
  output;

3.
output 'E:\data.sql'
  as execute procedure myproc;
  select * from IBE$$TEST_DATA where F_INTEGER < 3000;
  output;
```

ASINSERT option is available for compatibility.

RECONNECT

RECONNECT closes the current connection and creates a new one with the same parameters (database, user name, password etc.).

Syntax:

```
RECONNECT;
```

REINSERT

IBExpert has introduced the new REINSERT statement. Directly following an INSERT it is possible to perform further INSERTs with new contents.

SET BLOBFILE

IBExpert uses an original mechanism to extract values of blob fields into a script. This allows you to store the entire database (metadata and data) into script files and execute these scripts with IBExpert. A small example illustrates the method used to extract blob values.

For example, your database has a table named COMMENTS:

```
CREATE TABLE COMMENTS (
   COMMENT_ID INTEGER NOT NULL PRIMARY KEY,
   COMMENT_TEXT BLOB SUBTYPE TEXT);
```

This table has three records:

COMMENT_ID	*COMMENT_TEXT*
1	First comment
2	NULL
3	Another comment

If the *Extract BLOBs* option is not checked, you will receive the following script:

```
CREATE TABLE COMMENTS (
   COMMENT_ID INTEGER NOT NULL PRIMARY KEY,
   COMMENT_TEXT BLOB SUBTYPE TEXT);

INSERT INTO COMMENTS (COMMENT_ID) VALUES (1);
INSERT INTO COMMENTS (COMMENT_ID) VALUES (2);
INSERT INTO COMMENTS (COMMENT_ID) VALUES (3);
```

... and, of course, you will lose your comments if you restore your database from this script.

But if the *Extract BLOBs* option is checked IBExpert will generate quite a different script:

```
SET BLOBFILE 'C:\MY_SCRIPTS\RESULT.LOB';

CREATE TABLE COMMENTS (
   COMMENT_ID INTEGER NOT NULL PRIMARY KEY,
   COMMENT_TEXT BLOB SUBTYPE TEXT);

INSERT INTO COMMENTS (COMMENT_ID, COMMENT_TEXT) VALUES (1,
h0000000_0000000D);
INSERT INTO COMMENTS (COMMENT_ID, COMMENT_TEXT) VALUES (2, NULL);
```

```
INSERT INTO COMMENTS (COMMENT_ID, COMMENT_TEXT) VALUES (3,
h000000D_0000000F);
```

Also IBExpert generates a special file with the extension LOB where blob values are stored. In the current example result.lob will be 28 bytes long and its contents will be First commentAnother comment.

SET BLOBFILE is a special extension of script language that allows IBExpert's Script Executive to execute scripts containing references to blob field values.

SET CLIENTLIB

This defines the client library to be used while executing a script.

Syntax:

```
SET CLIENTLIB file_name;
```

Argument	*Description*
file_name	Client library file name

Description:

SET CLIENTLIB defines client library which will be used while executing a script. The default client library is gds32.dll.

Example:

```
SET CLIENTLIB 'C:\Program Files\Firebird\Bin\fbclient.dll';
```

SET PARAMFILE

PARAM file is an ini-file with param values.

For example, if your script contains some parameterized INSERT/UPDATE/DELETE statements you can define parameter values in an external file (params file):

```
param1=12-FEB-2003
param2=John Doe
param3=35
...
```

When IBEScript finds a query with parameters it looks for the values of these parameters in the specified params file.

SHELL

This allows execution of an operating system command.

Syntax:

```
SHELL os_command [NOWAIT];
```

Argument	*Description*
os_command	An operating system command
NOWAIT	Optional argument. If specified, execution of a script will be continued right after creation of the process executing the command of operating system, not waiting its completion.

Description:

The SHELL operator tries to execute the command os_command. If NOWAIT is not specified, the further execution of a script stops before completion of the process created by SHELL operator. Otherwise script execution will be continued immediately after beginning the execution of the command os_command.

Examples:

The following script tries to create a folder MyFolder in the current directory:

```
SHELL mkdir MyFolder;
```

The following example shows the use of the SHELL command to start Notepad.exe and the loading of C:\MyTexts\Shedule.txt file in it. It is necessary to use NOWAIT here, otherwise it is not possible to execute the script further, and it will be impossible to resume work in IBExpert until the Notepad is closed.

```
SHELL "notepad.exe C:\MyTexts\Shedule.txt" NOWAIT;
```

The next example illustrates the use of the SHELL statement together with OUTPUT.

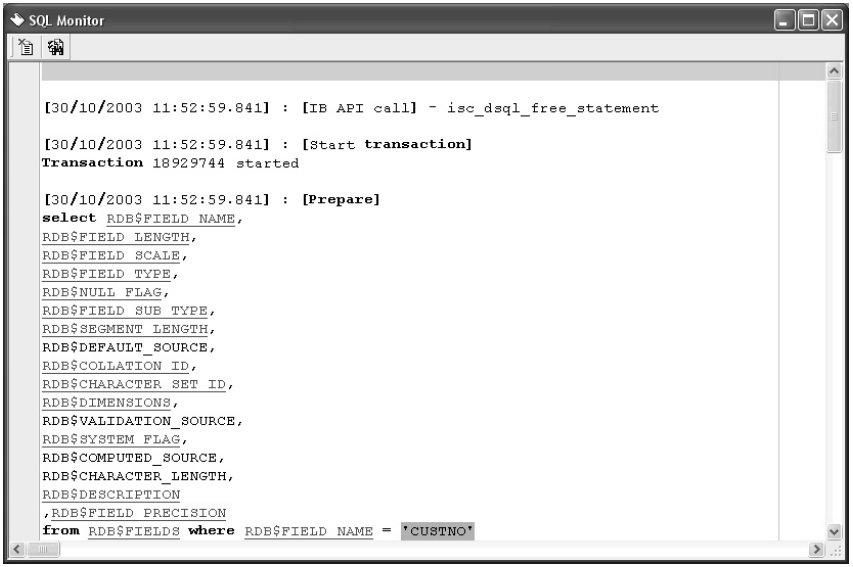
```
/* First create a folder C:\MyData*/
SHELL MKDIR C:\MyData;

/* Try to delete mydata.csv */
SHELL DEL C:\MyData\mydata.csv;

/* Redirect output of SELECTs into mydata.csv */
OUTPUT C:\MyData\mydata.csv DELIMITER ';'
                           DATEFORMAT 'MMMM-dd-yyyy'
                           TIMEFORMAT 'hh:nn:ss.zzz'
                           QUOTECHAR '"';

SELECT * FROM MY_TABLE;

/* Close C:\MyData\mydata.csv */
OUTPUT;

/* Try to open just created CSV-file with Windows Notepad */
SHELL notepad.exe C:\MyData\mydata.csv NOWAIT;
```

```
/* Try to open C:\MyData\mydata.csv with the application
   associated with CSV files */
SHELL C:\MyData\mydata.csv NOWAIT;
```

8.6 SQL Monitor

The SQL Monitor can be started in the IBExpert Tools menu, using the respective icon in the Tools toolbar or using the key combination [Ctrl + M].

The SQL Monitor can be used if a detailed protocol is required. Once opened, it logs everything performed in IBExpert, allowing the user to view all actions as SQL code.



It provides detailed background information, for those wishing to learn and analyze the way IBExpert works. It is also an ideal tool for analyzing certain problems or error messages that can otherwise not easily be solved.

The SQL Monitor always includes a timestamp, regardless of whether this option is checked in the Database Registration Info / Log Files or not.

The SQL code cannot be edited directly; it can however be copied to clipboard, saved to file or printed, using the right-click SQL Editor menu. Further operations, such as Incremental Search, are explained under SQL Editor Menu.

Please note that the SQL Monitor is not able to log all SQL calls to the database server; it only logs IBExpert calls.

Please refer to SQL Monitor Options for details of customization.

8.6.1 SQL Monitor Options

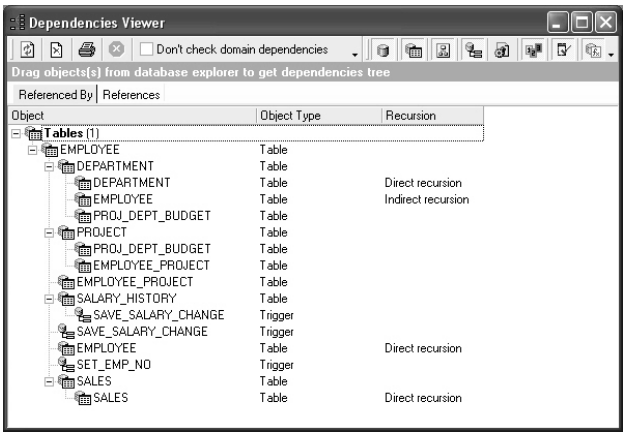
The Monitor Options icon:

- **Connect/Disconnect** - whether the database connection should also be protocolled.
- **Prepare / Execute / Fetch** - which phases of the SQL queries should be monitored.
- **Transactions** - whether each individual transaction should be monitored.
- **Services** - monitoring of the individual commands at API level
- **API calls** - direct InterBase/Firebird calls (ICE files). This option may really only be of interest to hardcore C programmers!

8.7 Dependencies Viewer

The IBExpert Dependencies Viewer is an ideal tool for ascertaining any dependencies upon an object or an object's dependency upon other objects - particularly important before deleting objects!

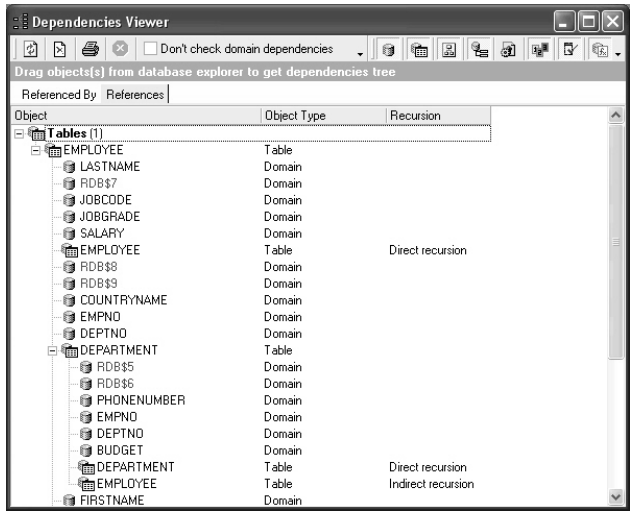
It can be found in the IBExpert Tools menu.



Database objects can be simply moved from the DB Explorer into the Viewer using drag 'n' drop.

The Reference By page displays which objects reference the selected object, i.e. the higher-ranking objects (in the above illustration EMPLOYEE) are referenced by the subordinate objects (in the above example: DEPARTMENT, PROJECT, EMPLOYEE_PROJECT, SALARY_HISTORY, EMPLOYEE (references itself = direct recursion), SET_EMP_NO and SALES).

The References page:



shows which objects are used by the selected object. In the above example, this includes, among others, the EMPLOYEE and DEPARTMENT tables.

It is possible to specify whether domains should be displayed or not, by simply checking the *Don't Show Domains* box in the toolbar. As it is possible for domains to reference other domains, and each table field is based either on a user-defined or system domain, this may slow work with the Dependencies Viewer if it is not checked.

Further object display criteria are offered by the icons in the toolbar (please refer to Dependencies Viewer toolbar for details).

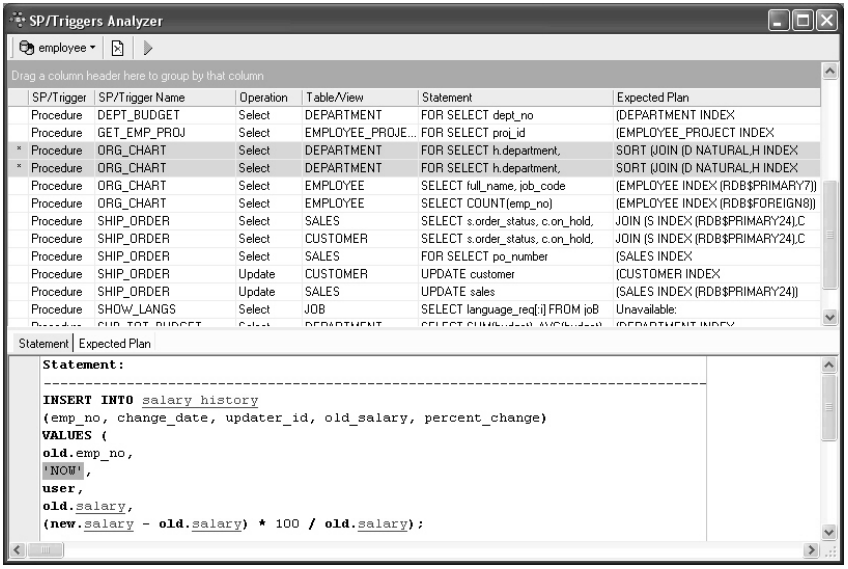
- **Direct recursion** indicates that an object references itself.
- **Indirect recursion** indicates that an object references itself indirectly via one or more other objects, for example EMPLOYEE references itself indirectly via DEPARTMENT (each employee belongs to a department; each department has a manager, who is an employee).

Double-clicking on any of the objects in the Viewer opens the respective object dialog.

8.8 SP/Triggers/Views Analyzer

The Stored Procedure/Trigger/Views Analyzer is new to IBExpert version 2.5.0.47 and can be found in the Tools menu. (This feature is unfortunately not included in the Personal Edition.)

It allows the user to view and analyze how the database performs the individual operations/statements in a stored procedure, trigger or view. For example, certain indices may not be used by the database server, as the statistics are too high; this can be solved simply by using the IBExpert Database menu item Recompute selectivity of all indices. Or when backing up an older InterBase version and restoring to a new InterBase/Firebird version, the procedures and triggers appear not to work, as it is often necessary to first recompute selectivity of all stored procedures and triggers (found in the IBExpert Database menu).



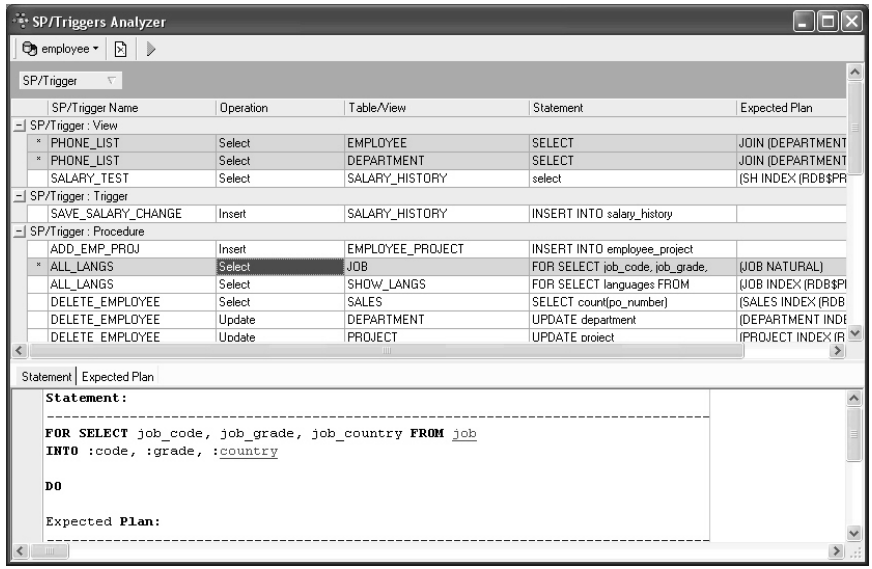
The database to be analyzed can be selected from the pull-down list of all connected databases (first toolbar item). By clicking on the Start Analyzing icon, it loads all stored procedures and triggers for the active database.

They are all automatically analyzed, i.e. each procedure/trigger is split up into the individual statements (the first SQL row is displayed in the Statement column; the full code is displayed in the lower Statement window). The indices used for each operation are displayed in the right-hand Expected Plan column; details may be viewed in the lower Expected Plan window. Those not using indices (i.e. NATURAL) are highlighted.

As with all IBExpert editors the contents can be sorted by clicking on the desired column header (e.g. sort according to Name, Table/View, statement etc.). By clicking on the left-hand column header, the red highlighted objects (i.e. those including a NATURAL plan) are grouped together.

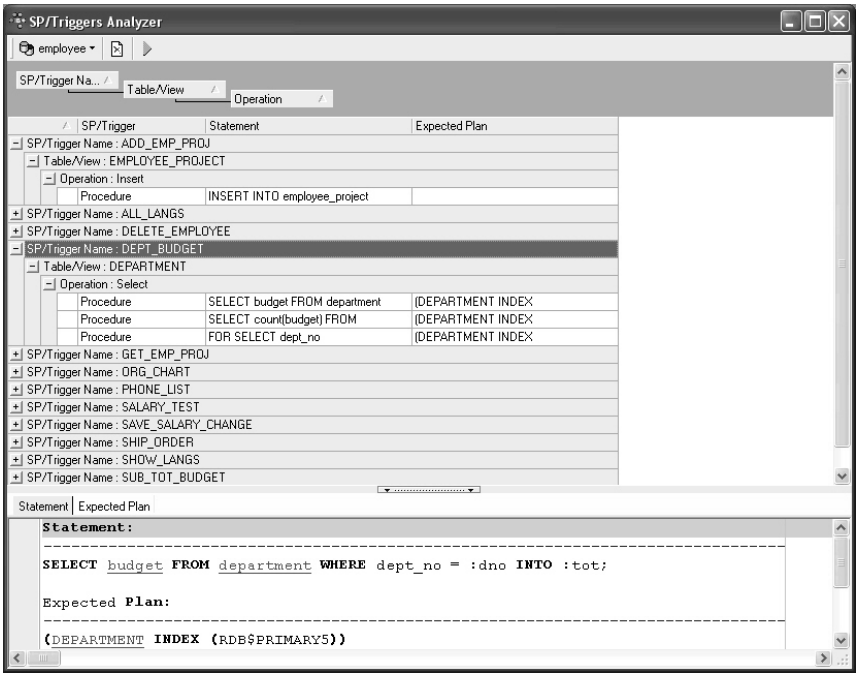
The Procedure, Trigger, Table or View editors can be quickly started by double-clicking on a selected field, allowing the user for example, to quickly and easily insert an index.

Column headers can also be dragged to the gray area below the toolbar, to group by the column selected:



The above illustration displays all stored procedures and triggers grouped by the procedure or trigger name. By clicking '+' or '-', or double-clicking on the list name, the individual operations can be easily blended in or out.

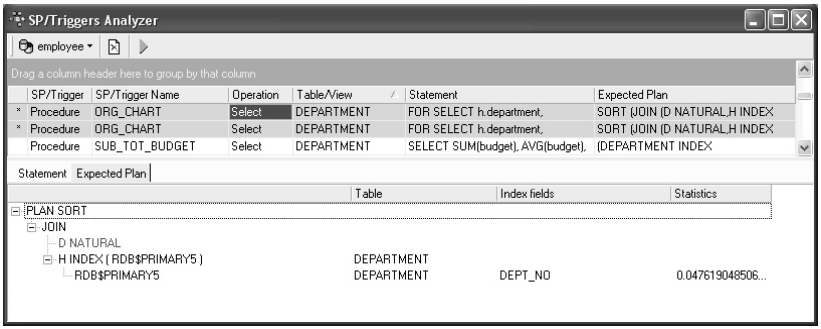
It is also possible to group by more than one criteria:



A filter function is soon to be incorporated, so that it is possible, for example, to display only those object using a NATURAL plan.

The lower window displays the SQL text for a selected operation on the Statement page, in the lower half of the window. The statements can easily be copied and inserted into a text editor or the IBExpert SQL Editor, using the context-sensitive right-click menu (please refer to SQL Editor Menu for further details).

The Expected Plan page displays the plan in a tree form:



In case it is of interest, the SP/Triggers/Views Analyzer was realized using the Developer Express component.

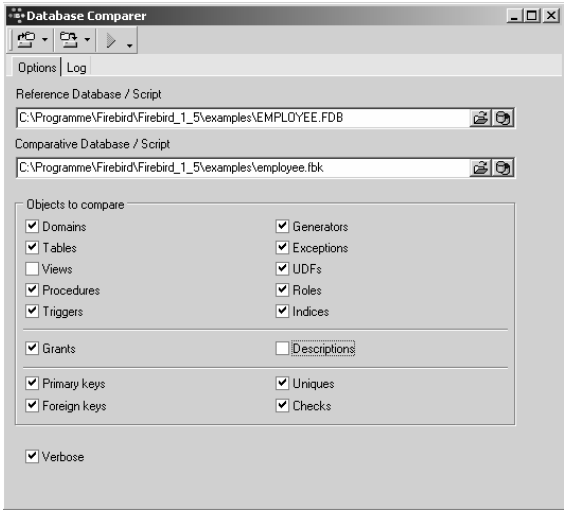
8.9 Database Comparer

The IBExpert Database Comparer can be found in the Tools menu. This tool is new to IBExpert version 2.5.0.47. This feature is unfortunately not included in the Personal Edition.

It allows developers to compare database versions or database SQL scripts. This is particularly useful for example, before installing an updated client application, which contains new tables, procedures, exceptions, etc. etc., as it is possible to compare the databases, and - by analyzing the resulting script, view both the changes to the software, as well as those data changes made by the client, erasing any irrelevant alterations, and applying those which are relevant, by executing the script.

On the Options page, first select the Master or Reference Database or SQL script, by clicking the icons to the right of the path/file input area. This is the reference database, to which the second database is to be compared. Then select the Comparative or Target Database, i.e. the database which needs to be assessed and altered in order to conform with the reference database.

Since IBExpert version 2004.04.01.1 scripts can also be selected and compared. It is also possible to store into or load from an external file (using the toolbar icons), and use this together with IBECompare (IBExpert command-line tool).



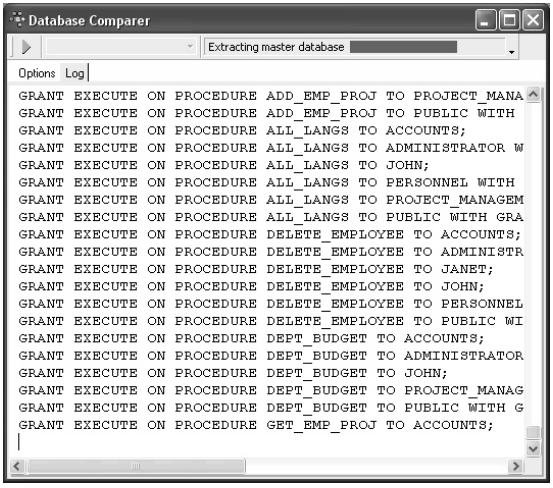
There are a number of options, which can be checked if they should be included in the comparison. These include:

- *Objects to compare*:
  domains
  tables
  views
  procedures (see stored procedure for further information)
  triggers

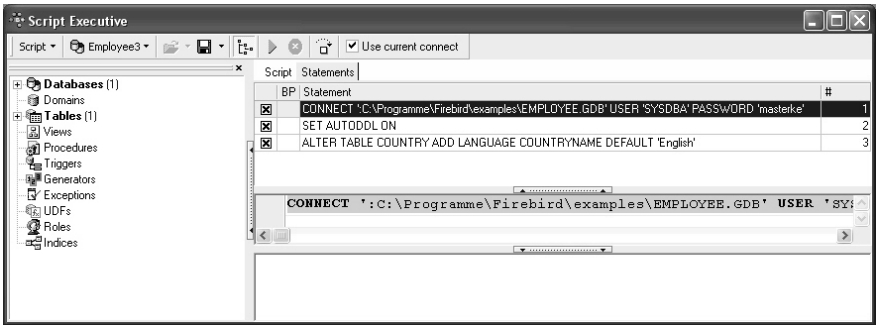
generators
exceptions
UDFs
roles
indices (see index for further information)

- grants (see Grant Manager for further information)
  descriptions (see Description page for further information)
- *Keys and constraints*:
  primary keys
  foreign keys
  uniques
  checks (see check constraint for further information)
- verbose: this displays each step that IBExpert performs and when, allowing a detailed comparison.

Click the Compare icon to start the comparison.



The Log page logs the comparison, which can be halted and restarted at any time by using the Stop and Compare icons. The results are automatically loaded in the Script Executive. Here it is easy to see which operations need to be performed, in order to make the comparative database identical to the reference database. Using the Statements page:



It is simple to unselect or select individual statements using point and click. Please refer to Script Executive for further details. By executing all SQL statements the comparative database becomes identical to the master database.

Please note that certain alterations may cause serious problems with your database, due to restrictions and limitations in Firebird/InterBase. For example, changing a data type from `CHAR` to `INT`.

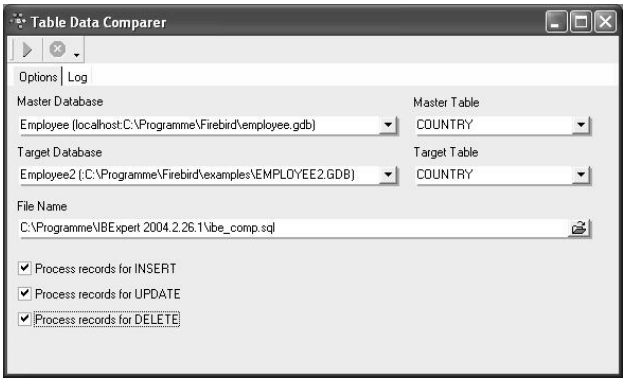
We at IBExpert are aiming to generate comments for all such items that cannot be modified. Please mail us (see below) if you incur problems, which are not yet reported by IBExpert.

Firebird also seems to have problems with certain dependencies. For example, when dropping a view with dependent procedures, the Firebird server removes records from `RDB$DEPENDENCIES` and doesn't recreate them when the view is recreated.

As the Database Comparer is a new feature in IBExpert, we would like to hear from anyone who may have discovered an example of a database comparison, which does not function one hundred per cent. Please mail details (with the database, if it is not too large!) to support@ibexpert.com.

8.10 Table Data Comparer

The Table Data Comparer is new to IBExpert version 2004.2.26.1 and can be found in the Tools menu. It allows you to compare data of two tables in different databases and obtain a script detailing all discrepancies which includes corresponding `INSERT`, `UPDATE` and `DELETE` statements. This feature is unfortunately not included in the Personal Edition.

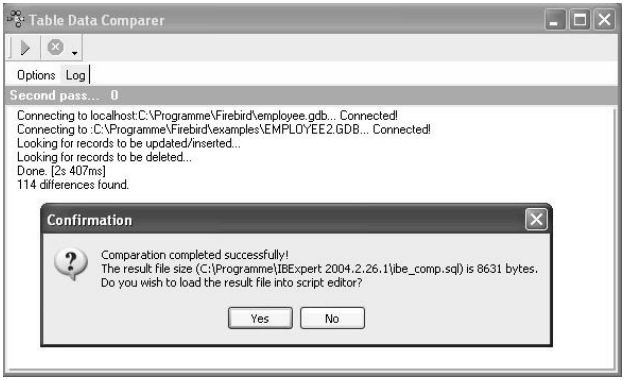


On the Options page, first specify the Master or Reference Database from the pull-down list of all registered databases, followed by the Master Table, which is to be used as the basis for the comparison. This is the reference table, to which the second table is to be compared. Then select the Target Database and Target Table, i.e. the database and table which needs to be assessed and altered in order to conform with the reference database and table. The databases and tables must already exist. The tables may have different names but they must have the same structure. Since IBExpert version 2004.8.5.1 an error is raised if there is no primary key defined for the reference table.

The default File Name for the resulting script may be altered if wished.

Finally check whether the records should be processed for INSERT, UPDATE or DELETE, before clicking the Compare button (green arrow) or [F9].

The resulting log:



displays whether the database connections were successful, records searched, time taken and the number of discrepancies found.

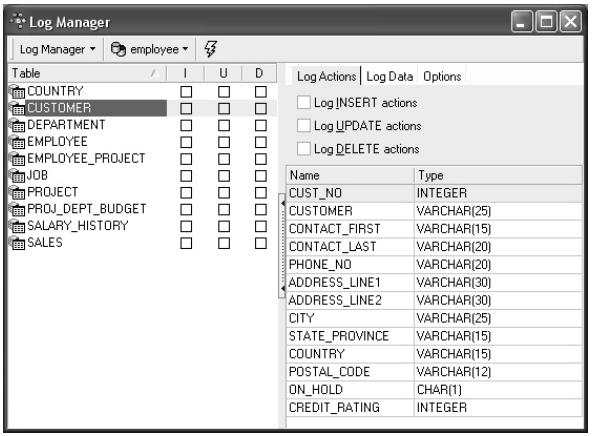
The resulting script file may then be loaded into the Script Executive if wished.

Since IBExpert version 2005.03.12 the speed of the Table Data Comparer has been considerable increased (up to five times faster!).

8.11 Log Manager

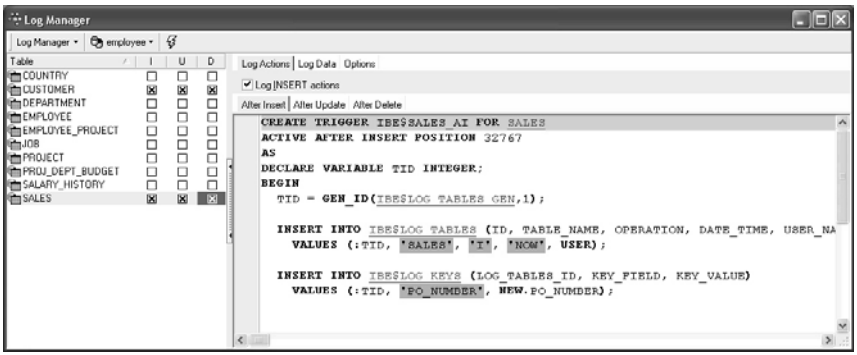
The IBExpert Log Manager can be found in the Tools menu. This tool is new to IBExpert version 2.5.0.47. This feature is unfortunately not included in the Personal Edition.

Select the database to be logged from the pull-down list of registered databases. When initially opened, the Log Actions page displays check options for logging INSERT, UPDATE and DELETE actions,



below which the selected table's fields and field types are displayed. The logging options, for example which INSERT, UPDATE and DELETE actions on which tables, can be checked individually or alternatively, the Log Manager pull-down menu can be used to either Prepare All Tables or to Unprepare All Tables. Although it should be taken into consideration, that when all actions on all tables are to be logged, this could slow the database performance somewhat.

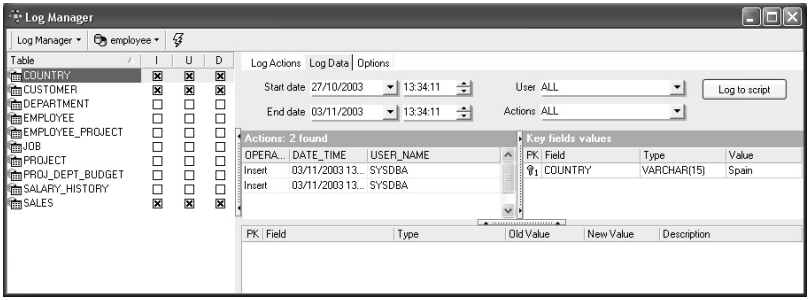
Once the actions have been selected, the Log Actions page displays the SQL code:



which can be copied to clipboard, if wished, using the right-click SQL Editor Menu.

New in version 2004.6.17 - templates have been added for data logging triggers. Please refer to Options / General Templates / Data Logging Triggers for more details.

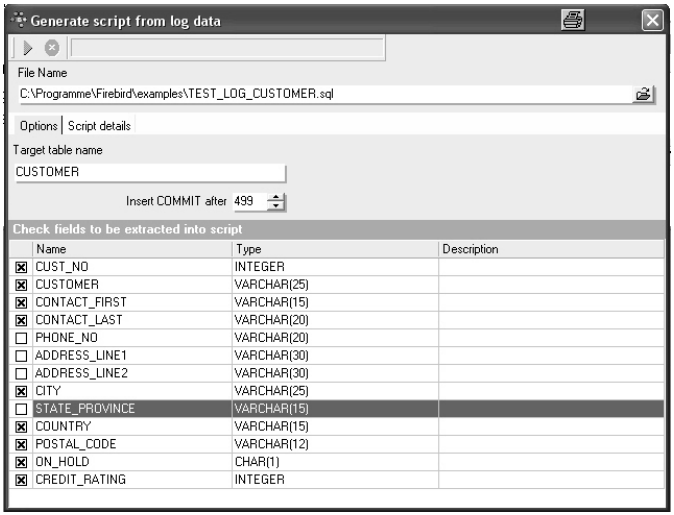
The Log Data page displays the new and old values:



In IBExpert version 2004.12.12.1 a new feature was added, allowing you to generate a log script for several tables simultaneously. Simply select the required tables using the [Ctrl + Shift] keys. And since IBExpert version 2005.02.12.1 64-bit IDs are now used when working with SQL Dialect 3 databases.

If a system error message appears when clicking on this page, stating that an IBExpert system table is missing, open any table from the DB Explorer and click on the Logging page in the Table Editor. You will then be automatically asked, whether IBExpert should generate certain system tables. After confirming and committing, you should have no further problems!

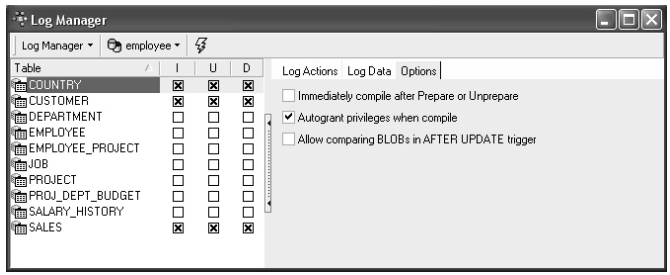
The following can be user-specified: Start Date, End Date (both with timestamp), individual or all users and individual or all actions. The specified log can also be logged to file if wished, by clicking on the Log to Script button, which produces a new dialog box:



where the Script File Name can be specified, and on the Options page, the Target Table, Commit Interval, and user specification of which fields are to be extracted into the script. The Script Details page allows the user to write his own Start of Script and End of Script.

This Log file can even be used as a sort of replication. This is because, as opposed to the logging specified in the Database Registration, which only logs all IBExpert actions, the Log Manager logs all actions and operations on the database itself, including those of all users.

Back to the Log Manager Editor, the Options page:



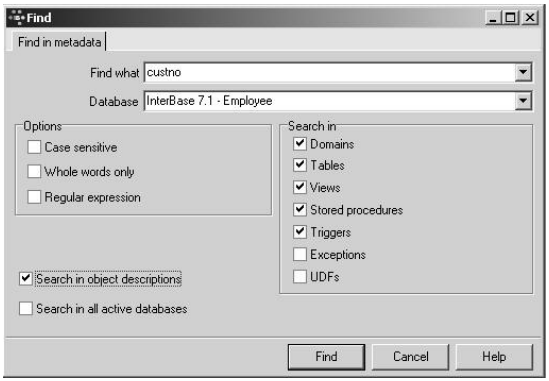
allows the user to specify the following options:

- Immediately compile after Prepare or Unprepare
- Autogrant privileges when compiling (generally this should be activated).
- Allow comparing BLOBS in AFTER UPDATE trigger (new to version 2004.1.22.1)

8.12 Search in Metadata

The Search in Metadata option can be found in the IBExpert Tools menu, using the respective icon in the Tools toolbar, or started using the key combination [Shift + Alt + F]. It is identical to the Edit menu's Find option - Find in Metadata page.

This option is useful for finding individual words/digits or word/digit strings in metadata (and since IBExpert version 2004.8.5 also in object descriptions). It even searches for and displays field names, as opposed to the DB Explorer Filter, which only searches for object names. The Find Metadata dialog offers a number of options:

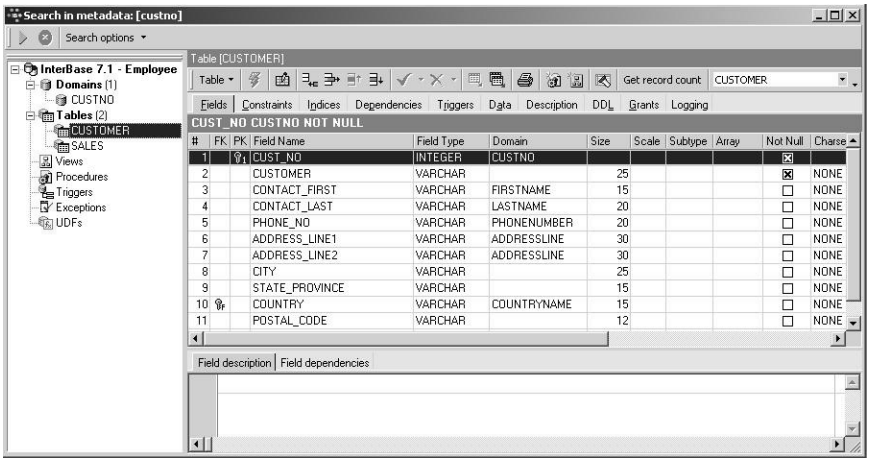


Here the user can specify what he is looking for; the pull-down list displays previous search criteria. A single active database may be selected from the second pull-down list; alternatively the Search in all Active Database option can be checked, in the bottom left-hand corner of the dialog.

Further Search options include:

- **Options**:
- *Case sensitive* - differentiates between upper and lower case
- *Whole words only* - as opposed to whole or parts of words
- *Regular Expression* - recognizes regular expressions in the search string.
- **Search in**: determines which object types should be searched - domains, tables, views, stored procedures, triggers, exceptions, UDFs.

After clicking on the Find button, a new Search dialog is opened:



The Search Options button in the toolbar can be used to restart the Find dialog, in order to specify new Search conditions. The arrow to the right of this produces a drop-down overview of the search criteria specified.

The results of the Metadata Search are displayed in the usual IBExpert tree form, sorted by database object type. By clicking on an object, the object editor is opened in the Search in Metadata dialog, and can be edited as wished. Alternatively, a double-click on the tree object opens the object editor.

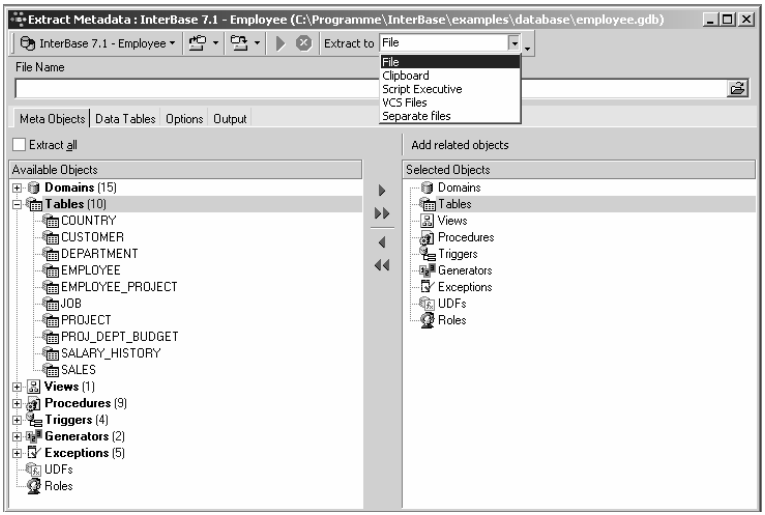
8.13 Extract Metadata

The Extract Metadata menu item can be found in the IBExpert Tools menu, or started using the respective icon in the Tools toolbar.

The Extract Metadata module can be used to generate a partial or full database metadata script, including table data, privileges and objects descriptions if wished. It allows the user to extract metadata to file or clipboard. It is even possible to extract blob data.

And since version 2004.1.22.1, it is also possible to extract date/timestamp/time values with ANSI-prefixes:

```
  INSERT INTO MY_TABLE (DATE_FIELD, TIME_FIELD, TIMESTAMP_FIELD)
  VALUES (date '01.01.2004', time '12:15:45',timestamp '01.01.2004
12:15:45');
```



Since IBExpert version 2004.04.01.1 it is possible to extract table data into separate files (`TABLE_1.sql`, `TABLE_2.sql`, `TABLE_3.sql` etc.). This version also includes support for default values of input parameters (Firebird 2).

Support for the InterBase 7.5 temporary tables feature was added in IBExpert version 2004.12.12.1.

First a database needs to be selected from the toolbar's pull-down list of all registered databases. The toolbar's Extract to options include:

- File
- Clipboard
- Script Executive (default)
- VCS Files (previously, before IBExpert version 2004.9.12.1, named "Separate Files")
- Separate Files (new to IBExpert version 2004.9.12.1)

The Separate Files mode extracts metadata (and data if specified) into a set of files: two files with metadata (`_ibe$start_.sql` and `_ibe$finish_.sql`), files containing table data (one or more files for each database table) and a `runme.sql` file, that consists of a number of `INPUT <file_name>` statements in the correct order.

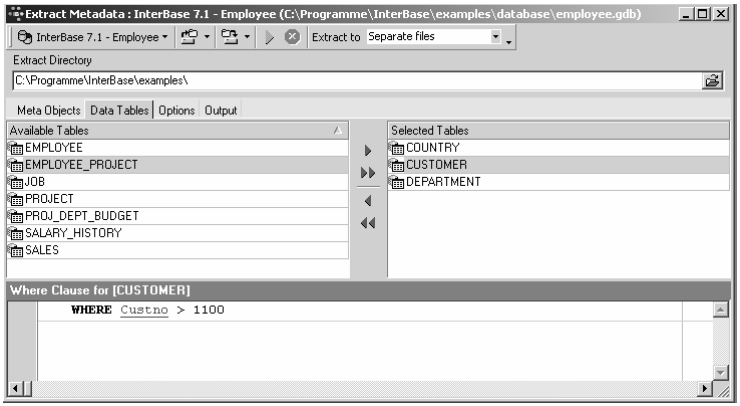
If either the *File, VCS Files* or *Separate Files* options are chosen, it is of course necessary to specify a file path and name (`*.sql` or Metadata Extract Configuration `*.mec`).

Meta Objects Page:

The first dialog page Meta Objects displays the Select Objects Tree (please refer to this subject for further information).

Data Tables Page:

The Data Tables page can be used to specify whether data should also be extracted. This allows both user-defined and system tables to be selected - either all or individually:

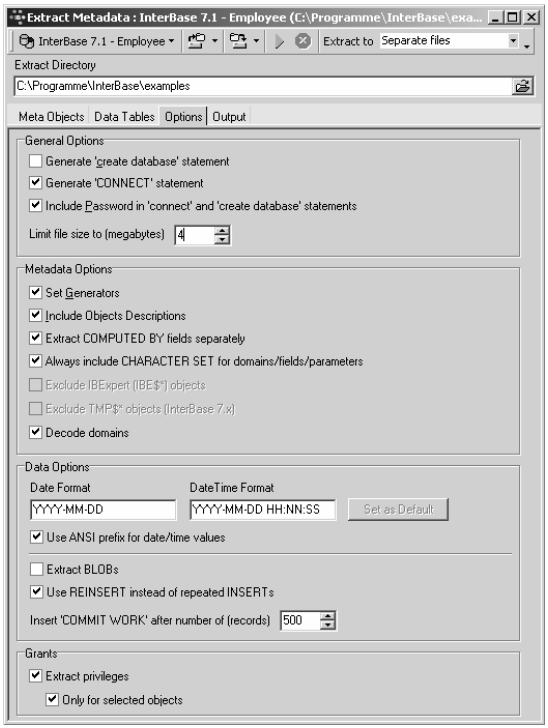


again using the <, >>, > or >> buttons, drag 'n' dropping or double-clicking.

By selecting one of the tables in the Selected Tables list, it is possible to add a `WHERE` clause, if wished.

Extract Metadata Options Page:

The Extract Metadata Options page offers a wide range of further options:



These include:

General Options:

- **Generate 'CREATE DATABASE' statement** - this determines whether a `CREATE DATABASE` statement should be included at the beginning of the generated script. If this option is unchecked, the `CONNECT statement` will be included instead.
- **Include password into 'CONNECT' and 'CREATE DATABASE' statements** - this determines whether the password should be included into the `CREATE DATABASE` or the `CONNECT` statement in the resulting SQL script.
- A **Limit File Size** option was added in IBExpert version 2004.9.12.1. This defines the maximum file size of the resulting script(s). When this option is specified and the maximum file size is reached, IBExpert automatically creates the next file with suffixes 0001, 0002 etc.

Metadata Options:

- **Set Generators** - If this option is checked, the `SET GENERATOR` statement for each generator will be included into the resulting script.
- **Include object descriptions** - this determines whether database objects descriptions should be included into the generated script. See "How does IBExpert extract objects descriptions?" for more details.

- **Extract COMPUTED BY fields separately** - this option can be used to specify whether computed fields should be extracted separately (useful if there are bugs in the database; realistically however this option is seldom used).
- **Always include the CHARACTER SET for domains/fields/parameters.**
- Since version 2004.2.26.1 there is also the added option **Decode domains.** If enabled, the domain types will be inserted as comments just after domain names. For example:

```
CREATE TABLE Z (
  B BOOL /* INTEGER DEFAULT 0 CHECK (VALUE IN(0,1)) */
);
```

Data Options:

- **Date Format** - this can be used to specify the date format and datetime format, with options to use an ANSI prefix for date/time values and to set the specified format as default.
- **Extract Blobs** - IBExpert cannot "read" blobs; it therefore uses a detour to make a reference to a separate database file containing such blobs. Only IBExpert has been able to do this so far. Other products only extract the definition of the blobs, and not the contents themselves.
- **Use REINSERT instead of repeated INSERTs** - uses the IBExpert `REINSERT` command, to insert multiple data records.
- **Insert 'COMMIT WORK' after number of (records)** – this option defines the number of records before inserting the `COMMIT` statement into the script. The default value is 500, i.e. 500 Insert commands are performed and then committed.

Grants:

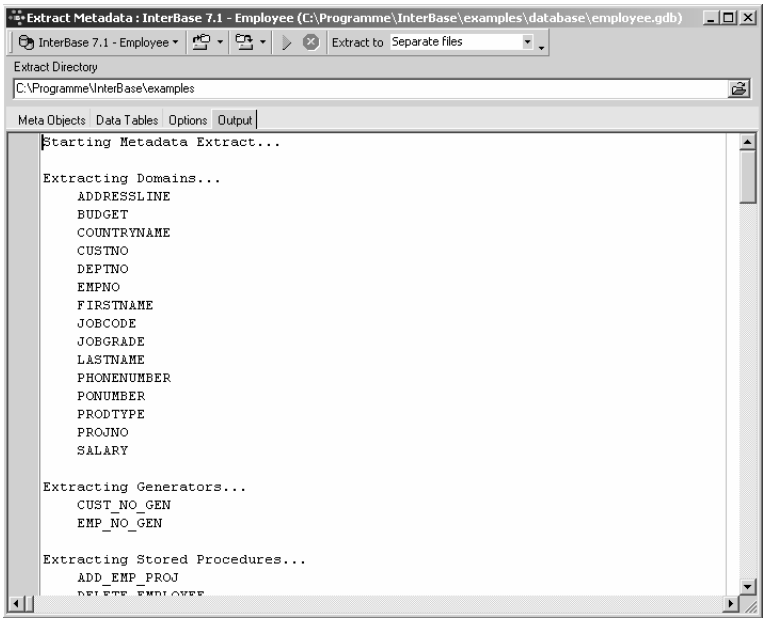
- **Extract privileges** - for all or only for selected objects.

Finally, if wished, use the toolbar icon Save Configuration to File or the key combination [Ctrl + S] to save this configuration as a template for future use. The next time round, the template can be quickly and easily loaded using the Load Configuration icon (or [Ctrl + L]); the template specifications amended if necessary, and the extract started!

Once all objects have been selected, and all options specified, the extract can be started using the green > button or [F9].

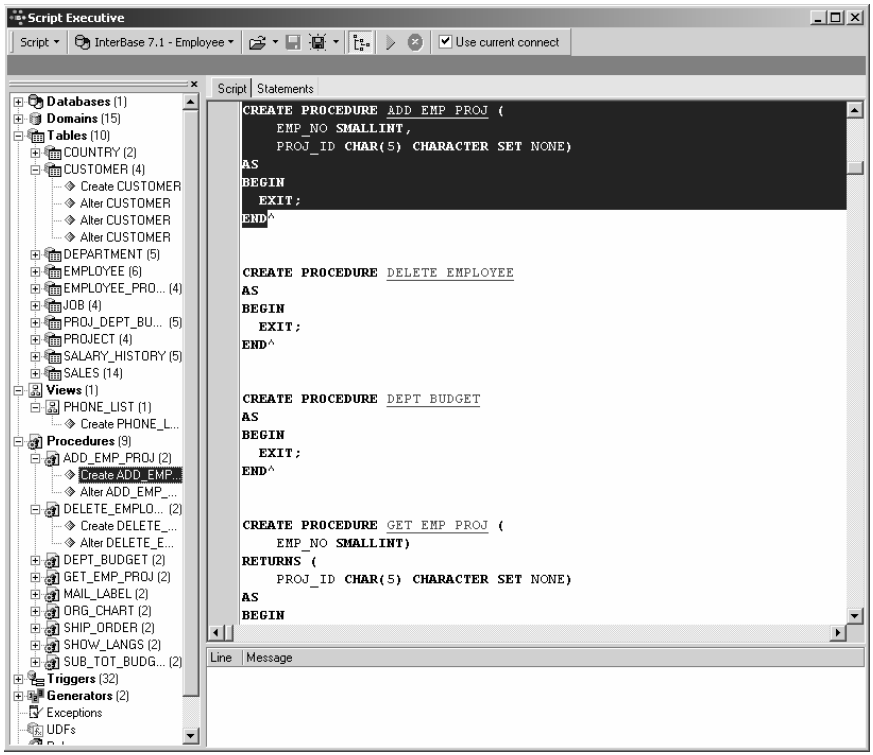
Output Page:

The Output page displays the IBExpert log during the extraction. Following completion, if a file was specified, IBExpert asks whether the file should be loaded into the script editor.



If the Script Executive has been specified as the output option, the Script Executive is automatically loaded.

The object tree on the left-hand side can be opened to display the individual statements relating to an object. By clicking on any of these statements, IBExpert springs to that part of SQL code, which is displayed on the right:



The statements display what IBExpert is doing and in which order. The script displays the creation of all objects, and then the subsequent insertion of the content data, using the ALTER command.

Extract Metadata is a great tool, and can be useful in a variety of situations. For example, it can be used to perform an incremental backup, should it be necessary for example, to back up just the EMPLOYEE table every evening.

8.13.1 Metadata

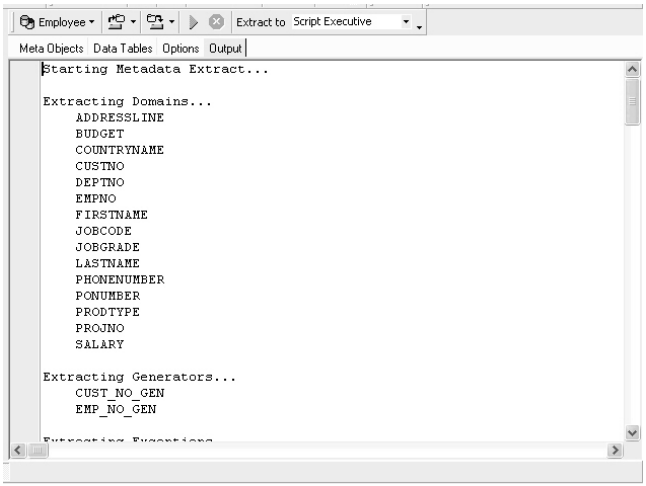
Metadata includes the definition of the database and database objects such as domains, generators, tables, constraints, indices, views, triggers, stored procedures, user-defined functions (UDFs), blob filters.

Metadata is stored in system tables, which are themselves part of every InterBase/Firebird database.

Metadata includes all those SQL statements necessary to recreate the database object. It includes the following elements:

- CREATE DATABASE statement
- CREATE DOMAIN statements
- CREATE TABLE statements

- declarative referential integrity using the ALTER TABLE statement
- CREATE GENERATOR statements
- CREATE VIEW statements
- check constraints using ALTER TABLE statements
- CREATE EXCEPTION statements
- procedure definitions using CREATE PROCEDURE or ALTER PROCEDURE
- trigger definitions using CREATE TRIGGER statements
- granting of user authorizations for tables, views and stored procedures.

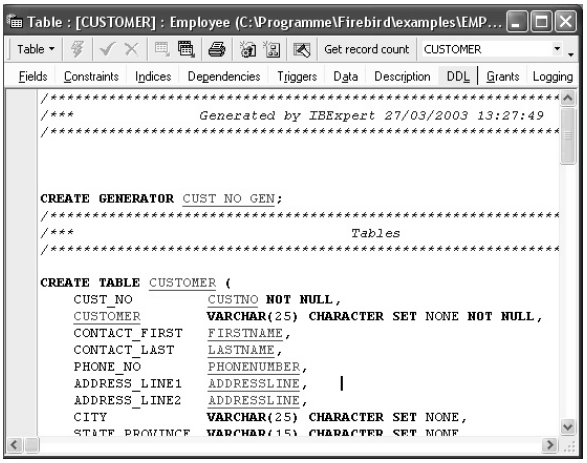


Metadata for a table includes all domains and generators used by these tables plus the CREATE TABLE statement.

It does not include any referential integrity definitions from this table to other tables or from other tables to this table.

Metadata for a view only includes the CREATE VIEW statement.

The current metadata definitions can be viewed on the DDL page in the individual object editors.



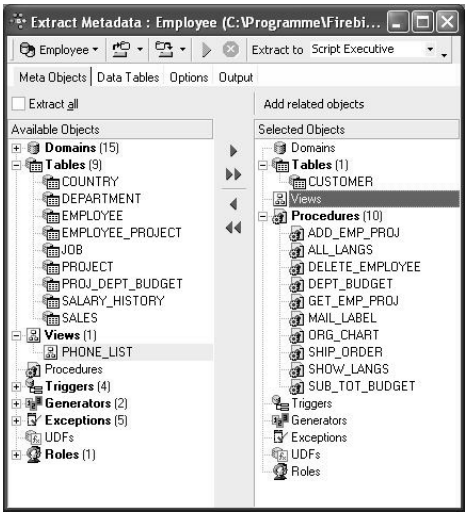
The IBExpert menu item Tools - Extract Metadata can be used to extract all metadata for a database. The resulting script can be used to create a new empty database. When the Options Data Tables and Options - Extract Blobs are used, the script contains the complete database with all data.

8

8.13.2 Select Objects Tree

The Select Objects Tree dialog can be found in the following editors:

- Extract Metadata Editor on the first page, Meta Objects,
- Generate HTML Documentation Editor, also on the Objects page, ,
- Print Metadata dialog.



The Select Objects Tree feature offers the user the choice whether to extract all database objects (check option), or specify individual objects, (using the < or > buttons, drag 'n' dropping the object names or double-clicking on them), or object groups (using the << or >> buttons, drag 'n' dropping the object headings or double-clicking on them).

Multiple objects can be selected using the [Ctrl] or [Shift] keys. There is even the option to *Add Related Objects* by using the button above the Selected Objects window.

8.13.3 How does IBExpert extract objects descriptions?

IBExpert uses a special extension of script language that enables it to extract objects' descriptions into script and then execute one using the Script Executive.

8.13.4 How does IBExpert extract blobs?

IBExpert uses an original mechanism to extract values of blob fields into a script. This allows you to store an entire database (metadata and data) in script files and execute these scripts with IBExpert. The following small example illustrates out method to extract blob values.

For example, a database has a table named COMMENTS:

```
CREATE TABLE COMMENTS (   COMMENT_ID INTEGER NOT NULL PRIMARY KEY,   COM-
MENT_TEXT BLOB SUBTYPE TEXT);
```


This table has three records:

COMMENT_ID	*COMMENT_TEXT*
1	First comment
2	NULL
3	Another comment

If the *Extract BLOBs* option is unchecked you will get following script:

```
CREATE TABLE COMMENTS (   COMMENT_ID INTEGER NOT NULL PRIMARY KEY,   COM-
MENT_TEXT BLOB SUBTYPE TEXT);

INSERT INTO COMMENTS (COMMENT_ID) VALUES (1);
INSERT INTO COMMENTS (COMMENT_ID) VALUES (2);
INSERT INTO COMMENTS (COMMENT_ID) VALUES (3);
```

... and, of course, you will lose your comments if you restore your database from this script.

But if the *Extract BLOBs* option is checked, IBExpert will generate a somewhat different script:

```
SET BLOBFILE 'C:\MY_SCRIPTS\RESULT.LOB';
```

```
CREATE TABLE COMMENTS (
   COMMENT_ID INTEGER NOT NULL PRIMARY KEY,
   COMMENT_TEXT BLOB SUBTYPE TEXT);

INSERT INTO COMMENTS (COMMENT_ID, COMMENT_TEXT) VALUES (1,
h0000000_0000000D);
INSERT INTO COMMENTS (COMMENT_ID, COMMENT_TEXT) VALUES (2, NULL);
INSERT INTO COMMENTS (COMMENT_ID, COMMENT_TEXT) VALUES (3,
h000000D_0000000F);
```

IBExpert also generates a special file with the extension LOB, where blob values are stored. In the current example `result.lob` will be 28 bytes long and its contents will be `First commentAnother comment`.

`SET BLOBFILE` is a special extension of script language that allows the IBExpert Script Executive to execute scripts containing references to blob field values.

8.13.5 Obtain current generator values

There are two methods to obtain the current generator values in a database. The first is using the IBExpert menu item Tools / Extract Metadata, where there is an option to set generators on the Options Page.

In Firebird this can also be done using a stored procedure:

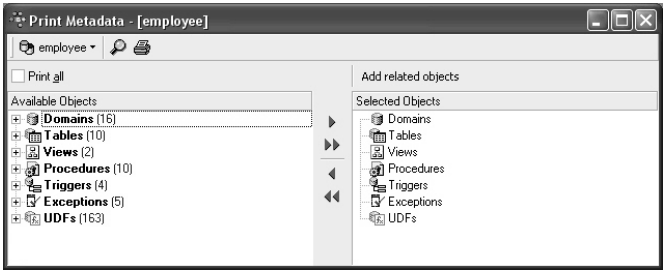
```
CREATE PROCEDURE GET_GENERATORS
RETURNS (
    GENERATOR_NAME CHAR(31),
    CURR_VAL BIGINT)
AS
declare variable sql varchar(100);
BEGIN
  FOR
    select r.rdb$generator_name generator_name, cast(0 as bigint)
curr_val from rdb$generators r
    where r.rdb$generator_name not containing '$'
    INTO :GENERATOR_NAME,
         :CURR_VAL
  DO
  BEGIN
    sql='Select gen_id('||GENERATOR_NAME||',0) from rdb$database';
    execute statement :sql into :curr_val;
    SUSPEND;
  END
END
```

8.14 Print Metadata

Print Metadata prints the database metadata, along with dependencies, description, DDL and other options for any database object or object group, providing a quick and yet extremely comprehensive database documentation. The information is printed as a report, using IBExpert's report templates. Using Report Manager, these reports can

also be customized (the Print Metadata standard report templates can be found in the IBExpert\Reports\ directory). This is of particular importance for those businesses working according to DIN certification/ISO standards.

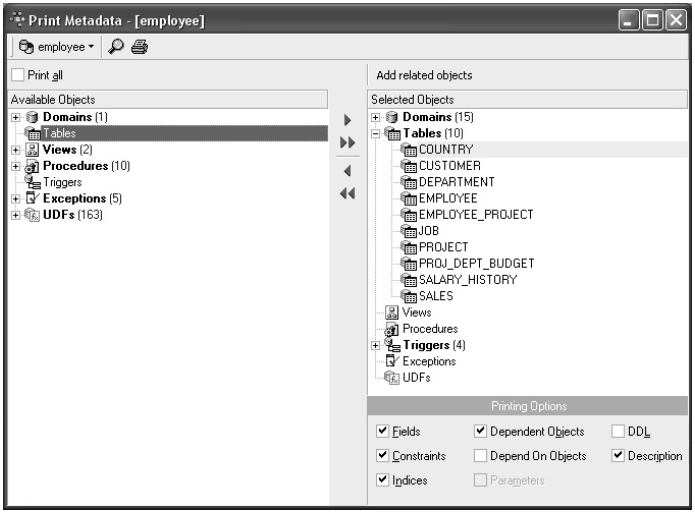
The Print Metadata menu item can be found in the IBExpert Tools menu, or started using the printer icon in the Tools toolbar.



The Print Metadata Editor is similar to the Extract Metadata Editor. First select one of the registered databases using the top left toolbar button. Then select the objects to be printed. It is possible to check Print All, or specify individual database objects (using the < or > buttons, drag 'n' dropping the object names or double-clicking on them), or object groups (using the << or >> buttons, drag 'n' dropping the object headings or double-clicking on them). Multiple objects can be selected using the [Ctrl] or [Shift] keys.

There is even the option to Add Related Objects by using the button above the Selected Objects window.

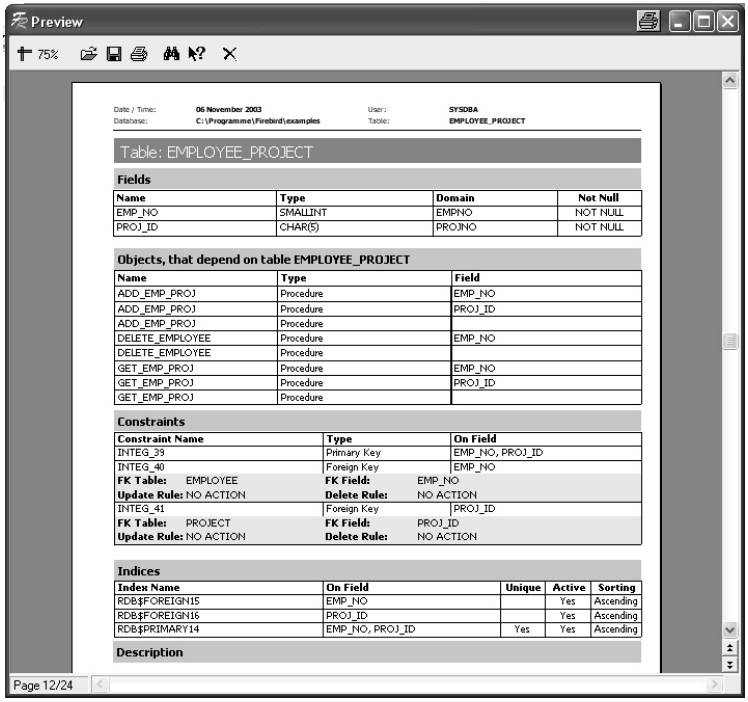
When one of the selected database objects or object groups is highlighted, a number of check options appear in the lower right panel. These include:



- fields
- constraints
- indices
- dependent objects
- depend on objects
- parameters
- DDL
- description

In order to print a complete database documentation it is of course necessary to select all database objects, and then check all options for each object group. This could however lead to difficulties in the case of very large databases, despite the Report Manager's amazing speed!

It is possible to print the report directly from this dialog or preview it first, using the magnifying glass icon.

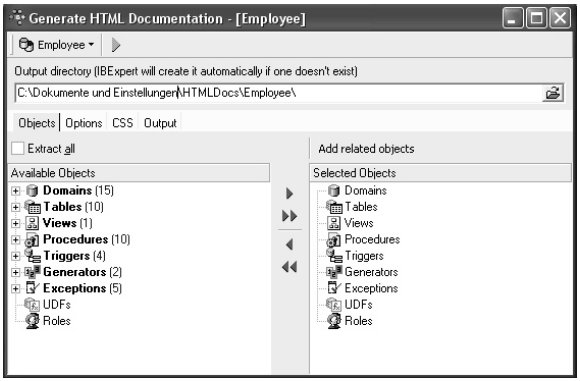


This opens the Fast Report Preview page, which displays the report as it will be printed, and furthermore offers options such as saving the report to file and searching for text.

8.15 Generate HTML Documentation

Using the IBExpert Tools menu, HTML documentation can be generated for a named, connected database.

This option is an excellent feature for software documentation, particularly if an object description was always inserted as objects were created. For those working with an older version of IBExpert: versions before 2.5.0.47 do not include all of the features detailed here.

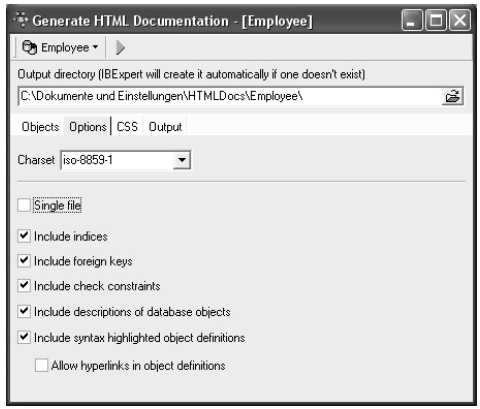


The toolbar displays the selected connected database. The pull-down lists offers a choice of all connected databases.

The default output directory can be overwritten if wished.

The Generate HTML Documentation Editor is similar to the Extract Metadata Editor. The Objects page allows single or groups of database objects to be selected for the HTML documentation. Database objects can be specified individually using the < or > buttons, drag 'n' dropping the object names or double-clicking on them, or object groups may be specified using the << or >> buttons, drag 'n' dropping the object headings or double-clicking on them. Multiple objects can be selected using the [Ctrl] or [Shift] keys. Alternatively the *Extract All* box can be checked, allowing documentation to be generated for the complete database.

There is even the option to Add Related Objects by using the button above the Selected Objects window.



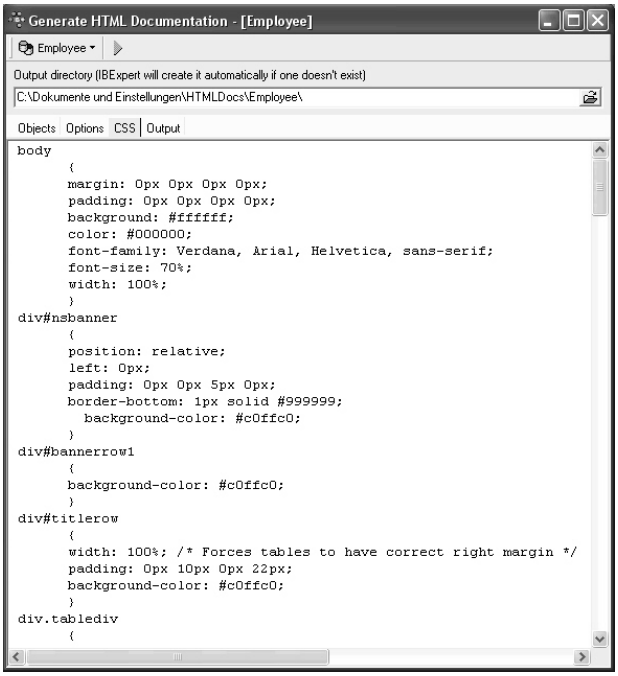
The Options page lists a series of check boxes including:

- single file (i.e. whether one complete file, as opposed to several smaller files should be generated)

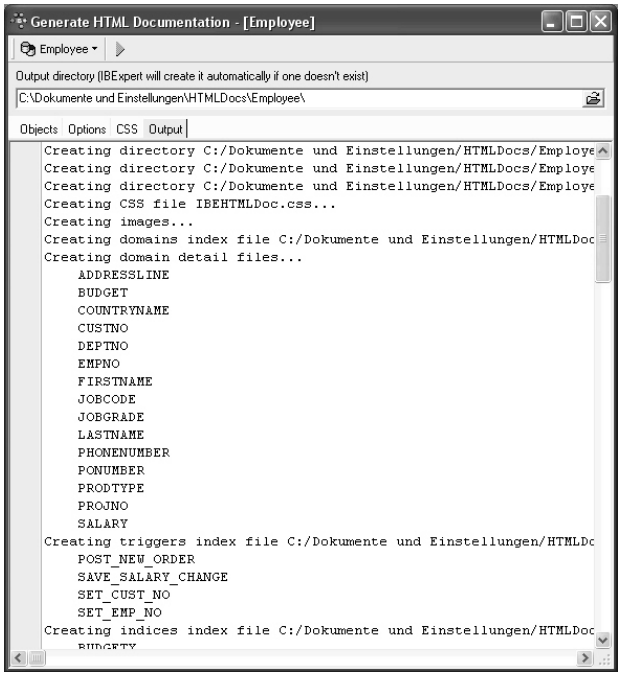
and whether:

- indices
- foreign keys
- check constraints
- database object descriptions
- syntax highlighted object definitions
- hyperlinks in object definitions

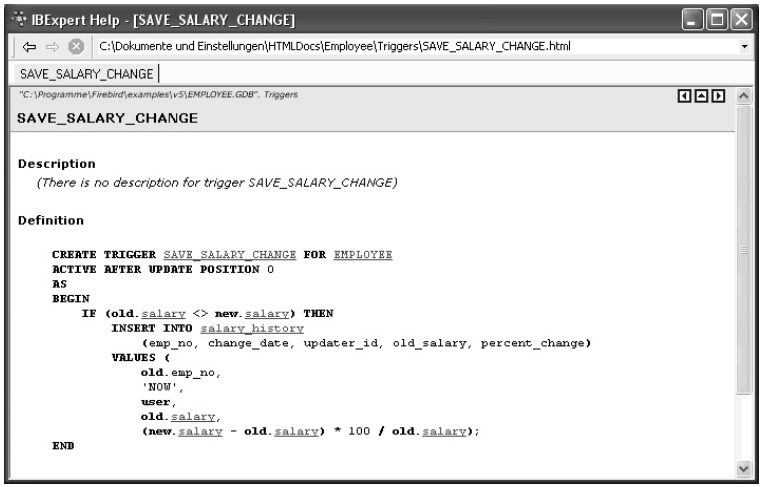
should be included.



The CSS or cascaded style sheets page displays the code for the HTML page template (please refer to CSS for more information).



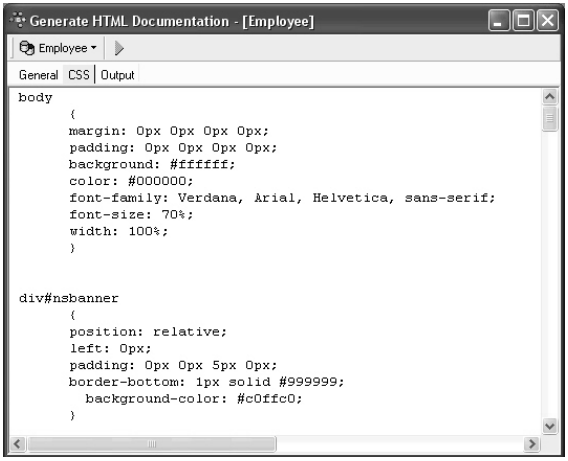
The Output page displays the code used to generate the HTML documentation.



By clicking on one of the object subjects, such as triggers, a table of all such objects (i.e. all triggers) for this database appear. Clicking on the individual objects then automatically displays the description (if existent) and the definition.

8.15.1 CSS – Cascaded Style Sheets

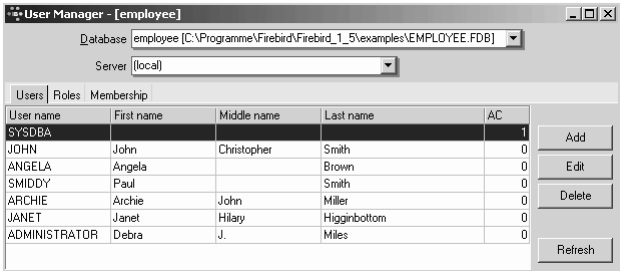
Cascaded style sheets (CSS) are an option included in the Generate HTML Documentation menu (second page in the main dialog). With knowledge of HTML these style sheets can be adapted as wished.



8.16 User Manager

The User Manager administrates database users and their roles. Here individual users can be allocated database and server access. The User Manager applies to the database server and not the individual database (please refer to server and database security for more information).

To start the User Manager select the Tools / User Manager menu item, or click the relevant icon in the Tools toolbar. The User Manager Editor displays all databases (drop-down list) on the current connected InterBase/Firebird server. The server connection may be altered using the pull-down list.



Select the database and server (local or remote) to administrate.

User rights for the database:

All users must be logged in, in order to access the server. What they are actually allowed to do on the server is then determined using the InterBase/Firebird GRANT and REVOKE commands (see Grant Manager for further details), or the front-end program.

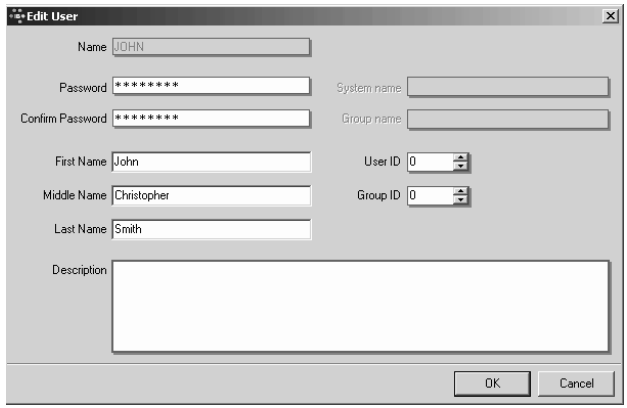
Note! To create, edit and delete users and roles you should have the rights of server administrator.

Users page:

On the Users page, a full list of users registered for the named server connection is displayed. Even if the selected database is not currently connected, the user list can still be seen. This is because the users are registered directly in the security database on the server, and can therefore be granted rights for all databases on this server. Since version 2.5.0.61 there is the additional column AC (Active Users) displayed in the users list. It shows how many active connections a user has to the specified database. This works only with active databases. And since version 2005.02.12.1 the Refresh button has been added to refresh a list of all users.

You may be asked for a password, when selecting an unconnected database, in order to ascertain your authority.

A user can be added by the SYSDBA (not a database owner, as users are created for all databases on the server). Simply click the Add button, and complete the New User form:



Support for the InterBase 7.5 embedded user authentication was added in IBExpert version 2004.12.12.1.

Again, only the SYSDBA or is allowed to edit or delete users. When editing, only the user name used for logging in may not be changed. It is here that a new password may be entered, if the user has forgotten his old one; or a change of name be input, for example, if a user marries.

This list contains currently existing users. To add, edit or delete users click buttons at the right of the list. In the Add / Edit User window set the user name and password and (optionally) his first, middle and last name.

Password:

The password is always user-oriented. Passwords are stored encrypted in the server database. When a user enters his password, this is passed onto the server, which compares the string entered with the string of the encrypted password stored on the server. The password is NEVER passed on from the server to the client.

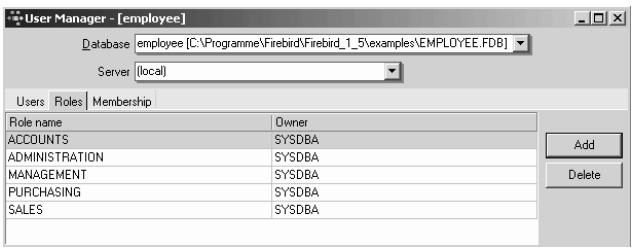
If a user forgets his password, the SYSDBA can enter a new one to replace the old one. Alternatively a UDF can be incorporated into the program, to allow the user to change his password himself, without having to disturb the SYSDBA or reveal the new password to a third person.

An example of such a UDF can be found in the `FreeUDFlib.dll`, which can be downloaded from www.ibexpert.com. (Please refer to FreeUDFLib for details.)

Users can be entered and assigned rights directly, although it often makes more sense if the majority of users are assigned user rights using roles. Roles are used to assign groups of people the same rights. When changes need to be made, only the role needs to be altered and each user individually. Please refer to roles for further information.

Roles page:

The Roles page can be used to create and delete roles exactly in the same way as with the database object Roles. All roles and their owners are displayed for the selected database. Other databases on the same server may be selected to display their full range of existing roles.

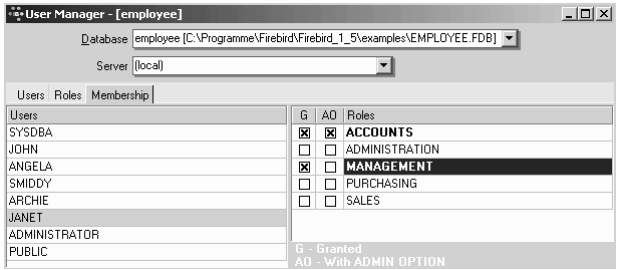


This list contains existing roles. To add or delete roles click buttons at the right of the list. When creating or deleting a role the Compile Window appears. Commit the transaction and if it is successful the new role is created or dropped. After the role has been created, users need to be added to the role. Role users and rights can be specified, edited and deleted using IBExpert's Grant Manager.

Roles can only be altered at system table level. They can however be deleted and new roles added using the User Manager.

Membership page:

The Membership page shows which users have been granted rights to which roles.



The abbreviations *G* and *AO* stand for *Granted* and *With Admin Option*. Users can be assigned roles simply by selecting the user, and checking either the Grant boxes or the Admin Option boxes. For example, all sales staff could be given the user name SALES with the role SALES. When logging into the system, both these names need to be entered. Checking the *Admin Option* automatically entitles the user to pass his rights on to other users.

8.16.1 Server security ISC4.GDB / SECURITY.FDB

When InterBase/Firebird is installed on a server, a database of authorized users is also installed. This is vital for server security, to protect the server from being accessed, manipulated or damaged by unauthorized users.

The database's security database is called `ISC4.GDB`; since Firebird 1.5 `SECURITY.FDB`, the change of suffix being due to Windows XP's eternal copying problems with .GDB files.

The `ISC4.GDB` provides a user page detailing rights for the InterBase/Firebird server. Here all users are entered, that are allowed to use the server. The user password is server-oriented and not database-oriented. It is important to employ users and rights to limit access and control manipulation, and is particularly advantageous, for example, to trace who has done what and when, as user names are included in the log.

Any user listed in the server security database's user list can open a database by providing the appropriate user name and password. If a user name and password is specified when the database is created, this user becomes the database owner. Only the SYSDBA and database owner are allowed to drop the database. If no database owner is specified at the time of database creation, then only the SYSDBA is authorized to drop the database.

If a user creates a table, InterBase/Firebird appoints that user as the table owner, and only the table owner and the SYSDBA are authorized to drop the table.

The SYSDBA and database owner can `GRANT`, `REVOKE` and grant access rights to users in the database; the SYSDBA and table owner can `GRANT`, `REVOKE` and grant access rights for tables. These rules also apply to views and stored procedures.

Simply allowing users into the database is not particularly helpful if they have not been granted access to the objects in this database. Therefore server security is administrated in IBExpert using the User Manager; user rights can then be assigned and controlled using the IBExpert Grant Manager.

Further security features include the following:

- Views - as they can be used to hide many table details from users; the users only have access to those columns and rows that they really need to see.
- Referential integrity - protects the data against orphaned rows and other operations, which could possibly damage the database integrity.
- GRANT and REVOKE statements - can be used in the IBExpert Grant Manager to specify which users may access which tables and views, and whether they are also allowed to manipulate data.
- An object may not be dropped if it is referenced elsewhere in the database. For example, a table cannot be dropped if it is referenced in a view, check constraint, trigger, stored procedure or other object.

8.16.2 Change user password per batch

To alter a user's password at command-line level, use the following syntax:

```
gsec -modify SYSDBA -pw password
```

or:

```
gesec -user SYSDBA -password oldpassword -modify SYSDBA -pw newpassword
```

An example for batch:

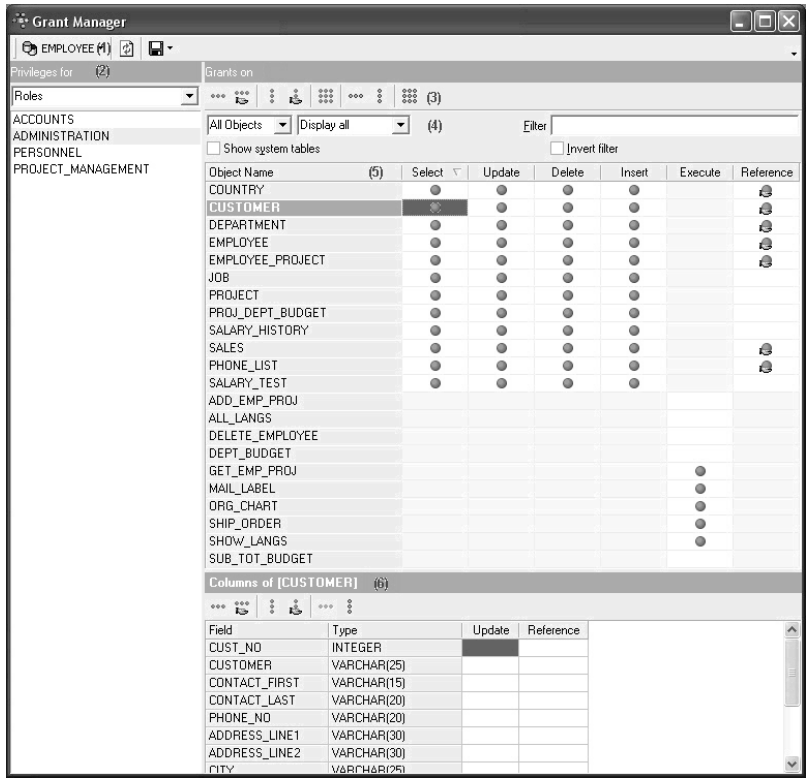
```
set isc_user=sysdba
set isc_password=masterke
gsec -add username -pw password
```

8.17 Grant Manager

The Grant Manager is used to administrate database security by controlling user permissions for a specific database. It allows you to specify the access rights for users, roles and database objects.

To start the Grant Manager select the menu item Tools / Grant Manager, use the respective icon in the Tools toolbar, or double-click on a role in the DB Explorer. Alternatively use the DB Explorer's right mouse-click menu item Edit Role or key combination [Ctrl + O]. This feature is unfortunately not included in the Personal Edition.

The Grant Manager Editor appears:



(1) Select Database: The toolbar displays the alias name for the current selected connected database. Another database on this server can be selected from the drop-down list at the top of the window

(2) Privileges for: The pull-down list (default = *Users*) allows a group for the processing of privileges to be selected. The options include:

- users,
- roles,
- views,
- triggers,
- procedures.

Once a database object has been selected, a full list of such users/objects in this database is displayed in the panel directly below.

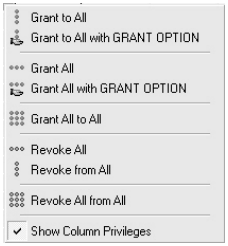
(3) Grants toolbar: The Grants toolbar enables the user to quickly assign or revoke rights to one or more objects, or for one or more operations. These can also be found in the right-click pop-up menu (see below).

(4) Filters: It is possible, using the pull-down lists, to specify exactly which grants should be displayed, i.e. for all database objects (default), just the tables, just the

views or just the procedures. Furthermore the user can determine whether all of the selected objects should be displayed, or only those with grants, or only those not granted. To the right of these pull-down lists is an empty filter field for user-defined filters. It is also possible to specify whether system tables should be included or the user-defined filter inverted, using the check boxes provided.

(5) The main window displays the object grants in a grid, displaying the granted operations (*Select*, *Update*, *Delete*, *Insert*, *Execute* and *Reference*) for the listed objects. A green circle indicates that access for this operation on this database object has been granted; a green circle held by a hand indicates that the *GRANT WITH GRANT AUTHORITY* option has been granted. An empty field indicates logically that either no rights have been granted, or they have been revoked.

The right-click pop-up menu offers the various GRANT and REVOKE options also displayed in the Grant Manager toolbar.



A further menu option here is *Show Column Privileges* (checkbox). This blends the lower window in and out (6), which displays the individual columns for tables and views, allowing *Update* and *Reference* rights to be granted and revoked for individual fields in the selected object.

Rights can be simply granted and revoked by double-clicking (or using the space bar) on the grid fields (in both the upper (object) and lower (column) windows). Alternatively, to assign several rights (i.e. *select*, *update*, *delete* and *insert*) to a single object or to assign one operative right to all objects displayed, use either the Grant Manager toolbar or the right-click menu.

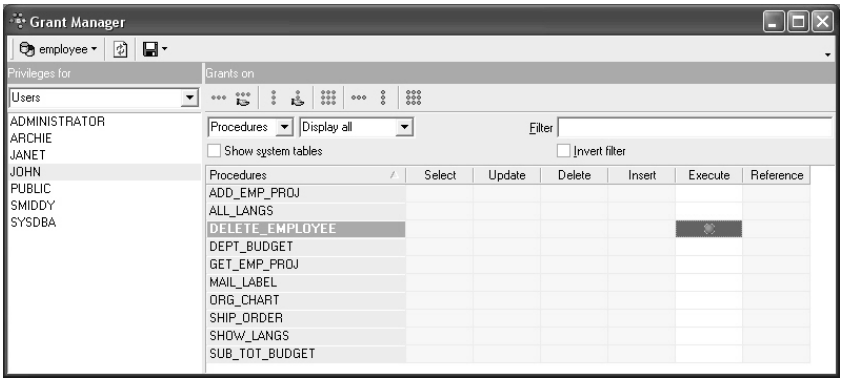
Please note that *Reference* rights only allow the user to read data sets, if there is a foreign key relationship to other data. And the Grant All to All command may only be performed by the database owner or the SYSDBA.

The majority of these operations can also be performed in the Grants pages, found in the individual database object editors. These were introduced to remind the developer not to forget the assignment of rights! They allow the developer to check existing rights for the object concerned and, if necessary, subsequently assign rights for a new or existing object.

Rights are however in practice usually administered at the front end. There is, as a rule, only one system user, with which the program can log into the database. For those preferring direct SQL input, please refer to GRANT and REVOKE.

8.17.1 Granting access to stored procedures

To grant a user the right to execute stored procedures, use the IBExpert Grant Manager EXECUTE column:



or the SQL EXECUTE statement. For example, to grant Janet and John the right to execute the stored procedure SP_Delete_Employee, use the following:

```
GRANT EXECUTE
ON PROCEDURE SP_Delete_Employee
TO Janet, John;
```

InterBase/Firebird considers stored procedures as virtual users of the database. If a stored procedure modifies a table, the procedure needs the relevant privileges on that table. So the user only needs EXECUTE privileges on the procedure and not any separate rights for the table. In this situation, the stored procedure performs the changes on behalf of the user.

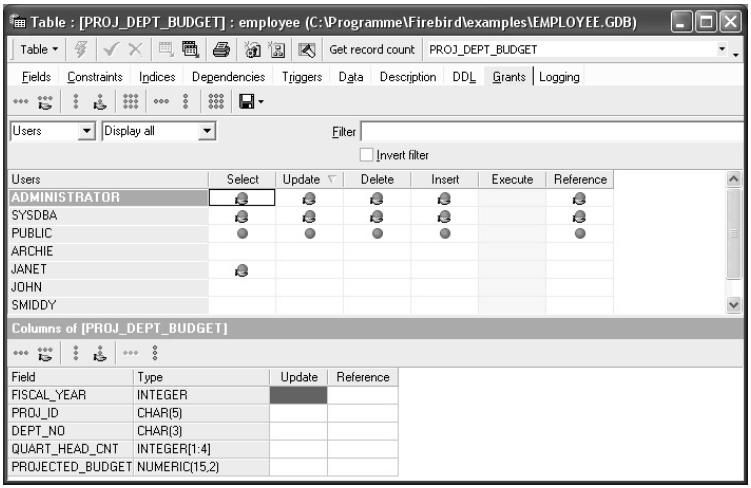
If a stored procedure needs the ability to execute another stored procedure, simply select Procedures from the *Privileges For* list and Procedures from the *Grants On* list, to grant the EXECUTE privilege on the desired procedure. Using SQL the GRANT statement is used, naming the procedure instead of one or more users (<user_list>).

8.17.2 Using the GRANT AUTHORITY option

A user, that has been granted certain privileges, may also be assigned the authority to grant those privileges in turn to other users. This is known as assigning grant authority.

InterBase/Firebird allows by default only the creator of a table and the SYSDBA to grant additional privileges onto other users.

Grant authority can be assigned in the IBExpert Grant Manager or the Grants pages in the relevant object editors, using the *Grant All with GRANT OPTION* or the *Grant to All with GRANT OPTION* icons or right-click menu items:

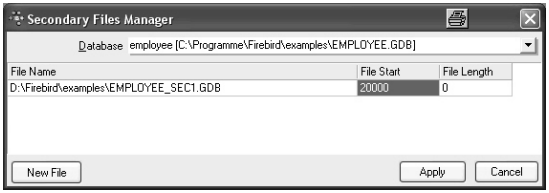


It is also simple to see which grant authorities have already been assigned to which users and roles.

In SQL the `WITH GRANT OPTION` clause may be used in conjunction with a grant of privileges, to assign users the authority to grant their privileges in turn to other users (refer to `GRANT` statement for the full syntax and examples).

8.18 Secondary Files Manager

The Secondary Files Manager can be found in the IBExpert Tools menu.



First select the database for which the secondary files are to be created, from the pull-down list of connected databases.

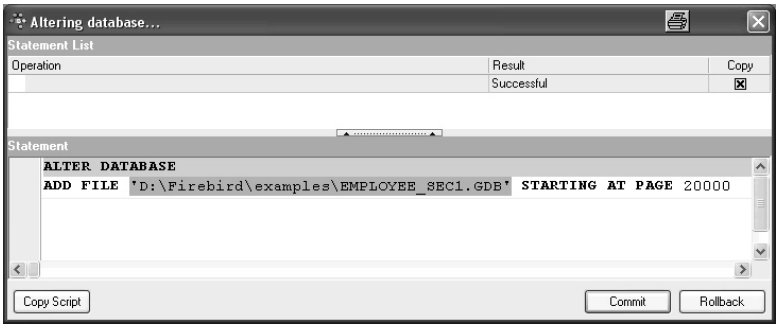
Then simply click on the *New File* button (bottom left corner) to specify a secondary file. As a database file is being created here, it is important not to forget to also specify the drive and path, as well as the file name and suffix (usually `.GDB`). Otherwise the file will be created and stored anywhere on the system (usually in the Windows System32 folder). Should this happen, the file drive and path can be viewed when the Secondary Files Manager is restarted.

After specifying the secondary file's name, either the starting page (*File Start*) or length in pages (*File Length*) can be specified by selecting the field, and clicking or using the space bar to activate the counter or allow numerical entry. Specifying both

these parameters is unnecessary, and only provides an error source, as the starting pages of two files must of course concur with the number of pages of the first file.

When using the IBExpert Secondary Files Manager, the first secondary file starts at the current position in the primary file, i.e. the primary file is immediately considered to be "full", and all new data and metadata from this point onwards is stored in this first secondary file. This can be viewed in the IBExpert Services / Database Statistics. See below for the specification of the primary file size at the time of database creation. Of course, multiple secondary files may be specified here if wished. It is not necessary to specify the length of the last secondary file; this can therefore become as large as the physical disk space allows.

When all files have been specified satisfactorily, simply click the *Apply* button,



and check before finally committing or rolling back.

There are no performance advantages to be expected by distributing the database across several files, so it is not recommended that secondary files be used, unless the disk storage space and database size absolutely require it.

The secondary files' size, path and name can only be altered when the database is restored, as this is the only option which allows secondary files to be redefined.

For those preferring direct SQL input the syntax is as follows:

```
CREATE DATABASE "database name"
LENGTH <number > PAGES
FILE <secondary file 1> LENGTH <number> PAGES
FILE <secondary file 2> LENGTH <number> PAGES
...
FILE <secondary file N>;
```

The alternative syntax, using STARTING (AT PAGE), is as follows:

```
CREATE DATABASE "database name"
FILE <secondary file 1> STARTING AT PAGE <number>
FILE <secondary file 2> STARTING AT PAGE <number>
...
FILE <secondary file N> STARTING AT PAGE <number>;
```

The AT and PAGE keywords are optional. InterBase/Firebird recognizes any of the following variations:

```
STARTING AT PAGE 5000
STARTING AT 5000
STARTING 5000
```

Please note that when a database is dropped/deleted, all secondary and shadow files are also deleted. The complete structure and all the data is permanently deleted!

8.18.1 Primary file

A database's primary file is the main database file. If no secondary files are specified, it is the only database file.

When secondary files are used, the length in pages needs to be specified for the primary file, or alternatively the first secondary file needs to be specifies with the STARTING (AT PAGE) parameter.

Primary and secondary files can be specified in the IBExpert Tools / Secondary Files Manager.

8.18.2 Secondary files

One or more secondary files may be specified by the database creator, to be used for database storage once the primary file has reached its specified limit. The database can be distributed across as many secondary files as wished.

Usually InterBase/Firebird databases grow dynamically, when database objects, program code or data are added. The only limitations are the physical limits of the hard disk or file system on which the database is stored.

Some file systems such as, for example, HP UNIX have additional limitations which do not enable the partition size to go over two Gigabytes. To avoid such a limitation, the InterBase database can be spanned across multiple file systems. Each file can be assigned a maximum size. Due to the automatic administration in InterBase/Firebird, the primary file is first filled until the maximum page size has been reached. Subsequent information is then packed into the secondary files until their capacity has been reached. As many secondary files can be created as wished.

Since InterBase 6.5/Firebird secondary files are really no longer necessary. In those particular cases, where secondary files may need to be considered, please consult the respective database Release Notes.

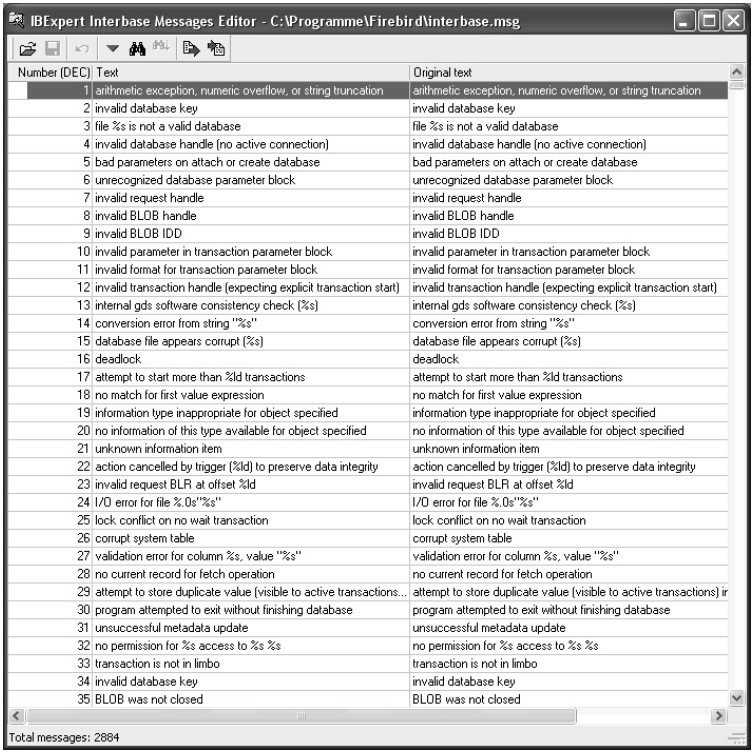
There are no performance advantages to be expected by distributing the database across several files, so it is not recommended that secondary files be used, unless the disk storage space and database size absolutely require it.

Secondary files can be simply and easily created using the IBExpert menu item Tools / Secondary Files Manager.

Please note that when a database is dropped/deleted, all secondary and shadow files are also deleted. The complete structure and all the data is permanently deleted!

8.19 Localize IB Messages

Localize IB Messages can be found in the IBExpert Tools menu. It enables the user to translate InterBase/Firebird messages into another language.



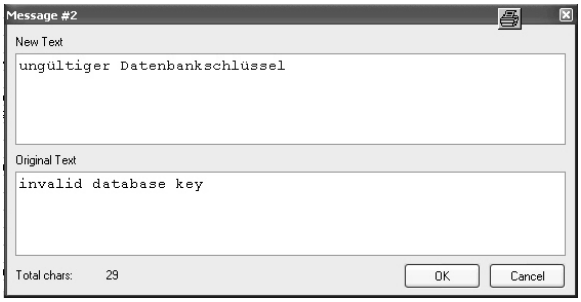
Number (DEC)	Text	Original text
1	arithmetic exception, numeric overflow, or string truncation	arithmetic exception, numeric overflow, or string truncation
2	invalid database key	invalid database key
3	file %s is not a valid database	file %s is not a valid database
4	invalid database handle (no active connection)	invalid database handle (no active connection)
5	bad parameters on attach or create database	bad parameters on attach or create database
6	unrecognized database parameter block	unrecognized database parameter block
7	invalid request handle	invalid request handle
8	invalid BLOB handle	invalid BLOB handle
9	invalid BLOB IDD	invalid BLOB IDD
10	invalid parameter in transaction parameter block	invalid parameter in transaction parameter block
11	invalid format for transaction parameter block	invalid format for transaction parameter block
12	invalid transaction handle (expecting explicit transaction start)	invalid transaction handle (expecting explicit transaction start)
13	internal gds software consistency check (%s)	internal gds software consistency check (%s)
14	conversion error from string ''%s''	conversion error from string ''%s''
15	database file appears corrupt (%s)	database file appears corrupt (%s)
16	deadlock	deadlock
17	attempt to start more than %ld transactions	attempt to start more than %ld transactions
18	no match for first value expression	no match for first value expression
19	information type inappropriate for object specified	information type inappropriate for object specified
20	no information of this type available for object specified	no information of this type available for object specified
21	unknown information item	unknown information item
22	action cancelled by trigger (%ld) to preserve data integrity	action cancelled by trigger (%ld) to preserve data integrity
23	invalid request BLR at offset %ld	invalid request BLR at offset %ld
24	I/O error for file %.0s''%s''	I/O error for file %.0s''%s''
25	lock conflict on no wait transaction	lock conflict on no wait transaction
26	corrupt system table	corrupt system table
27	validation error for column %s, value ''%s''	validation error for column %s, value ''%s''
28	no current record for fetch operation	no current record for fetch operation
29	attempt to store duplicate value (visible to active transactions...	attempt to store duplicate value (visible to active transactions) ir
30	program attempted to exit without finishing database	program attempted to exit without finishing database
31	unsuccessful metadata update	unsuccessful metadata update
32	no permission for %s access to %s %s	no permission for %s access to %s %s
33	transaction is not in limbo	transaction is not in limbo
34	invalid database key	invalid database key
35	BLOB was not closed	BLOB was not closed

The InterBase/Firebird messages can be loaded by clicking on the *Open File* icon and specifying the drive and path (`Firebird\interbase.msg or Inter-Base\interbase.msg`).

The messages are displayed in tabular form. The first column displays the message number (the total number of messages is displayed in the status bar).

The second column shows the editable text; the third column the original English text.

To translate a message, simply double-click to open the Edit window, enter the desired translation, confirm to return to the main window, and save (or undo). When saving it is recommended a new file name be specified, for example `interbase_german.msg`, as otherwise the original English text is overwritten by the translation.

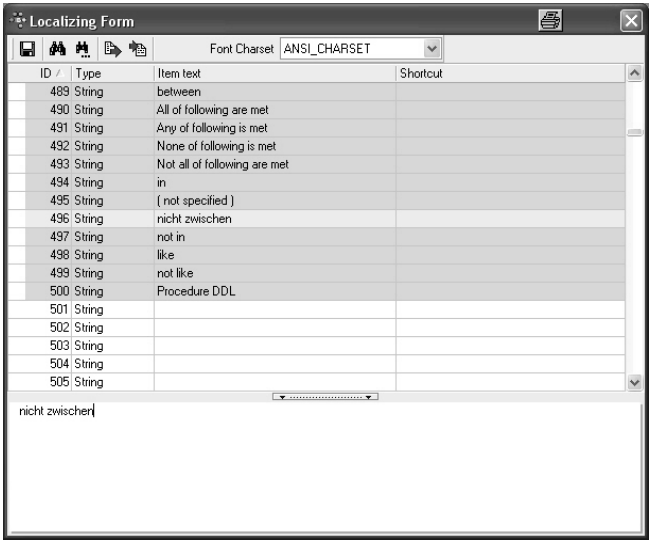


Other options offered in the Localize IB Messages toolbar include:

- **Save to File** - saves all changes to the file named in the title bar.
- **Undo** - allows the message text to be reverted to the original, provided it has not yet been saved to file.
- **Goto Message Number** - spring to specified message number.
- **Find and Search Again** - search options for finding message texts.
- **Export to Text File** - enables the message list to be exported to a text file.
- **Import from Text File** - allows a message list to be loaded from a text file as opposed to loading the `interbase.msg` file).

8.20 Localize IBExpert

Localize IBExpert can be found in the IBExpert Tools menu. It enables the user to translate InterBase/Firebird messages into another language.



The InterBase/Firebird messages are automatically loaded. An alternative Font Character Set may be selected if necessary from the pull-down list offered in the Localize IBExpert toolbar.

The Localizing Form displays all IBExpert messages in a tabular form. The first column displays the ID number (there are 2,999 ID records altogether). The second column shows the message type (e.g. string), the third the editable item text; and the fourth column the respective shortcut. Initially pink highlighted records show messages already created and assigned in the original English version. Blank rows (non-highlighted) indicate non-assigned messages.

To translate a message, simply select it, enter the desired translation in the lower editing panel and save. When saving it is recommended an new file name be specified, for example `interbase_german.msg`, as otherwise the original English text is overwritten by the translation.

Other options offered in the Localize IBExpert toolbar include:

- **Save to File** - saves all changes to the file named in the title bar.
- **Find and Search Again** - search options for finding message texts.
- **Export to Text File** - enables the message list to be exported to a text file.
- **Import from Text File** - allows a message list to be loaded from a text file (as opposed to loading the standard IBExpert original English file).

If you have succeeded in translating this file into a language that IBExpert does not yet offer, please contact hk@ibexpert.com. We would love to hear from you!

8.20.1 Find IBExpert Message

This Search dialog is useful for finding individual words or word strings in the long lists of IBExpert language translations. It can be called using the Binocular icon in the Localizing Form toolbar. The dialog offers a number of options:

The Text to Find field allows direct input, or the pull-down list may be used to select a text recently searched for.

The Direction *forward* (default) or *backward* may be selected, as well as the area to be searched (from a *selected area* or across the *entire scope*).

Use the OK button to spring to the first occurrence of the text specified.

The

icon can be used to search for further occurrences, should any exist, of the specified string.

8.21 Report Manager

Using the menu item Tools / Report Manager or the respective icon in the Tools toolbar, the Report Manager dialog is opened. (This feature is unfortunately not included in the Personal Edition.)

A new report can be created on any volume or in the database (double-click on a database entry to create the necessary objects automatically). To edit the report, just use [Ctrl+D] and the editor will open. To create a new report, simply right -click on *Page1 header* and add a new dialog form. On this form you can add a database and one or more query components. Go back to *Page1* and Insert some bands and rectangular objects. All data connections can be viewed in the Object Inspector or following a double click.

Take a look at http://www.fast-report.com/ to see some examples and the original components, which can be used in any Delphi/CBuilder project as an extremely powerful, quick and stable replacement for Quickreport and other report tools.

We personally have still not found anything that Fast Report cannot do!

8.22 Blob Viewer/Editor

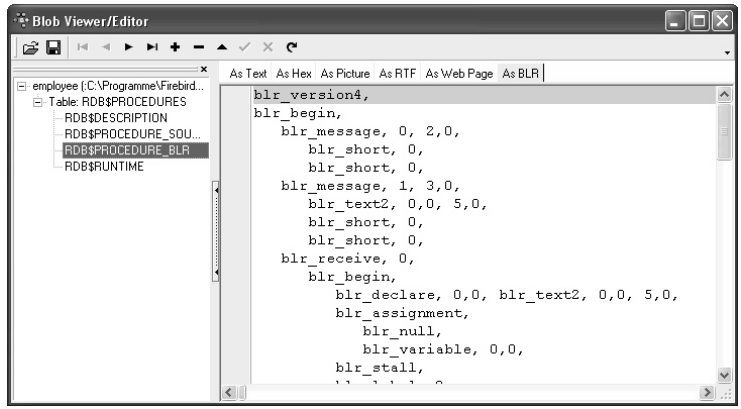
The IBExpert Blob Viewer/Editor can be found in the Tools menu.

(This feature is unfortunately not included in the Personal Edition.)

It enables blob fields in an open grid (e.g. the Table Editor / Data page, the SQL Editor / Results page) to be viewed as *Text*, *Hex*, *Picture*, *RTF* or as a web page.

The individual fields in the blob column can be viewed and navigated using the editor's navigational toolbar (please refer to Blob Viewer/Editor toolbar for details).

New to IBExpert version 2003.11.6.1 is the added syntax highlighting for SQL. This is useful if your blobs contain SQL queries. Furthermore, there is now a new As BLR page. This allows blobs with subtype 2 data to be displayed:



This shows what is really physically in the database.

Since IBExpert version 2005.03.12 there is also added support for PNG (Portable Network Graphics) images.

8.23 Database Designer

The IBExpert Database Designer is a comprehensive tool, which allows database objects to be managed visually. It can be used to represent an existing database visually, or create a new database model, and then create a new database, based upon this model. It is possible to add, edit and drop tables and views, edit table fields, set links between tables, edit and drop procedures, and so on. This feature is unfortunately not included in the Personal Edition.

The Database Designer can be started from the IBExpert Tools menu.

The Designer Menu offers the following options:

- Reverse Engineer ...
- Generate Script ...
- New Diagram
- Load Diagram from File
- Save Diagram
- Export ...
- Print
- Manage Subject Areas
- Manage Layers
- Model Options...

There are also a number of toolbars (please refer to Database Designer toolbars for further information).

Using the Designer menu items or icons, an existing diagram can be opened from file, or a new diagram created.

Reverse Engineering will be used here for the sake of demonstration. (Please refer to Reverse Engineer for details.) By simply creating a model of the sample EMPLOYEE database using the Reverse Engineer ... menu item, it is possible to view and test the many features the Database Designer has to offer.

The magnifying glass icons in the Menu and Palette toolbar can be used to increase or reduce the diagram size. Using the pointer icon (= normal editing mode), tables and views can be selected by clicking on them with the mouse, or dragged 'n' dropped as wished; the connecting lines (= links) automatically move as well.

Insert new tables or views by simply clicking on the relevant icon in the Palette toolbar, and positioning in the main diagram area. Since IBExpert version 2004.10.30.1 templates can be used (IBExpert menu item Environment Options / Templates) to create foreign and constraint names automatically.

Alternatively, existing objects may be dragged and dropped from the DB Explorer and SQL Assistant into the main editing area. When an object node(s) is dragged from the DB Explorer or SQL Assistant, IBExpert will offer various versions of text to be inserted into the Code Editor. It is also possible to customize the highlighting of variables. Use Options / Editor Options / Color to choose color and font style for variables.

And since IBExpert version 2004.9.12.1 there is now a Model Navigator in the SQL Assistant, enabling you to navigate models quickly. Use the corresponding tab in the SQL Assistant (Model Navigator). The Database Explorer now offers an additional Diagrams page, displaying all objects in the database model in a tree form. Simply click on any object, and it is automatically marked for editing in the main Database Designer window.

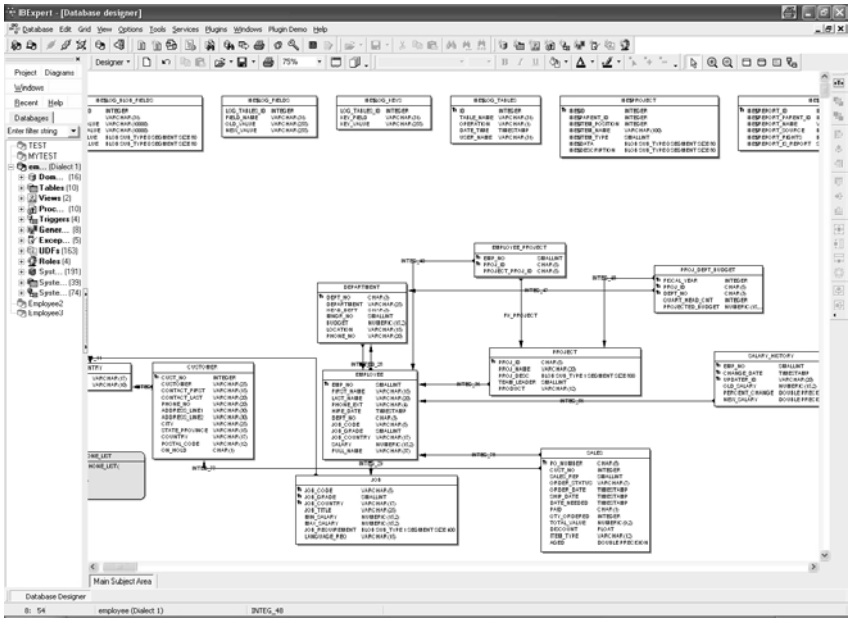
The Comment box icon allows comments to be added to the diagram. Insert and position a comment box, double-click to add the comment text in the Model Options window on the Database Designer Comment Box page.

Reference lines, i.e. foreign key relationships can be drawn between tables/views using the right-hand icon in the Menu and Palette toolbar, and dragging the mouse from one table to the next.

Context-sensitive right-click menus offer a number of options for selected tables, views or links (please refer to Database Designer Right-Click Menus for further information).

Double clicking on any table or view opens the Model Options menu item in the lower window, where information can be viewed, altered or specified. Please refer to Model Options for further information.

By double-clicking on the line between two tables, the relationships are shown in detail. The name and automatic tracing of links are options, as already mentioned, included in Model Options.



Database objects may be grouped using the [Shift] key and selecting objects with the mouse, and then using the respective Layout toolbar icons to group or ungroup objects.

Objects can also be aligned (left, center, right, top, middle, bottom), again by holding the [Shift] key and selecting objects with the mouse, and using the respective Layout icons. Using these key combinations, it is also possible to select a group of objects, and make them the same size, height or width, size to grid, or center horizontally or vertically.

And since IBExpert version 2005.03.12 there is the added option, using the right-click context-sensitive menus, to lock visual objects, to protect them against casual modification of size and position.

Don't forget, the white pointer icon returns the mouse to the normal editing mode!

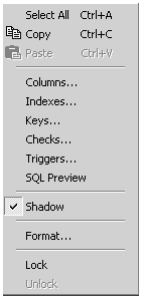
It is also possible to Manage Subject Areas and Manage Subject Layers (please refer to these subjects for further information).

When the database model has been designed/altered as wished, a script can be generated (please refer to Generate Script) and executed, to apply these alterations to the database itself.

8.23.1 Database Designer right–click menus

The main Database Designer design area offers a selection of context-sensitive right-click menus.

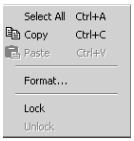
When a table is selected, the following options are offered:



These include options to Select All, Copy and Paste; Columns, Indexes, Keys, Checks, Triggers and SQL Preview are those options also offered in the Model Options dialog in the lower part of the screen; a check box to specify whether a selected table should be depicted with a shadow or not; and Format. This menu item opens a new dialog - for tables however, this only offers the shadow option, also listed as a check option in the menu.

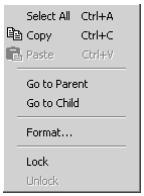
The Lock / Unlock option is new to IBExpert version 2005.03.12, and allows visual objects to be locked, to protect them against casual modification of size and position.

When a view is selected, the right-click menu offers the following options:



Again the option to Select All, Copy and Paste is offered, along with the Format option. This dialog must be opened and the shadow option checked or unchecked, if the appearance of the view is to be altered.

When a link is selected, the following options are offered:

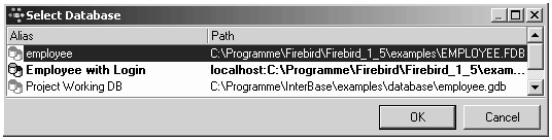


Again there is the option to Select All, Copy and Paste. Furthermore, it is possible to spring to either the Parent or Child (i.e. primary key able or foreign key table), and again the Format option opens a new dialog, where, on the Links page, the rounded corners option may be checked or unchecked as wished.

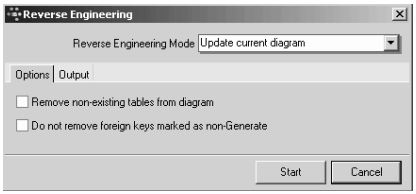
8.23.2 Reverse Engineer

Reverse engineering creates a diagram of an existing database.

When reverse engineering, select the database to be visually displayed from the list of registered databases.

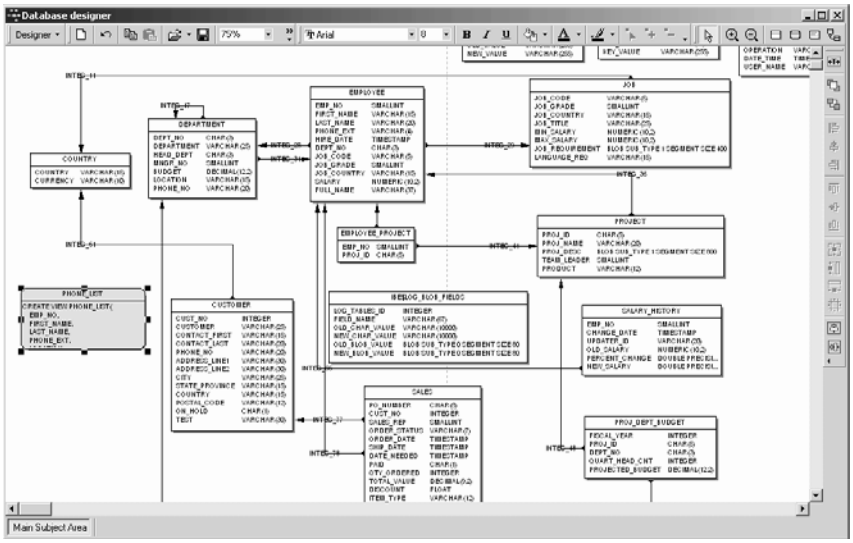


In the case of the selection of an unconnected database, IBExpert asks whether it should connect. Specify whether a new diagram should be created or an existing one updated, and check the *Clear Diagram* option if necessary:



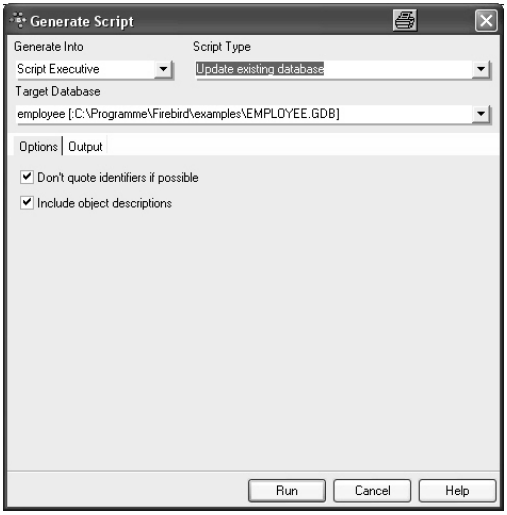
The option Do not remove foreign keys marked as non-Generate was added in IBExpert version 2004.12.12.1 and is useful to prevent fake relationships from being deleted.

Start the reverse engineering, and see how quickly IBExpert creates a diagram of the database:



8.23.3 Generate Script

It is also possible to generate a script for the model using the Generate Script menu item. This is necessary in order to apply any changes made to the model to the database itself.



The script can be generated into the Script Executive, to a file or to clipboard. The Script Type options include:

- Create new database
- Update existing database

- Difference script (for testing only)

Specify the file name if saving to file and check/uncheck the options

- Don't quote identifiers if possible
- Include object descriptions

if wished, and run.

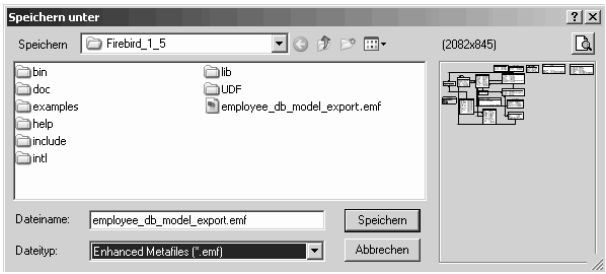
Since version IBExpert 2004.8.5.1 generators and triggers are now processed during generation of the update database script. View dependencies are also now taken into account when the script is generated.

Since IBExpert version 2004.9.12.1 the `SET NAMES`, `SET SQL DIALECT`, `CREATE DATABASE` statements were removed from the resulting `CREATE DATABASE` script. You now need to use the model prescript (Model Options) to specify necessary `INIT` statements.

The generation of update scripts was improved in IBExpert version 2004.12.12.1. to include analysis of exceptions and procedures.

8.23.4 Export

The database model can be exported, either as a bitmap (`.bmp`) or an enhanced metafile (`.emf`). This is new to IBExpert version 2.5.0.61. Simply load the model to be exported, click the Export menu item, and specify the name and format.

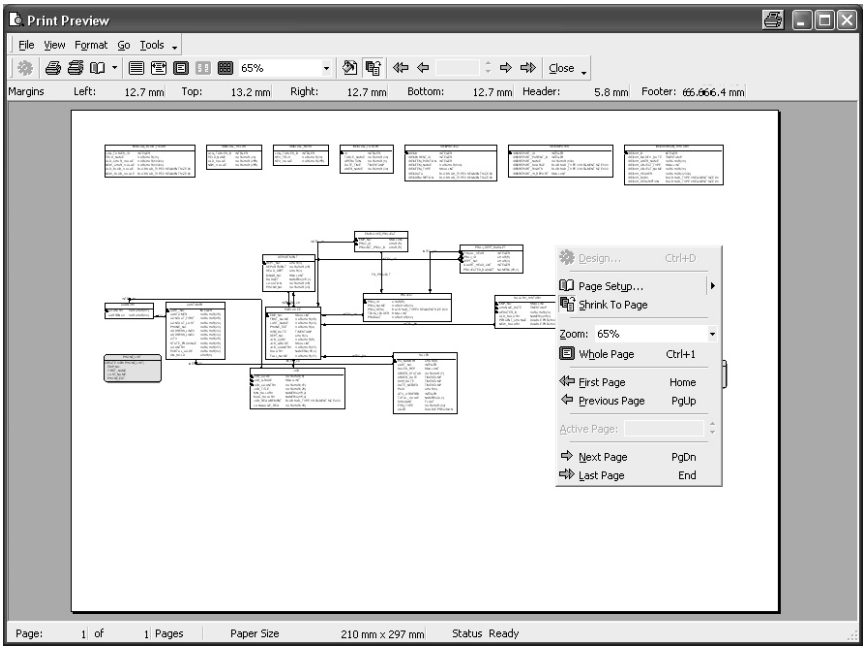


8.23.5 Print

The database model can be printed, using the respective Database Designer menu item or icon. This option firstly produces a print preview, allowing adjustments to be made before printing.

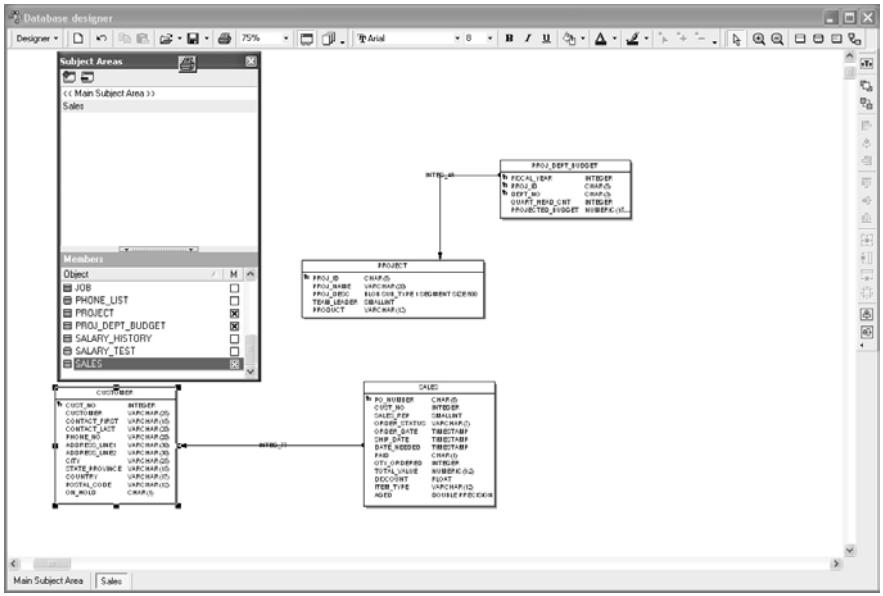
New Features:

- Since IBExpert version 2.5.0.61 it is possible to store printing options between sessions.
- Since version 2003.12.18.1 it is now possible to display borders of pages (printable parts) with dashed lines. You can customize the page options (size, headers and footers etc.) using the Print Preview form.



8.23.6 Manage Subject Areas

The IBExpert Database Designer menu item Manage Subject Areas is particularly useful, for example, to administrate or visualize certain sub-areas of the database, e.g. Sales or Administration, independently or separately from the rest of the database. Use the Manage Subject Areas menu item.



Using the two dialog icons, new subjects can be defined by entering a name and checking those tables to be included; or existing subjects altered or deleted. Since IBExpert version 2004.12.12.1 it is possible to drag 'n' drop objects from the DB Explorer (Diagrams page) to the subject areas to include them as members of this area. It is also possible to drag objects from the list of objects in the Subject Areas Manager.

Several subject areas can be opened and administrated simultaneously; switch from subject to subject by clicking on the window buttons underneath the main editing area.

These subject areas are stored with the main subject area when the diagram is saved to file.

8.23.7 Manage Subject Layers

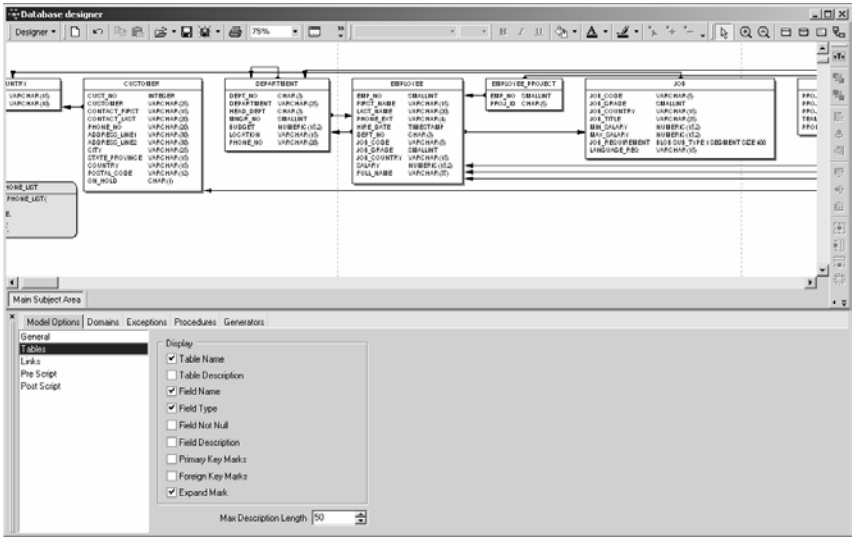
This filter option allow certain specified tables and their relationships to be viewed. Simply click the New Layer icon, name the layer, and check those objects to be included.

In order to view everything again, it is necessary to reopen the Manage Layers dialog, and click the icon Show All.

The diagram created may be saved to file or exported using the respective Designer menu item or Save icon.

8.23.8 Model Options

The Model Options menu item opens a new window in the lower half of the Database Designer dialog. Here the following visual display and script options may be selected:



When a table or view is double-clicked in the main editing area, an additional window appears automatically in the model options dialog.

General: Since version 2004.6.17 it is possible to specify the font character set for model objects. Simply click *General* on the left-hand list, and specify the character set using the pull-down list.

Table: Options to display the following: *Table Name* and *Description*, *Field Name*, *Type*, *Not Null* and *Description*, *Primary Key* and *Foreign Key Marks* and *Expand Mark*. It is even possible to specify the maximal description length.

Links: *Display Link Names* (i.e. display FK relationships) and *Automatically Trace Links* (displays the links as horizontal/vertical lines with 90° corners).

The *pre-* and *postscript* options were added in IBExpert version 2003.12.18.1. This offers the possibility to define pre- and postscripts for your database model. The prescript will be inserted into the model script just after `CREATE DATABASE` or `CONNECT` statement. The postscript will be added to the end of the model script. There is also now an added option allowing you to define pre- and postscripts for each table separately.

By double-clicking on the line between two tables, the relationships are shown in detail. The name and automatic tracing of links, are options already mentioned, included in the Model Options menu item.

The Model Options window may be closed by clicking the small black **X** in the top left-hand corner.

Domains

The Model Options also included a Domains page with various insert, alter and delete options, similar to the Domain Editor in the DB Explorer.

Exceptions

The Model Options also included an Exceptions page with various insert, alter and delete options, similar to the Exception Editor in the DB Explorer. The support of exceptions and stored procedures has been included since IBExpert version 2.5.0.6.1.

Procedures

The Model Options also included a Procedures page, similar to the Procedure Editor in the DB Explorer. The support of stored procedures has been included since IBExpert version 2.5.0.6.1.

It is possible to insert a new procedure or delete a selected procedure, using the relevant icons. Procedures can be selected from the pull-down list to the right of these icons.

The code can be altered as wished; the editing page offering all those features included in all IBExpert Edit pages (such as Code Completion, comprehensive right-click menu (SQL Editor Menu) etc).

Generators

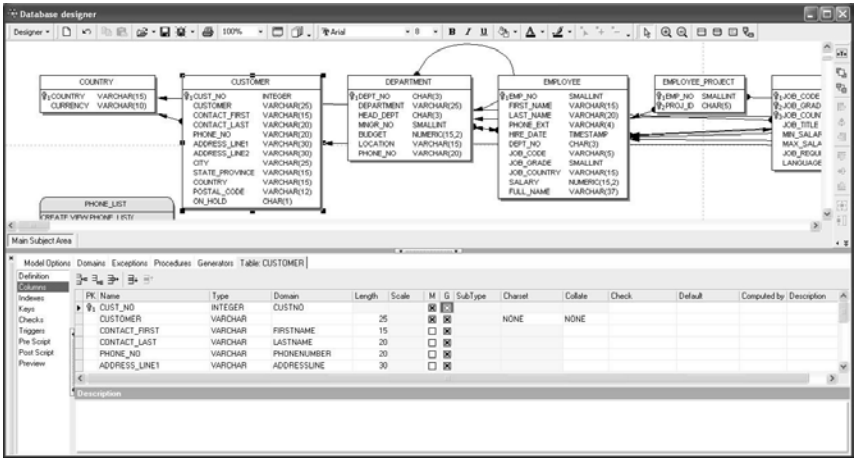
The Model Options also include a Generators page with various insert, alter and delete options, similar to the Generator Editor in the DB Explorer. The support of generators has been included since IBExpert version 2004.1.22.1.

Selected Table / Selected View

Table <selected table>: The options allow columns, indices, keys, checks and triggers to be added, amended or deleted. This version of the IBExpert Table Editor can be

used to create a new table or view, or alter an existing selected table. For details please refer to Create Table and Table Editor.

View <selected view>: A new view can only be created in the Database Designer using SQL. Alternatively create a new view in the DB Explorer, and update an existing diagram using Reverse Engineer... For further information regarding view creation in the IBExpert DB Explorer, please refer to New View.



The Definitions page displays the table or view name, allows a description to be displayed/entered and the *Generate* check option allows the selected table or view to be updated in the diagram.

The Selected Table options: *Columns*, *Indexes*, *Keys*, *Checks, Triggers* and *Preview*, and the Selected View options: *SQL*, *Triggers* and *Preview*, are based on those pages found in the Table Editor and View Editor in the DB Explorer. There is however a number of abbreviations included in these frames, which are not included in the DB Explorer editors. These have the following meaning:

- G = generate, i.e. include into the result script.
- U = unique (for indices)
- A = active (for triggers)
- M = mandatory (for columns, i.e. NOT NULL)

Since version 2003.12.18.1 there is also the possibility to define pre- and postscripts for each table separately. The prescript will be inserted into the model script just after the `CREATE DATABASE` or `CONNECT` statement. The postscript will be added to the end of the model script. You can also define pre- and postscripts for each table separately.

Since version 2.5.0.61 IBExpert has increased the flexibility with regard to customizing table layout. It is possible to toggle on/off displaying of field name, field type, NOT NULL flag, field description and foreign key mark in any combination. It is also possible to display the table description instead of, or together with the table name.

In IBExpert version 2003.11.6.1 the View Editor and Note Editor were redesigned. They are now no longer modal.

Comment Box

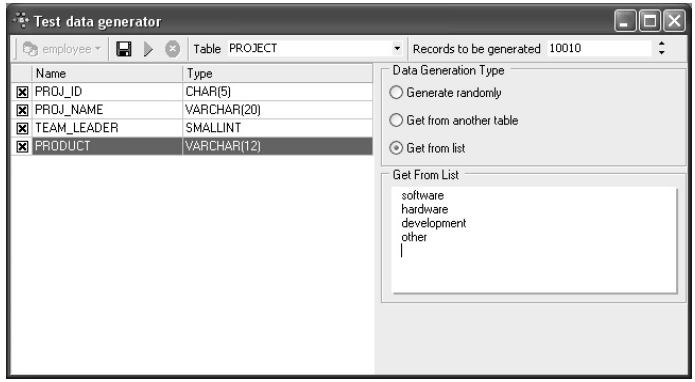
When a Comment Box is inserted into the main diagram, double-clicking upon this box produces a new Comment Box page in the Model Options dialog. This can be used to insert, alter or delete a comment text as wished.

In IBExpert version 2003.11.6.1 the View Editor and Note Editor were redesigned. They are now no longer modal.

8.24 Test Data Generator

The IBExpert Test Data Generator can be found in the Tools menu.

(This feature is unfortunately not included in the Personal Edition.)



A database connection must already exist. Select the database for which test data is to be generated, if more than one database is connected. To generate data for a specific table, select the table, then select the number of data sets to be generated. Over 100,000 data sets are not a problem for IBExpert here, even when working locally, although it may take a little time. Click on the individual fields and specify the contents on the right. It is possible to specify the following:

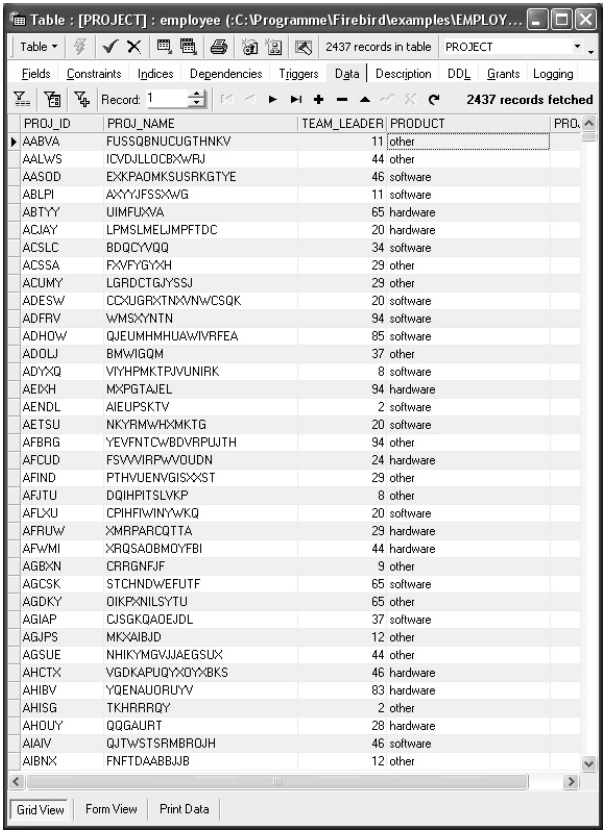
Data Generation Type: options here include:

- **Generate randomly:** User-defined constraints include the following:
  - *Integer*: the minimum and maximum value.
  - *Float*: check option *Fixed Float Number*, and user specification of number of digits and level of precision.
  - *String*: the minimum and maximum length; the range of characters within the character set which may be used for the data content.
  - *Date*: the minimum and maximum date, and a check option, whether a time slice should also be included.

- **Get from another table:** Specify table, field and number of records. This is a useful way of generating test data for a foreign key field.
- **Get from a list:** A list can be typed or pasted in the panel.
- **Autoincrement:** This option is of course only offered for integral fields, and enables the developer to specify an initial value, and the interval (step).

Finally execute (green > icon or [F9]), and watch the counter generate the test data!

The data can finally be viewed in the Table Editor on the Data page:



PROJ_ID	PROJ_NAME	TEAM_LEADER	PRODUCT	PRO
AABVA	FUSSQBNUCUGTHNKV	11	other	
AALWS	ICVDJLLOCBXWRJ	44	other	
AASOD	EXKPAOMKSUSRKGTYE	46	software	
ABLPI	AXYYJFSSXWG	11	software	
ABTYY	UIMFUXVA	65	hardware	
ACJAY	LPMSLMELJMPFTDC	20	hardware	
ACSLC	BDQCYVQQ	34	software	
ACSSA	FXVFYGYXH	29	other	
ACUMY	LGRDCTGJYSSJ	29	other	
ADESW	CCXJGRXTNXVNWCSQK	20	software	
ADFRV	WMSXYNTN	94	software	
ADHOW	QJEUMHMHUAWIVRFEA	85	software	
ADOLJ	BMWIGQM	37	other	
ADYXQ	VIYHPMKTPJVUNIRK	8	software	
AEDXH	MXPGTAJEL	94	hardware	
AENDL	AIEUPSKTV	2	software	
AETSU	NKYRMWHXMKTG	20	software	
AFBRG	YEVFNTCWBDVRPUJTH	94	other	
AFCUD	FSVWIRPWVOUDN	24	hardware	
AFIND	PTHVUENVGISXXST	29	other	
AFJTU	DQIHPITSLVKP	8	other	
AFLXU	CPIHFIWINYWKQ	20	software	
AFRUW	XMRPARCQTTA	29	hardware	
AFWMI	XRQSAOBMOYFBI	44	hardware	
AGBXN	CRRGNFJF	9	other	
AGCSK	STCHNDWEFUTF	65	software	
AGDKY	OIKPXNILSYTU	65	other	
AGIAP	CJSGKQAOEJDL	37	software	
AGJPS	MKXAIBJD	12	other	
AGSUE	NHIKYMGVJJAEGSUX	44	other	
AHCTX	VGDKAPUQYXOYXBKS	46	hardware	
AHIBV	YQENAUORUYV	83	hardware	
AHISG	TKHRRRQY	2	other	
AHOUY	QQGAURT	28	hardware	
AIAIV	QJTWSTSRMBRQJH	46	software	
AIBNX	FNFTDAABBJJB	12	other	

8.25 IBExpert Command–Line Tools

IBExpert offers the following command-line tools:

- IBEBlock,
- IBECompare,
- IBEExtract,
- IBEScript.

These cover the majority of the options offered by the InterBase command-line utilities and much more.

In order to distribute any of the IBExpert modules (ibexpert.exe, ibescript.dll, ibeextract.exe and ibecompare.exe) together with your application, you need either an:

- IBExpert Site License, if the distribution is located only on computers within your own company, or an
- IBExpert VAR License, if the distribution is located on any computer outside your company.

If you are already an IBExpert customer, you can upgrade to a Site or VAR License and purchase the 24 month Extension product. Please refer to www.ibexpert.com Purchase Area for details.

8.25.1 IBEBLOCK (EXECUTE IBEBLOCK)

IBExpert version 2004.9.12.1 introduced an important, new and powerful feature - `EXECUTE IBEBLOCK`.

What is IBEBLOCK?

It is a set of DDL, DML and other statements that are executed on the server and on the client side, and which include some specific constructions applicable only in IBExpert or IBEScript (excluding the free versions of these products), independent of the database server version.

With `EXECUTE IBEBLOCK` you will be able to:

- Work with different connections within the single `IBEBLOCK` at the same time.
- Move (copy) data from one database to another.
- Join tables from different databases.
- Compare data from different databases and synchronize them.
- Populate a table with test data using random values or values from other tables or even from other databases.
- ... and much more.

The syntax of `IBEBLOCK` is similar to that of stored procedures but there are many important extensions. For example:

- You can use `EXECUTE STATEMENT` with any server, including InterBase 5.x, 6.x, 7.x.
- You can use one-dimensional arrays (lists) of untyped variables and access them by index.
- It isn't necessary to declare variables before using them.
- You can use data sets (temporary memory tables) to store data.
- Since IBExpert version 2005.02.12.1 there is added support for `ROW_COUNT` and `ROWS_AFFECTED` variables.
- Since version 2005.02.12.1 Code Insight also supports IBEBlock constants and functions.
- ... and much more.

You can debug `IBEBLOCK`s in the same way as stored procedures and triggers.

IBEScript supports EXECUTE IBEBLOCK too.

This section describes the following topics:

- Procedural extensions of IBEBlock
- IBEBlock functions
- Examples of usage of IBEBlock

As this important feature is still new to IBExpert, some areas are still incomplete or in work. Check regularly for the latest revisions by using the What's New function in the online documentation: http://www.ibexpert.info/documentation/whatsnew.html

Procedural extensions of IBEBlock

The following procedural extensions are available in IBEBlock:

- CREATE CONNECTION
- USE *connection*
- CLOSE CONNECTION
- CREATE DATABASE
- DROP DATABASE
- FOR ... DO loops
- SELECT ... AS DATASET
- SELECT ... EXPORT AS ...
- CLOSE DATASET
- EXECUTE IBEBLOCK
- EXECUTE STATEMENT
- INSERT INTO *connection.table*
- COMMIT
- ROLLBACK
- FOR EXECUTE STATEMENT ... DO
- Default calues and comments for parameters and variables

CREATE CONNECTION

Creates a named connection to a database.

Syntax

```
CREATE CONNECTION connection DBNAME 'filespec'
   USER 'username' PASSWORD 'password'
   [CLIENTLIB 'libfile']
   [NAMES charset]
   [SQL_DIALECT dialect]
   [ROLE rolename]
```

Argument	*Description*
connection	Connection name.
DBNAME *'filespec'*	Database file name; can include path specification and node.

USER *'user-name'*	String that specifies a user name for use when attaching to the database. The server checks the user name against the security database. User names are case insensitive on the server.
PASSWORD *'password'*	String, up to 8 characters in size, that specifies password for use when attaching to the database. The server checks the user name and password against the security database. Case sensitivity is retained for the comparison.
CLIENTLIB *'libfile'*	Client library file name; default: gds32.dll.
NAMES *charset*	Name of a character set that identifies the active character set for a given connection; default: NONE.
SQL_DIALECT *dialect*	The SQL Dialect for database access, either 1, 2, or 3.
ROLE *role-name*	String, up to 31 characters in size, which specifies the role that the user adopts on connection to the database. The user must have previously been granted membership in the role to gain the privileges of that role. Regardless of role memberships granted, the user has the privileges of a role at connect time only if a ROLE clause is specified in the connection. The user cannot adopt more than one role per connection, and cannot switch roles except by reconnecting.

Description

Example

```
execute IBEBlock
as
begin
  CREATE CONNECTION Con1 DBNAME 'localhost:c:\mydata\mydb.gdb'
  USER 'SYSDBA' PASSWORD 'masterkey'
  CLIENTLIB 'C:\Program Files\Firebird\Bin\fbclient.dll'
  SQL_DIALECT 3 NAMES WIN1251 ROLE ADMIN;

  USE Con1;

  ...

  CLOSE CONNECTION Con1;
end
```

USE connection

Makes an existing connection the active connection.

Syntax

```
USE connection;
```

Argument***Description******connection*****Name of an existing connection created with the `CREATE CONNECTION` statement.**

Description

Example

```
execute IBEBlock
as
begin
  CREATE CONNECTION Con1 DBNAME 'localhost:c:\mydata\mydb.gdb'
  USER 'SYSDBA' PASSWORD 'masterkey'
  CLIENTLIB 'C:\Program Files\Firebird\Bin\fbclient.dll'
  SQL_DIALECT 3 NAMES WIN1251 ROLE ADMIN;

  USE Con1;

  ...

  CLOSE CONNECTION Con1;
end
```

CLOSE CONNECTION

Closes an existing connection.

Syntax

```
CLOSE CONNECTION connection;
```

Argument	*Description*
connection	Name of an existing connection opened with the `CREATE CONNECTION` statement.

Description

Example

```
execute IBEBlock
as
begin
  CREATE CONNECTION Con1 DBNAME 'localhost:c:\mydata\mydb.gdb'
  USER 'SYSDBA' PASSWORD 'masterkey'
  SQL_DIALECT 3 NAMES WIN1251;

  USE Con1;

  ...

  CLOSE CONNECTION Con1;
end
```

CREATE DATABASE

Syntax

```
CREATE DATABASE 'filespec' USER 'username' PASSWORD 'password'
   [CLIENTLIB 'libfile']
   [SQL_DIALECT dialect]
   [PAGE_SIZE int]
   [DEFAULT CHARACTER SET charset]
```

Argument	*Description*
'filespec'	A new database file specification; file naming conventions are platform-specific.
USER *'username'*	Checks the username against valid user name and password combinations in the security database on the server where the database will reside.
PASSWORD *'password'*	Checks the password against valid user name and password combinations in the security database on the server where the database will reside; can be up to 8 characters.
CLIENTLIB *'libfile'*	Client library file name; default: gds32.dll.
SQL_DIALECT *dialect*	The SQL Dialect for the new database, either 1, 2, or 3.
PAGE_SIZE *int*	Size, in bytes, for database pages; int can be 1024 (default), 2048, 4096, or 8192.
DEFAULT CHARACTER SET *charset*	Sets default character set for a database; charset is the name of a character set; if omitted, character set defaults to NONE.

Description

Example

```
execute IBEBlock
as
begin
  CREATE DATABASE 'localhost:c:\db2.fdb'
  USER 'SYSDBA' PASSWORD 'masterkey'
  PAGE_SIZE 4096 SQL_DIALECT 3
  DEFAULT CHARACTER SET WIN1251
  CLIENTLIB 'C:\Program Files\Firebird\bin\fbclient.dll';

  CREATE CONNECTION Con1 'localhost:c:\db2.fdb'
  USER 'SYSDBA' PASSWORD 'masterkey'
  CLIENTLIB 'C:\Program Files\Firebird\Bin\fbclient.dll'
  SQL_DIALECT 3 NAMES WIN1251;

  USE Con1;
```

```
  ...

  CLOSE CONNECTION Con1;
end
```

DROP DATABASE

Deletes specified database.

Syntax

```
DROP DATABASE 'filespec' USER 'username' PASSWORD 'password'
   [CLIENTLIB 'libfile'];
```

Argument	*Description*
'filespec'	A database file specification; file naming conventions are platform-specific.
USER *'username'*	Checks the username against valid user name and password combinations in the security database on the server where the database will reside.
PASSWORD *'password'*	Checks the password against valid user name and password combinations in the security database on the server where the database will reside; can be up to 8 characters.
CLIENTLIB *'libfile'*	Client library file name; default: `gds32.dll`.

Description

`DROP DATABASE` deletes specified database, including any associated secondary, shadow, and log files. Dropping a database deletes any data it contains.

A database can be dropped by its creator, the SYSDBA user, and any users with operating system root privileges.

Example

```
  execute ibeblock
  as
  begin
    drop database 'localhost/3060:c:\db1.fdb' user 'SYSDBA' password
'masterkey'
    clientlib 'C:\Program Files\Firebird\bin\fbclient.dll';
  end
```

DO loops

`FOR ... DO` loops were implemented in IBExpert version 2005.03.12. Examples of usage:

```
EXECUTE IBEBLOCK
RETURNS (I INTEGER)
AS
BEGIN
  FOR I = 0 TO 100 DO
    SUSPEND;
END
```

It is possible to use the CONTINUE statement within FOR loop to proceed to the next iteration of FOR:

```
EXECUTE IBEBLOCK
RETURNS (I INTEGER)
AS
BEGIN
  FOR I = 0 TO 100 DO
  BEGIN
    IF (I < 20) THEN
      CONTINUE; -- SUSPEND will not be executed
    SUSPEND;
  END
END
```


AS DATASET

Syntax

<*select_statement*> AS DATASET *dataset*;

Argument	*Description*
<*select_statement*>	Regular SELECT statement.
dataset	Name of the dataset.

Description

Example

```
execute ibeblock
returns (FieldName varchar(31), FieldType varchar(100))
as
begin
  select * from rdb$fields
  where (1 = 0)
  as dataset RdbFields;

  iCount = ibec_ds_FieldCount(RdbFields);
  i = 0;
  while (i < iCount) do
  begin
    FieldName = ibec_ds_FieldName(RdbFields, i);
```

```
    FieldType = ibec_ds_FieldTypeN(RdbFields, i);
    suspend;
    i = i + 1;
  end;

  close dataset RdbFields;
end
```

EXPORT AS ... INTO

`SELECT ... EXPORT AS ...` was implemented in IBExpert version 2005.03.12.

Examples of usage:

1.
```
SELECT * FROM RDB$FIELDS
EXPORT AS HTML INTO 'E:\TestExport.html'
OPTIONS 'ColorShema=MSMoney; FontFace=Verdana';
```

Possible ColorShemes are `BW`, `Classic`, `ColorFull`, `Gray`, `MSMoney`, `Murky`, `Olive`, `Plain`, `Simple`.

2.
```
SELECT * FROM RDB$FIELDS
EXPORT AS XLS INTO 'E:\TestExport.xls' OPTIONS '';
```

3.
```
SELECT * FROM RDB$FIELDS
EXPORT AS TXT INTO 'E:\TestExport.txt'
OPTIONS 'OmitCaptions';
```

4.
```
SELECT * FROM RDB$FIELDS
EXPORT AS CSV INTO 'E:\TestExport.txt'
OPTIONS 'OmitCaptions; Delimiter=";"';
```

5.
```
SELECT * FROM RDB$FIELDS
EXPORT AS XML INTO 'E:\TestExport.xml'
OPTIONS 'Encoding=windows-1251; MemoAsText; StringAsText';
```

CLOSE DATASET

Closes an existing dataset.

Syntax

```
CLOSE DATASET dataset;
```

Argument	*Description*
dataset	Name of an existing dataset created with `SELECT ... AS DATASET`

statement.

Description

Example

```
execute ibeblock
returns (FieldName varchar(31), FieldType varchar(100))
as
begin
  select * from rdb$fields
  where (1 = 0)
  as dataset RdbFields;

  iCount = ibec_ds_FieldCount(RdbFields);
  i = 0;
  while (i < iCount) do
  begin
    FieldName = ibec_ds_FieldName(RdbFields, i);
    FieldType = ibec_ds_FieldTypeN(RdbFields, i);
    suspend;
    i = i + 1;
  end;

  close dataset RdbFields;
end
```


EXECUTE IBEBLOCK

The `EXECUTE IBEBLOCK` statement was implemented in IBExpert verison 2005.03.12. Using this statement you can call other IBEBlocks from the main block.

Examples of usage:

1.

```
EXECUTE IBEBLOCK
AS
BEGIN
  ...
  MyFunc = 'EXECUTE IBEBLOCK (
              IntVal INTEGER)
            RETURNS (
              Square INTEGER)
            AS
            BEGIN
              Square = IntVal * IntVal;
            END';
  EXECUTE IBEBLOCK MyFunc (2) RETURNING_VALUES :Square;
  ...
END
```

2.

```
EXECUTE IBEBLOCK
AS
BEGIN
 ...
  MyFunc = ibec_LoadFromFile('C:\MyBlocks\Square.ibeblock');
  EXECUTE IBEBLOCK MyFunc (2) RETURNING_VALUES :Square;
  ...
END
```

EXECUTE STATEMENT

Executes specified SQL statement.

Syntax

```
EXECUTE STATEMENT 'statement'
   [INTO :var [, :var ...]]
   [VALUES :var];
```

Argument	*Description*
'statement'	Any valid DML or DDL statement except CREATE/DROP DATABASE. DML statements may contain parameters.
INTO *:var [, :var ...]*	Specifies a list of variables into which to retrieve values. Only singleton SELECT operators may be executed with this form of EXECUTE STATEMENT.
VALUES *:var*	Array of variants which values will be used to fill parameters if any exist in the statement.

Description

Example

```
  execute ibeblock
  returns (TableName varchar(31))
  as
  begin
    TableID = 0;
    Stmt = 'select rdb$relation_name from rdb$relations where
rdb$relation_id = :rel_id';
    while (TableID < 35) do
    begin
      execute statement :Stmt into :TableName values :TableId;
      suspend;
      TableID = TableID + 1;
    end
  end
```

INSERT INTO connection.table

Syntax

```
INSERT INTO connection.table [(col [, col ...])]
     {VALUES (<val> [, <val> ...]) | <select_expr>};
```

COMMIT

Makes a transaction's changes to the database permanent, and ends the transaction.

Syntax

```
COMMIT;
```

Description

Example

```
execute IBEBlock
as
begin
  ...

  EXECUTE STATEMENT 'create table mytable (id integer, data var-
char(50))';
  COMMIT;

  INSERT INTO MYTABLE (ID, DATA) VALUES (1, NULL);
  COMMIT;

  ...
end
```

ROLLBACK

Restores the database to its state prior to the start of the current transaction.

Syntax

```
ROLLBACK;
```

Description

ROLLBACK undoes changes made to a database by the current transaction, then ends the transaction.

Example

DO

Example

```
  execute ibeblock
  returns (TableName varchar(31))
  as
  begin
    TableID = 0;
    Stmt = 'select rdb$relation_name from rdb$relations where
rdb$relation_id = :rel_id';
    while (TableID < 35) do
    begin
      execute statement :Stmt into :TableName values :TableId;
      suspend;
      TableID = TableID + 1;
    end
  end
```

Default values and comments for parameters and variables

Default values and comments for input/output parameters and variables were implemented in IBExpert version 2005.03.12.

Example:

```
EXECUTE IBEBLOCK (
  CodeDir VARCHAR(1000) = 'C:\MyBlocks\' COMMENT 'Path to my IBEBlocks',
  SQLDialect INTEGER = 3 COMMENT 'Database SQL Dialect')
RETURNS (
  TotalTime DOUBLE PRECISION = 0 COMMENT 'Total time spent')
AS
DECLARE VARIABLE MyVar INTEGER = 0 COMMENT 'Just a comment'
BEGIN
 ...
END
```

- Comments for input parameters will be displayed in `Description` column of `Request Input Parameters` form.
- Comments for output variables will be used as column captions of the result dataset.
- Comments for local variables are ignored.

IBEBlock functions

There are following groups of IBEBlock functions:

- String-handling functions
- Mathematical functions
- Functions to work with files
- Dataset functions
- Miscellaneous functions

String–handling functions

The following string-handling functions are available in IBEBlock:

Function	*Description*
ibec_Copy	Returns a substring of a string
ibec_Length	Returns the number of characters in a string
ibec_Pos	Returns the index value of the first character in a specified substring that occurs in a given string
ibec_Trim	Trims leading and trailing spaces and control characters from a string

ibec_Copy

Returns a substring of a string.

Syntax

```
function ibec_Copy(S : string; Index, Count: Integer): string;
```

Description

`S` is an expression of a string. `Index` and `Count` are integer-type expressions. `ibec_Copy` returns a substring containing `Count` characters starting at `S[Index]`.

If `Index` is larger than the length of `S`, `ibec_Copy` returns an empty string.

If `Count` specifies more characters than are available, only the characters from `S[Index]` to the end of `S` are returned.

Example

```
execute IBEBlock
returns (proc_name varchar(31), proc_src varchar(100))
as
begin
  for
    select rdb$procedure_name, rdb$procedure_source
    from rdb$procedures
    order by rdb$procedure_name
    into :proc_name, :proc_src
  do
  begin
    proc_src = ibec_Copy(proc_src, 1, 100);
    suspend;
  end
end
```

ibec_Length

Returns the number of characters in a string.

Syntax

```
function ibec_Length(S : string): string;
```

Description

No additional description...

Example

```
execute IBEBlock
returns (iresult integer)
as
begin
  for select rdb$relation_name
      from rdb$relations
      into :sname
  do
  begin
    sname = ibec_Trim(sname);
    iresult = ibec_Length(sname);
    suspend;
  end
end
```

ibec_Pos

Returns the index value of the first character in a specified substring that occurs in a given string.

Syntax

```
function ibec_Pos(Substr: string; S : string): integer;
```

Description

No additional description...

Example

```
execute IBEBlock
returns (vcresult varchar(100))
as
begin
  for select rdb$relation_name
      from rdb$relations
      into :sname
  do
  begin
    sname = ibec_trim(sname);
    vcresult = '';
    if (ibec_Pos('RDB$', sname) = 1) then
      vcresult = sname || ' is a system table';
    else if (ibec_Pos('IBE$', sname) = 1) then
```

```
        vcresult = sname || ' is an IBExpert table';
      else
        vcresult = sname || ' is an user table';
      suspend;
    end
  end
```

ibec_Trim

Trims leading and trailing spaces and control characters from a string.

Syntax

```
function ibec_Trim(S : string): string;
```

Description

No additional description...

Example

```
execute IBEBlock
returns (proc_name varchar(31), proc_src varchar(100))
as
begin
  for
    select rdb$procedure_name, rdb$procedure_source
    from rdb$procedures
    order by rdb$procedure_name
    into :proc_name, :proc_src
  do
  begin
    proc_src = ibec_Trim(ibec_Copy(proc_src, 1, 100));
    suspend;
  end
end
```

Mathematical functions

The following mathematical functions are available in IBEBlock:

Function	*Description*
ibec_Div	Returns the value of x/y rounded in the direction of zero to the nearest integer
ibec_Mod	Returns the remainder obtained by dividing its operands

ibec_Div

The value of x div y is the value of x/y rounded in the direction of zero to the nearest integer.

Syntax

```
function ibec_div(Operand1, Operand2 : integer) : integer;
```

Description

No additional description...

Example

```
execute IBEBlock
returns (cout varchar(100))
as
begin
  i = 1;
  while (I < 50) do
  begin
    if ((i/2 - ibec_div(i, 2)) > 0) then
      cout = i || ' is odd number';
    else
      cout = i || ' is even number';
    suspend;
    i = i + 1;
  end
end
```


ibec_Mod

Returns the remainder obtained by dividing its operands.

Syntax

```
function ibec_mod(Operand1, Operand2 : integer) : integer;
```

Description

No additional decription...

Example

```
execute IBEBlock
returns (cout varchar(100))
as
begin
  i = 1;
  while (I < 50) do
  begin
    if (ibec_mod(i, 2) = 0) then
      cout = i || ' is even number';
    else
      cout = i || ' is odd number';
    suspend;
```

```
    i = i + 1;
  end
end
```

File functions

The following file-handling functions are available in IBEBlock:

Function	Description
ibec_DeleteFile	Erases the file from the disk
ibec_FileExists	Tests if a specified file exists
ibec_FileSize	Returns the size of the specified file
ibec_GetFiles	See online help
ibec_LoadFromFile	Loads file data into variable
ibec_SaveToFile	Saves value of variable into file

The following functions are intended for working with files in stream mode:

Function	Description
ibec_fs_CloseFile	Closes the file opened with the ibec_fs_OpenFile function
ibec_fs_Eof	Tests whether the file position is at the end of a file
ibec_fs_OpenFile	Opens a file for reading or writing
ibec_fs_Position	Returns the current offset into the stream for reading and writing
ibec_fs_ReadIn	Reads a line of text from a file
ibec_fs_Seek	Resets the current position of the file stream
ibec_fs_Size	Returns the length, in bytes, of the file stream
ibec_fs_Writeln	See online help
ibec_fs_WriteString	See online help

ibec_DeleteFile

Erases the file from the disk.

Syntax

```
function ibec_DeleteFile(FileName : string): boolean;
```

Description

The ibec_DeleteFile function erases the file named by FileName from the disk. If the file cannot be deleted or does not exist, the function returns False.

Example

```
execute IBEBlock
as
begin
  FileName = 'C:\mydata.txt';
  if (ibec_FileExists(FileName)) then
    ibec_DeleteFile(FileName);
end
```

ibec_FileExists

Tests if a specified file exists.

Syntax

```
function ibec_FileExists(FileName : string): boolean;
```

Description

ibec_FileExists returns True if the file specified by FileName exists. If the file does not exist, the function returns False.

Example

```
execute IBEBlock
as
begin
  FileName = 'C:\mydata.txt';
  if (ibec_FileExists(FileName)) then
    ibec_DeleteFile(FileName);
end
```

ibec_FileSize

Returns the size of the specified file.

Syntax

```
function ibec_FileSize(FileName : string): variant;
```

Description

The ibec_FileSize function returns the size in bytes of the file specified by FileName. If the file does not exist, the function returns NULL.

Example

```
  execute ibeblock
  returns (fname varchar(100), isize integer)
  as
  begin
    options = __gfFullName;
    files_count = ibec_getfiles(files_list, 'E:\Projects_5\', '*.*', op-
tions);
    if (files_count > 0) then
    begin
      i = 0;
      while (i < ibec_high(files_list)) do
      begin
        fname = files_list[i];
        isize = ibec_filesize(fname);
        suspend;
        i = i + 1;
      end
    end
  end
```

ibec_LoadFromFile

Loads file data into variable.

Syntax

```
  function ibec_LoadFromFile(FileName : string): string;
```

Description

Example

See Inserting file data into database.

ibec_SaveToFile

Saves value of variable into file.

Syntax

```
  function ibec_SaveToFile(FileName : string; Value : variant; Mode : in-
teger): variant;
```

Description

Example

ibec_fs_CloseFile

Closes the file opened with the `ibec_fs_OpenFile` function.

Syntax

```
function ibec_fs_CloseFile(FileHandle : variant): variant
```

Description

The `ibec_fs_CloseFile` function closes the file opened with the `ibec_fs_OpenFile` function.

This function always returns 0.

Example

```
execute  IBEBlock
as
begin
  FileName = 'C:\mydata.txt';
  FH = ibec_fs_OpenFile(FileName, __fmCreate);
  if (not FH is NULL) then
  begin
    ibec_fs_Writeln(FH, 'just a test');
    ibec_fs_CloseFile(FH);
  end
end
```


ibec_fs_Eof

Tests whether the file position is at the end of a file.

Syntax

```
function ibec_fs_Eof(FileHandle : variant): boolean;
```

Description

The `ibec_fs_Eof` function tests whether the file position is at the end of a file. `ibec_fs_Eof` returns `True` if the current file position is beyond the last character of the file or if the file is empty; otherwise, `ibec_fs_Eof` returns `False`.

Example

```
execute IBEBlock
returns (vcout varchar(1000))
as
begin
  FileName = 'C:\mydata.csv';
  FH = ibec_fs_OpenFile(FileName, __fmOpenRead);
  if (not FH is NULL) then
```

```
  begin
    while (not ibec_fs_Eof(FH)) do
    begin
      vcout = ibec_fs_Readln(FH);
      suspend;
    end
    ibec_fs_CloseFile(FH);
  end
end
```

ibec_fs_OpenFile

Opens a file for reading or writing.

Syntax

```
function ibec_fs_OpenFile(FileName : string; Mode : integer): variant;
```

Description

The `ibec_fs_OpenFile` function opens file specified by `FileName` for reading or writing.

The `Mode` parameter indicates how the file is to be opened. The `Mode` parameter consists of an open mode and a share mode or'ed together. The open mode must be one of the following values:

Value	*Meaning*
__fmCreate	Create a file with the given name. If a file with the given name exists, open the file in write mode.
__fmOpenRead	Open the file for reading only.
__fmOpenWrite	Open the file for writing only. Writing to the file completely replaces the current contents.
__fmOpenReadWrite	Open the file to modify the current contents rather than replace them.

The share mode must be one of the following values:

Value	*Meaning*
__fmShareCompat	Sharing is compatible with the way FCBs are opened.
__fmShareExclusive	Other applications can not open the file for any reason.
__fmShareDenyWrite	Other applications can open the file for reading but not for writing.
__fmShareDenyRead	Other applications can open the file for writing but not for reading.
__fmShareDenyNone	No attempt is made to prevent other applications from reading from or writing to the file.

If the file can not be opened, ibec_fs_OpenFile returns NULL. Otherwise it returns the Handle for just opened file.

To close the file opened with ibec_fs_OpenFile use ibec_fs_CloseFile function.

Example

```
execute  IBEBlock
as
begin
  FileName = 'C:\mydata.txt';
  FH = ibec_fs_OpenFile(FileName, __fmCreate);
  if (not FH is NULL) then
  begin
    ibec_fs_Writeln(FH, 'just a test');
    ibec_fs_CloseFile(FH);
  end
end
```

ibec_fs_Position

Returns the current offset into the stream for reading and writing.

Syntax

```
function ibec_fs_Position(FileHandle : variant) : integer;
```

Description

Use ibec_fs_Position to obtain the current position of the stream. This is the number of bytes from the beginning of the streamed data.

Example

```
execute IBEBlock
returns (vcout varchar(1000))
as
begin
  FileName = 'C:\mydata.csv';
  FH = ibec_fs_OpenFile(FileName, __fmOpenRead);
  if (not FH is NULL) then
  begin
    while (ibec_fs_Position(FH) < ibec_fs_Size(FH)) do
    begin
      vcout = ibec_fs_Readln(FH);
      suspend;
    end
    ibec_fs_CloseFile(FH);
  end
end
```

ibec_fs_Readln

Reads a line of text from a file.

Syntax

```
function ibec_fs_Readln(FileHandle : variant) : string;
```

Description

The `ibec_fs_Readln` function reads a line of text and then skips to the next line of the file.

Example

```
execute IBEBlock
returns (vcout varchar(1000))
as
begin
  FileName = 'C:\mydata.csv';
  FH = ibec_fs_OpenFile(FileName, __fmOpenRead);
  if (not FH is NULL) then
  begin
    while (not ibec_fs_Eof(FH)) do
    begin
      vcout = ibec_fs_Readln(FH);
      suspend;
    end
    ibec_fs_CloseFile(FH);
  end
end
```


ibec_fs_Seek

Resets the current position of the file stream.

Syntax

```
  function ibec_fs_Seek(FileHandle : variant; Offset: integer; Origin:
integer): integer;
```

Description

Use `ibec_fs_Seek` to move the current position within the file by the indicated offset. `ibec_fs_Seek` allows to read from or write to a particular location within the file.

The `Origin` parameter indicates how to interpret the `Offset` parameter. `Origin` should be one of the following values:

Value	*Meaning*
__soFromBeginning	`Offset` is from the beginning of the resource. `ibec_fs_Seek`

	moves to the position `Offset`. `Offset` must be >= 0.
__soFromCurrent	`Offset` is from the current position in the resource. `ibec_fs_Seek` moves to `Position + Offset`.
__soFromEnd	`Offset` is from the end of the resource. `Offset` must be <= 0 to indicate a number of bytes before the end of the file.

`ibec_fs_Seek` returns the new current position in the file.

Example

ibec_fs_Size

Returns the length, in bytes, of the file stream.

Syntax

```
function ibec_fs_Size(FileHandle : variant) : integer;
```

Description

The `ibec_fs_Size` returns the length, in bytes, of the file identified by the `FileHandle`.

Example

```
execute IBEBlock
returns (vcout varchar(1000))
as
begin
  FileName = 'C:\mydata.csv';
  FH = ibec_fs_OpenFile(FileName, __fmOpenRead);
  if (not FH is NULL) then
  begin
    while (ibec_fs_Position(FH) < ibec_fs_Size(FH)) do
    begin
      vcout = ibec_fs_Readln(FH);
      suspend;
    end
    ibec_fs_CloseFile(FH);
  end
end
```

Dataset functions

The following dataset-handling functions are available in IBEBlock:

Function	*Description*
ibec_ds_Append	Adds a new, empty record to the end of the dataset

ibec_ds_Cancel	Cancels modifications to the active record if those changes are not yet posted
ibec_ds_Delete	Deletes the active record and positions the cursor on the next record
ibec_ds_Edit	Enables editing of data in the dataset
ibec_ds_Eof	Indicates whether or not a cursor is positioned at the last record in a dataset
ibec_ds_Bof	Indicates whether or not a cursor is positioned at the first record in a dataset
ibec_ds_FieldCount	Returns the number of fields associated with the dataset
ibec_ds_FieldName	Returns the name of specified field
ibec_ds_FieldType	See online help
ibec_ds_FieldTypeN	Returns the native type of specified field
ibec_ds_First	Positions the cursor on the first record in the dataset
ibec_ds_GetField	Returns value of specified field
ibec_ds_Insert	See online help
ibec_ds_Last	Positions the cursor on the last record in the dataset
ibec_ds_Next	Positions the cursor on the next record in the dataset
ibec_ds_Post	See online help
ibec_ds_Prior	Positions the cursor on the previous record in the dataset
ibec_ds_SetField	See online help

ibec_ds_Append

Adds a new, empty record to the end of the dataset.

Syntax

```
function ibec_ds_Append(Dataset : variant) : variant;
```

Description

Call `ibec_ds_Append` to:

- Open a new, empty record at the end of the dataset.
- Set the active record to the new record.
- After a call to `ibec_ds_Append`, you can enter data in the fields of the record, and can then post those changes to the dataset using `ibec_ds_Post`.

Example

ibec_ds_Cancel

Cancels modifications to the active record if those changes are not yet posted.

Syntax

```
function ibec_ds_Cancel(Dataset : variant) : variant;
```

Description

Call `ibec_ds_Cancel` to undo modifications made to one or more fields belonging to the active record. As long as those changes are not already posted to the dataset, `ibec_ds_Cancel` returns the record to its previous state, and sets the dataset state to `__dsBrowse`.

Example

ibec_ds_Delete

Deletes the active record and positions the cursor on the next record.

Syntax

```
function ibec_ds_Delete(Dataset : variant) : variant;
```

Description

Call `ibec_ds_Delete` to remove the active record from the database. If the dataset is inactive, `ibec_ds_Delete` raises an exception. Otherwise `ibec_ds_Delete`:

- Verifies that the dataset is not empty (and raises an exception if it is).
- Deletes the record.
- Frees the buffers allocated for the record.
- Puts the dataset into `__dsBrowse` mode :
- Resynchronizes the dataset to position the cursor on the next undeleted record.

Example

ibec_ds_Edit

Enables editing of data in the dataset.

Syntax

```
procedure ibec_ds_Edit(Dataset : variant) : variant;
```

Description

Call `ibec_ds_Edit` to permit editing of the active record in a dataset. `ibec_ds_Edit` determines the current state of the dataset. If the dataset is empty, `ibec_ds_Edit` calls `ibec_ds_Insert`.

Example

ibec_ds_Eof

Indicates whether or not a cursor is positioned at the last record in a dataset.

Syntax

```
function ibec_ds_Eof(Dataset : variant) : boolean;
```

Description

Call `ibec_ds_Eof` to determine if the cursor is positioned at the last record in a dataset. If `ibec_ds_Eof` returns `True`, the cursor is unequivocally on the last row in the dataset. Otherwise this function returns `False`.

Example

```
execute ibeblock
as
begin
  select * from RDB$FIELDS as dataset MyDataset;

  while (not ibec_ds_Eof(MyDataset)) do
  begin
    ...
    ibec_ds_Next(MyDataset);
  end

  ...

  close dataset MyDataset;
end
```

ibec_ds_Bof

Indicates whether or not a cursor is positioned at the first record in a dataset.

Syntax

```
function ibec_ds_Bof(Dataset : variant) : boolean;
```

Description

Call `ibec_ds_Bof` to determine if the cursor is positioned at the first record in a dataset. If `ibec_ds_Bof` returns `True`, the cursor is unequivocally on the first row in the dataset. Otherwise this function returns `False`.

Example

```
execute ibeblock
as
```

```
begin
  select * from RDB$FIELDS as dataset MyDataset;

  ibec_ds_Last(MyDataset);
  while (not ibec_ds_Bof(MyDataset)) do
  begin
    ...
    ibec_ds_Prior(MyDataset);
  end

  ...

  close dataset MyDataset;
end
```

ibec_ds_FieldCount

Returns the number of fields associated with the dataset.

Syntax

```
function ibec_ds_FieldCount(Dataset : variant) : integer;
```

Description

Call `ibec_ds_FieldCount` to determine the number of fields associated with the `Dataset`.

Example

ibec_ds_FieldName

Returns the name of specified field.

Syntax

```
function ibec_ds_FieldName(Dataset : variant; FieldIndex : integer) :
variant;
```

Description

Example

```
execute ibeblock
returns (FieldName varchar(31), FieldType varchar(100))
as
begin
  select * from rdb$fields
  where (1 = 0)
  as dataset RdbFields;

  iCount = ibec_ds_FieldCount(RdbFields);
```

```
  i = 0;
  while (i < iCount) do
  begin
    FieldName = ibec_ds_FieldName(RdbFields, i);
    FieldType = ibec_ds_FieldTypeN(RdbFields, i);
    suspend;
    i = i + 1;
  end;

  close dataset RdbFields;
end
```

ibec_ds_FieldTypeN

Returns the native type of specified field.

Syntax

```
  function ibec_ds_FieldTypeN(Dataset : variant; Field : variant) : vari-
ant;
```

Description

Example

```
  execute ibeblock
  returns (FieldName varchar(31), FieldType varchar(100))
  as
  begin
    select * from rdb$fields
    where (1 = 0)
    as dataset RdbFields;

    iCount = ibec_ds_FieldCount(RdbFields);
    i = 0;
    while (i < iCount) do
    begin
      FieldName = ibec_ds_FieldName(RdbFields, i);
      FieldType = ibec_ds_FieldTypeN(RdbFields, i);
      suspend;
      i = i + 1;
    end;

    close dataset RdbFields;
  end
```

ibec_ds_First

Positions the cursor on the first record in the dataset.

Syntax

```
function ibec_ds_First(Dataset : variant) : variant;
```

Description

Call `ibec_ds_First` to position the cursor on the first record in the dataset and make it the active record.

Example

ibec_ds_GetField

Returns value of specified field.

Syntax

```
function ibec_ds_GetField(Dataset : variant; Field : variant) : vari-
ant;
```

Description

Example

ibec_ds_Last

Positions the cursor on the last record in the dataset.

Syntax

```
function ibec_ds_Last(Dataset : variant) : variant;
```

Description

Call `ibec_ds_Last` to position the cursor on the last record in the dataset and make it the active record.

Example

```
execute ibeblock
as
begin
  select * from RDB$FIELDS as dataset MyDataset;

  ibec_ds_Last(MyDataset);
  while (not ibec_ds_Bof(MyDataset)) do
  begin
    ...
    ibec_ds_Prior(MyDataset);
  end

  ...
```

```
  close dataset MyDataset;
end
```

ibec_ds_Next

Positions the cursor on the next record in the dataset.

Syntax

```
function ibec_ds_Next(Dataset : variant) : variant;
```

Description

Call `ibec_ds_Next` to position the cursor on the next record in the dataset and make it the active record.

Example

```
execute ibeblock
as
begin
  select * from RDB$FIELDS as dataset MyDataset;

  while (not ibec_ds_Eof(MyDataset)) do
  begin
    ...
    ibec_ds_Next(MyDataset);
  end

  ...

  close dataset MyDataset;
end
```

ibec_ds_Prior

Positions the cursor on the previous record in the dataset.

Syntax

```
function ibec_ds_Prior(Dataset : variant) : variant;
```

Description

Call `ibec_ds_Prior` to position the cursor on the previous record in the dataset and make it the active record.

Example

```
execute ibeblock
as
```

```
begin
  select * from RDB$FIELDS as dataset MyDataset;

  ibec_ds_Last(MyDataset);
  while (not ibec_ds_Bof(MyDataset)) do
  begin
    ...
    ibec_ds_Prior(MyDataset);
  end

  ...

  close dataset MyDataset;
end
```

Miscellaneous functions

The following miscellaneous functions are available in IBEBlock:

Function	*Description*
ibec_BuildCube	Builds an OLAP cube using a specified SELECT statement
ibec_Chr	Returns the character for a specified ASCII value
ibec_CmpRecords	Compares two arrays of variants (records)
ibec_CmpVals	Compares two values
ibec_CreateModelScript	Creates an SQL script from specified Database Model file
ibec_FormatIdent	See online help
ibec_GetTickCount	Retrieves the number of milliseconds that have elapsed since Windows was started
ibec_High	Returns the highest value within the range of the index type of the array
ibec_IIF	Tests a condition and returns Value1 if the Condition is TRUE and Value2 if the Condition is FALSE
ibec_IntToHex	Returns the hex representation of an integer
ibec_Ord	Returns the ordinal value of the specified character
ibec_ParseCSVLine	See online help
ibec_Progress	Displays a progress message
ibec_Random	Generates random numbers within a specified range
ibec_Random2	Generates random numbers within a specified range
ibec_RandomChar	Generates random char within a specified range
ibec_RandomString	Returns a random string
ibec_RandomVal	See online help
ibec_SetLength	Sets the length of a dynamic-array variable

ibec_ShiftRecord	See online help

ibec_BuildCube

Builds an OLAP cube using a specified `SELECT` statement.

Syntax

```
function ibec_BuildCube(FileName : string;
                        SelectSQL : string;
                        Dimensions : Array of variants;
                        Measures : Array of variants;
                        Params : Array of variants): integer;
```

Description

Example

```
execute ibeblock
as
begin
  SelectSQL = 'select rf.rdb$relation_name, f.rdb$field_type,
f.rdb$field_length,

f.rdb$field_precision
                from rdb$relation_fields rf, rdb$fields f
                where rf.rdb$field_source = f.rdb$field_name';

  vDimensions[0] = 'FieldName=RDB$RELATION_NAME; Alias="Table Name"';
  vDimensions[1] = 'FieldName=RDB$FIELD_TYPE; Alias="Field Type';

  vMeasures[0] = 'FieldName=RDB$FIELD_TYPE; Alias="Field Count"; Calc-
Type=ctCount; Format=0';
  vMeasures[1] = 'FieldName=RDB$FIELD_LENGTH; Alias="Total Length";
CalcType=ctSum; Format=0';
  vMeasures[2] = 'FieldName=RDB$FIELD_PRECISION; Alias="Avg Precision";
CalcType=ctAverage';

  -- Build and save cube in binary format
  ibec_BuildCube('C:\test_cub.cub', SelectSQL, vDimensions, vMeasures,
null);

  -- Build and save cube in XML format
  ibec_BuildCube('C:\test_cub.xml', SelectSQL, vDimensions, vMeasures,
null);
end
```

ibec_Chr

Returns the character for a specified ASCII value.

Syntax

```
function ibec_Chr(X : integer): string;
```

Description

ibec_Chr returns the character with the ordinal value (ASCII value) of the byte-type expression, X.

Example

```
execute IBEBlock
returns (cout varchar(1))
as
begin
  i = 0;
  while (i < 256) do
  begin
    cout = ibec_Chr(i);
    i = i + 1;
    suspend;
  end
end
```


ibec_CmpRecords

Compares two arrays of variants (records).

Syntax

```
function ibec_CmpRecords(Record1, Record2 : array of variants): vari-
ant;
```

Description

Example

```
execute ibeblock
returns (iresult integer)
as
begin
  Val1[0] = 1; Val1[1] = 'ABC'; Val1[2] = 25.67;
  Val2[0] = 1; Val2[1] = 'ABC'; Val2[2] = 25.67;
  iresult = ibec_CmpRecords(Val1, Val2); /* iresult = 0 */
  suspend;

  Val2[2] = 15.43;
  iresult = ibec_CmpRecords(Val1, Val2); /* iresult = 2 */
```

```
  suspend;

  Val2[3] = 0;
  iresult = ibec_CmpRecords(Val1, Val2); /* iresult = NULL */
  suspend;
end
```

ibec_CmpVals

Compares two values.

Syntax

```
function ibec_CmpVals(Value1, Value2 : variant): variant;
```

Description

The `ibec_CmpVals` compares `Value1` and `Value2` and returns if they are equal.

If `Value1` is greater than `Value2`, `ibec_CmpVals` returns `1`.

If `Value1` is less than `Value2`, `ibec_CmpVals` returns `-1`.

If it is impossible to compare values the function returns `NULL`.

Example

```
execute IBEBlock
returns (iresult integer)
as
begin
  iresult = ibec_CmpVals(25, '25');
  suspend; /* Values are equal, iresult = 0 */

  iresult = ibec_CmpVals('25', 40);
  suspend; /* 25 is less then 40, iresult = -1 */

  iresult = ibec_CmpVals('ABC', 'abc');
  suspend; /* 'ABC' is less then 'abc', iresult = -1 */

  iresult = ibec_CmpVals(NULL, '25');
  suspend; /* NULL is less than any other value, iresult = -1 */

  iresult = ibec_CmpVals('25', NULL);
  suspend; /* Any value is greater than NULL, iresult = 1 */

  iresult = ibec_CmpVals(NULL, NULL);
  suspend; /* NULL is equal to NULL!!!, iresult = 0 */

  iresult = ibec_CmpVals('ABC', 25);
  suspend; /* Impossible to compare, iresult = NULL */
```

```
  iresult = ibec_CmpVals('24.56', 24.56);
  suspend; /* Values are equal, iresult = 0 */
end
```

ibec_CreateModelScript

Creates an SQL script from specified Database Model file.

Syntax

```
  function ibec_CreateModelScript(ModelFileName : string; ScriptFileName
: string; Options : cardinal): integer;
```

Description

Example

```
  execute ibeblock
  as
  begin
    ibec_create_model_script('C:\npfe_1.grc', 'C:\npfe_1.sql',
__msoDontQuoteIdents + __msoIncludeDescriptions);
  end
```

ibec_GetTickCount

Retrieves the number of milliseconds that have elapsed since Windows was started.

Syntax

```
  function ibec_GetTickCount : integer;
```

Description

No additional description...

Example

```
  execute IBEBlock
  returns (cout varchar(100))
  as
  begin
    Time1 = ibec_GetTickCount();

    select * from rdb$fields as dataset ds;
    close dataset ds;

    Time2 = ibec_GetTickCount();
    cout = 'Time elapsed: ' || ((Time2 - Time1) / 1000) || ' seconds';
    suspend;
  end
```

ibec_High

Returns the highest value within the range of the index type of the array.

Syntax

```
function ibec_High(AArray : array of variants): integer;
```

Description

No additional description yet...

Example

```
execute IBEBlock
returns (iresult integer)
as
begin
  vals = 0;
  iresult = ibec_High(vals);
  suspend; /* iresult = 0 */

  vals[1] = 12;
  iresult = ibec_High(vals);
  suspend; /* iresult = 1 */

  vals[10] = 'ibexpert';
  iresult = ibec_High(vals);
  suspend; /* iresult = 10 */

  ibec_SetLength(vals, 5);
  iresult = ibec_High(vals);
  suspend; /* iresult = 4 */

  ibec_SetLength(vals, 500);
  iresult = ibec_High(vals);
  suspend; /* iresult = 499 */

  ibec_SetLength(vals, 0);
  iresult = ibec_High(vals);
  suspend; /* iresult = 0 */
end
```

ibec_IIF

Tests a condition and returns `Value1` if the `Condition` is `TRUE` and `Value2` if the `Condition` is `FALSE`.

Syntax

```
function ibec_IIF(Condition : boolean; Value1, Value2 : variant): variant;
```

Description

Tests a condition and returns `Value1` if the `Condition` is `TRUE` and `Value2` if the `Condition` is `FALSE`.

Example

```
execute IBEBlock
returns (cout varchar(100))
as
begin
  i = 1;
  while (I < 50) do
  begin
    cout = ibec_IIF((ibec_mod(i, 2) = 0), i || ' is even number', i ||
' is odd number');
    suspend;
    i = i + 1;
  end
end
```

ibec_IntToHex

Returns the hex representation of an integer.

Syntax

```
function ibec_IntToHex(Value: Integer; Digits: Integer): string;
```

Description

`ibec_IntToHex` converts a number into a string containing the number's hexadecimal (base 16) representation. `Value` is the number to convert. `Digits` indicates the minimum number of hexadecimal digits to return.

Example

```
execute ibeblock
returns (iint integer, shex varchar(5))
as
begin
  iint = 0;
  while (iint < 1000) do
  begin
    shex = '$' || ibec_IntToHex(iint, 4);
    iint = iint + 1;
    suspend;
  end
end
```

ibec_Ord

Returns the ordinal value of the specified character.

Syntax

```
function ibec_Ord(Chr : char): integer;
```

Description

The `ibec_Ord` function returns the ordinal value of the specified character. If `Chr` is an empty string or NULL, then result is .

Example

```
execute IBEBlock
returns (cout varchar(1))
as
begin
  i = 0;
  while (i < 256) do
  begin
    cout = ibec_Chr(i);
    i = i + 1;
    suspend;
  end
end
```


ibec_ParseCSVLine

Syntax

```
function ibec_fs_ParseCSVLine(DestValues : array of variants; CSVLine :
string; QuoteChar : char; Delimiter : string; Options : cardinal): inte-
ger;
```

ibec_Progress

Displays a progress message.

Syntax

```
function ibec_Progress(Mes : string): string;
```

Description

Call `ibec_Progress` function to display a message. The `Msg` parameter is the message string that appears in the upper status panel of the SQL Editor or Script Editor. If you're executing an IBEBlock using the IBEScript tool the message will appear on the screen and will be included into log file

Example

```
  execute IBEBlock
  returns (table_name varchar(31), irecords integer)
  as
  begin
    for select rdb$relation_name
        from rdb$relations
        order by rdb$relation_name
        into :table_name
    do
    begin
      ibec_Progress('Counting records of ' || ibec_Trim(table_name));
      execute statement 'select count(*) from ' || ibec_Trim(table_name)
into :irecords;
      suspend;
    end
  end
```

ibec_Random

Generates random numbers within a specified range.

Syntax

```
  function ibec_Random(Range : integer): integer;
```

Description

`ibec_Random` returns a random number within the range `0 <= X < Range`. If `Range=0`, the result is a real-type random number within the range `0 <= X < 1`.

Example

```
  execute IBEBlock
  returns (iout integer, dpout double precision)
  as
  begin
    i = 0;
    while (i < 100) do
    begin
      iout = ibec_Random(100);
      dpout = ibec_Random(0);
      i = i + 1;
      suspend;
    end
  end
```

ibec_Random2

Generates random numbers within a specified range.

Syntax

```
function ibec_Random2(MinValue, MaxValue : integer): integer;
```

Description

ibec_Random2 returns a random number within the range MinValue <= X <= MaxValue.

Example

```
execute IBEBlock
returns (iout integer)
as
begin
  i = 0;
  while (i < 100) do
  begin
    iout = ibec_Random2(50, 100);
    i = i + 1;
    suspend;
  end
end
```

ibec_RandomChar

Generates random char within a specified range.

Syntax

```
function ibec_RandomChar(MinOrdValue, MaxOrdValue : integer): string;
```

Description

ibec_RandomChar returns a random char within the range MinOrdValue <= X <= MaxOrdValue.

Example

```
execute IBEBlock
returns (cout varchar(1))
as
begin
  i = 0;
  while (i < 100) do
  begin
    cout = ibec_RandomChar(1, 255);
    i = i + 1;
    suspend;
  end
end
```

ibec_RandomString

Returns a random string.

Syntax

```
  function ibec_RandomString(MinLen, MaxLen, MinOrdValue, MaxOrdValue :
integer): string;
```

Description

Example

ibec_SetLength

Sets the length of a dynamic-array variable.

Syntax

```
  function ibec_SetLength(AArray : array of variants; NewLength : inte-
ger): integer;
```

Description

`AArray` is a dynamic-array variable.

`ibec_SetLength` reallocates the array referenced by `AArray` to the given length. Existing elements in the array are preserved, the content of newly allocated elements is NULL.

`ibec_SetLength` returns the number of array elements.

Example

```
execute IBEBlock
returns (iresult integer)
as
begin
  vals = 0;
  iresult = ibec_SetLength(vals, 10);
  suspend; /* iresult = 10 */

  iresult = ibec_SetLength(vals, -1);
  suspend; /* illegal NewLength, iresult = 10 */

  iresult = ibec_SetLength(vals, '25');
  suspend; /* iresult = 25 */

  iresult = ibec_SetLength(vals, NULL);
  suspend; /* illegal NewLength, iresult = 25 */
end
```

ibec_ShiftRecord

Syntax

```
  function ibec_ShiftRecord(AArray : array of variants; Shift : integer):
integer;
```

Examples of usage of IBEBlock

This section includes a few examples illustrating the usage of `EXECUTE IBEBLOCK` (please refer to the individual subjects for details):

- Table data comparing
- Test data generator
- Joining tables from different databases
- Recreating indices #1
- Recreating indices #2
- Building an OLAP cube
- Inserting data from a file into a database
- Importing data from a CSV-file
- Creating script from Database Designer model file
- Creating an UPDATE-script with domain descriptions

IBEBLOCK and Test Data Generator

The following IBEBlock creates a table named `IBE$TEST_DATA` and populates it with random data.

```
execute ibeblock
returns (info varchar(100))
as
begin
  RecNum = 10000;

  if (exists (select rdb$relation_name from rdb$relations where
rdb$relation_name = 'IBE$$TEST_DATA')) then
  begin
    execute statement 'drop table IBE$$TEST_DATA';
    commit;
  end

 execute statement
  'create table IBE$$TEST_DATA (
     F_INTEGER integer,
     F_VARCHAR varchar(100),
     F_DATE date,
     F_TIME time,
     F_TIMESTAMP timestamp,
     F_NUMERIC numeric(15,2),
     F_BOOL char(1) check (F_BOOL in (''T'', ''F'')),
     F_BLOB blob sub_type 1,
     F_SEASON varchar(15) check(F_SEASON in (''Spring'', ''Summer'',
```

```
''Autumn'', ''Winter'')),
    F_RELS varchar(64))';
  commit;

  StartTime = ibec_gettickcount();

  i = 0;
  for select rdb$relation_name
  from rdb$relations
  into :rel_names
  do
  begin
    rels[i] = :rel_names;
    i = i + 1;
  end

  i = 0;
  while (i < RecNum) do
  begin
    fint    = ibec_random2(1, 100000);
    fvarc   = ibec_randomstring(1,100,65,90);
    fdate   = ibec_random2(20000,40000);
    ftime   = ibec_random(0);
    ftimest = ibec_random2(20000,40000) + ibec_random(0);
    fnum    = ibec_random2(1,40000) + ibec_random(0);
    fbool   = ibec_randomval('T','F');
    fblob   = ibec_randomstring(500, 1000, 0, 255);
    fseason = ibec_randomval('Spring', 'Summer', 'Autumn', 'Winter');
    frel    = rels[ibec_random2(0,ibec_high(rels))];

    insert into IBE$$TEST_DATA values (:fint, :fvarc, :fdate, :ftime,
:ftimest, :fnum, :fbool, :fblob, :fseason, :frel);
    i = i + 1;

    if (ibec_mod(i, 500) = 0) then
    begin
      ibec_progress(i || ' records inserted...');
      commit;
    end
  end

  commit;

  EndTime = ibec_gettickcount();
  info = 'Total time: ' || ((EndTime - StartTime) / 1000) || ' seconds';
  suspend;
  info = 'Per record: ' || ((EndTime - StartTime) / 1000 / RecNum) || '
seconds';
  suspend;
end
```

Joining tables from different databases

The following example illustrates how to join two tables from different databases:

```
execute ibeblock (iii integer, ivc varchar(100))
returns (id integer, ename varchar(100), company varchar(100))
as
begin

-- drop database 'localhost/3060:c:\db1.fdb' user 'SYSDBA' password 'mas-
terkey' clientlib 'C:\Program Files\Firebird\bin\fbclient.dll';

-- drop database 'localhost/3060:c:\db2.fdb' user 'SYSDBA' password 'mas-
terkey' clientlib 'C:\Program Files\Firebird\bin\fbclient.dll';

  create database 'localhost/3060:c:\db1.fdb' user 'SYSDBA' password
'masterkey'
  page_size 4096 sql_dialect 3
  clientlib 'C:\Program Files\Firebird\bin\fbclient.dll';

  create database 'localhost/3060:c:\db2.fdb' user 'SYSDBA' password
'masterkey'
  page_size 4096 sql_dialect 3
  clientlib 'C:\Program Files\Firebird\bin\fbclient.dll';

 create connection db1 dbname 'localhost/3060:c:\db1.fdb'
  password 'masterkey' user 'SYSDBA'
  clientlib 'C:\Program Files\Firebird\bin\fbclient.dll';

  create connection db2 dbname 'localhost/3060:c:\db2.fdb'
  password 'masterkey' user 'SYSDBA'
  sql_dialect 3
  clientlib 'C:\Program Files\Firebird\bin\fbclient.dll';

  use db1;

  vstmt = 'create table "employees" ( ' || '
     id integer not null primary key,
     full_name varchar(100),
     company_id integer)';

  execute statement :vstmt;

  commit;

  use default;

  select count(*) from help_items into :icount;

  use db1;
```

```
  insert into "employees" (id, full_name, company_id) values (1, 'Alexan-
der Khvastunov', 2);
  insert into "employees" (id, full_name, company_id) values (2, 'Bill
Gates', 1);
  insert into "employees" (id, full_name, company_id) values (3, 'John
Doe', NULL);
  insert into "employees" (id, full_name, company_id) values (4, 'Vladi-
mir Putin', 3);
  insert into "employees" (id, full_name, company_id) values (5, 'Some-
body', 15);

  use db2;

  execute statement
  'create table companies (
     id integer not null primary key,
     company_name varchar(100))';

  commit;

  insert into companies (id, company_name) values (1, 'Microsoft');
  insert into companies (id, company_name) values (2, 'HK-Software');
  insert into companies (id, company_name) values (3, 'The Kremlin?');

  commit;

  use db1;

  for execute statement 'select id, full_name, company_id from "employ-
ees"'
  into :id, :ename, :cid
  do
  begin
    use db2;

    company = NULL;

    select company_name from companies
    where id = :cid
    into :company;

    suspend;
  end

  close connection db1;
  close connection db2;
end
```

Recreating indices 1

The following example illustrates how to recreate database indices:

```
execute ibeblock
returns (info varchar(1000))
as
begin
  i = 0;

  for select i.rdb$index_name, i.rdb$relation_name, i.rdb$unique_flag,
             i.rdb$index_inactive, i.rdb$index_type
      from rdb$indices i
      left join rdb$relation_constraints rc on (i.rdb$index_name =
rc.rdb$index_name)
      where (i.rdb$system_flag is null) and (rc.rdb$index_name is null)
      into :IdxName, :IdxRelName, :IdxUnique, :IdxInactive, :IdxType
  do
  begin
    sFields = '';
    for select rdb$field_name from rdb$index_segments
        where rdb$index_name = :IdxName
        order by rdb$field_position
        into :ifields
    do
    begin
      if (sFields <> '') then
        sFields = sFields || ', ';
      sFields = sFields || ibec_formatident(ibec_trim(ifields));
    end

    DropStmt[i] = 'drop index ' || ibec_formatident(ibec_trim(IdxName));
    CreateStmt[i] = 'create ' || ibec_iif(IdxUnique = 1, 'unique ', '')
|| ibec_iif(IdxType = 1, 'descending ', '') ||

                    ' index ' || ibec_formatident(ibec_trim(IdxName)) ||
                    ' on ' || ibec_formatident(ibec_trim(IdxRelName)) ||
' (' || sFields || ')';

    i = i + 1;
  end

  i = 0;
  while (i <= ibec_high(DropStmt)) do
  begin
    s = DropStmt[i];
    info = s;
    suspend;
    ibec_progress(info);
    execute statement :s;
    commit;

    s = CreateStmt[i];
    info = s;
    suspend;
    ibec_progress(info);
```

```
    execute statement :s;
    commit;

    i = i + 1;
  end
end
```

Recreating indices 2

The following example illustrates how to recreate database indices using AS DATASET:

```
execute ibeblock
returns (info varchar(1000))
as
begin
  select i.rdb$index_name, i.rdb$relation_name, i.rdb$unique_flag,
         i.rdb$index_inactive, i.rdb$index_type
  from rdb$indices i
  left join rdb$relation_constraints rc on (i.rdb$index_name =
rc.rdb$index_name)
  where (i.rdb$system_flag is null) and (rc.rdb$index_name is null)
  as dataset ds_indices;

  while (not ibec_ds_eof(ds_indices)) do
  begin
    IdxName = ibec_trim(ibec_ds_getfield(ds_indices,0));
    IdxRelName = ibec_trim(ibec_ds_getfield(ds_indices,1));
    IdxUnique = ibec_ds_getfield(ds_indices,2);
    IdxInactive = ibec_ds_getfield(ds_indices,3);
    IdxType = ibec_ds_getfield(ds_indices,4);

    sFields = '';
    for select rdb$field_name from rdb$index_segments
        where rdb$index_name = :IdxName
        order by rdb$field_position
        into :IdxField
    do
    begin
      IdxField = ibec_trim(IdxField);
      if (sFields <> '') then
        sFields = sFields || ', ';
      sFields = sFields || ibec_formatident(IdxField);
    end

    DropStmt   = 'drop index ' || ibec_formatident(IdxName);
    CreateStmt = 'create ' || ibec_iif(IdxUnique = 1, 'unique ', '') ||
ibec_iif(IdxType = 1, 'descending ', '') ||

                 ' index ' || ibec_formatident(IdxName) ||
                 ' on ' || ibec_formatident(IdxRelName) || ' (' ||
sFields || ')';
```

```
    info = DropStmt;
    suspend;
    ibec_progress(info);
    execute statement :DropStmt;
    commit;

    info = CreateStmt;
    suspend;
    ibec_progress(info);
    execute statement :CreateStmt;
    commit;

    ibec_ds_next(ds_indices);
  end

  close dataset ds_indices;
end
```

Building an OLAP cube

The following illustrates the construction of an OLAP cube:


```
  execute ibeblock
  as
  begin
    SelectSQL = 'select rf.rdb$relation_name, f.rdb$field_type,
f.rdb$field_length, f.rdb$field_precision
                 from rdb$relation_fields rf, rdb$fields f
                 where rf.rdb$field_source = f.rdb$field_name';

    vDimensions[0] = 'FieldName=RDB$RELATION_NAME; Alias="Table Name"';
    vDimensions[1] = 'FieldName=RDB$FIELD_TYPE; Alias="Field Type';

    vMeasures[0] = 'FieldName=RDB$FIELD_TYPE; Alias="Field Count"; Calc-
Type=ctCount; Format=0';
    vMeasures[1] = 'FieldName=RDB$FIELD_LENGTH; Alias="Total Length";
CalcType=ctSum; Format=0';
    vMeasures[2] = 'FieldName=RDB$FIELD_PRECISION; Alias="Avg Precision";
CalcType=ctAverage';
```

Build and save cube in binary format:

```
    ibec_BuildCube('C:\test_cub.cub', SelectSQL, vDimensions, vMeasures,
null);
```

Build and save cube in XML format:

```
    ibec_BuildCube('C:\test_cub.xml', SelectSQL, vDimensions, vMeasures,
null);
  end
```

Inserting files into a database

IBEBlock can be used to insert files extremely simply and quickly into your database:

```
execute ibeblock
as
begin
  MyVar = ibec_LoadFromFile(C:\f.jpg);
  insert into ... values (..., :MyVar);
  commit;
end
```

Another possible way is to use different `SET BLOBFILE` statements before each `INSERT/UPDATE` statement:

```
SET BLOBFILE 'C:\f.jpg';
INSERT INTO ... VALUES (..., :h00000000_FFFFFFFF);
SET BLOBFILE 'C:\f2.jpg';
INSERT INTO ... VALUES (..., :h00000000_FFFFFFFF);
SET BLOBFILE 'C:\f3.jpg';
INSERT INTO ... VALUES (..., :h00000000_FFFFFFFF);
```

Inserting file data into a database

The following script should be executed in Script Executive or with IBEScript.

```
  set names win1251;
  set sql dialect 3;
  set clientlib 'C:\Program Files\Firebird\bin\fbclient.dll';

  create database 'localhost/3060:D:\allscripts.fdb'
  user 'SYSDBA' password 'masterkey'
  page_size 8192 default character set WIN1251;

  create generator gen_script_id;

  create table scripts (
    ID INTEGER NOT NULL PRIMARY KEY,
    FILENAME VARCHAR(2000),
    SCRIPT_TEXT BLOB SUB_TYPE TEXT);

  create trigger script_bi for scripts
  active before insert position 0
  as
  begin
    if (new.id is null) then
      new.id = gen_id(gen_script_id, 1);
  end;

  execute ibeblock
  as
  begin
```

```
    ibec_progress('Searching for script files...');
    files_count = ibec_getfiles(files_list, 'D:\',
'*.sql',_gfRecursiveSearch + __gfFullName);

    if (files_count > 0) then
    begin
      i = 0;
      while (i < ibec_high(files_list)) do
      begin
        file_name = files_list[i];
        file_size = ibec_filesize(file_name) / 1024 / 1024; -- File size
in megabytes
        if (file_size < 10) then
        begin
          script_data = ibec_loadfromfile(file_name);
          ibec_progress('Adding script file ' || :file_name);
          insert into scripts (filename, script_text) values (:file_name,
:script_data);
          commit;
        end
        i = i + 1;
      end
    end
  end;
```


Importing data from a CSV file

The following example creates a simple comma-separated values (CSV) file and imports its data into a database:

```
execute ibeblock
returns (outstr varchar(100))
as
begin
```

First, let's create a simple CSV-file with some data:

```
    FS = ibec_fs_OpenFile('C:\MyData.csv', __fmCreate);
    if (not FS is null) then
    begin
      s = '1:John:Doe:M';
      ibec_fs_Writeln(FS, s);
      s = '2:Bill:Gates:M';
      ibec_fs_Writeln(FS, s);
      s = '3:Sharon:Stone:F';
      ibec_fs_Writeln(FS, s);
      s = '4:Stephen:King:M';
      ibec_fs_Writeln(FS, s);
      ibec_fs_CloseFile(FS);
    end
```

If table IBE$$TEST_PEOPLE exists we'll drop it:

```
    if (exists(select rdb$relation_name from rdb$relations where
rdb$relation_name = 'IBE$$TEST_PEOPLE')) then
    begin
      s = 'DROP TABLE IBE$$TEST_PEOPLE';
      execute statement s;
      commit;
    end
```

Let's create a new table that will store the imported data:

```
    s = 'CREATE TABLE IBE$$TEST_PEOPLE (
         ID integer,
         FIRST_NAME varchar(50),
         LAST_NAME varchar(50),
         SEX varchar(1))';
    execute statement s;
    commit;

    i = 0; (-- Just a counter of inserted records)
    FS = ibec_fs_OpenFile('C:\MyData.csv', __fmOpenRead);
    if (not FS is null) then
    begin
      while (not ibec_fs_Eof(FS)) do
      begin
        s = ibec_fs_Readln(FS);
        ValCount = ibec_ParseCSVLine(Vals, s, '', ':',
__csvEmptyStringAsNull);
        INSERT INTO IBE$$TEST_PEOPLE (ID, FIRST_NAME, LAST_NAME, SEX)
VALUES :Vals;
        commit;
        i = i + 1;
      end
      ibec_fs_CloseFile(FS);
    end

    outstr = i || ' records inserted into IBE$$TEST_PEOPLE';
    suspend;
  end
```

Creating a script from a Database Designer model file

The following IBEBlock illustrates how to create a script from Database Model file:

```
  execute ibeblock
  as
  begin
    FileName = 'C:\model.grc';
    if ibec_FileExists(FileName) then
      ibec_CreateModelScript(FileName, 'C:\model.sql',
__msoDontQuoteIdents + __msoIncludeDescriptions);
  end
```

Creating an UPDATE script with domain descriptions

The following IBEBlock creates a script with UPDATE statements for all database domains that have a description:

```
execute ibeblock
  as
  begin
    FHSQL = ibec_fs_OpenFile('E:\DomDescs.sql', __fmCreate);
    FHBlobs = ibec_fs_OpenFile('E:\DomDescs.lob', __fmCreate);
    if ((not FHSQL is null) and (not FHBlobs is null)) then
    begin
      ibec_fs_Writeln(FHSQL, 'SET BLOBFILE ''E:\DomDescs.lob'';');
      ibec_fs_Writeln(FHSQL, '');
      for select rdb$field_name, rdb$description
          from rdb$fields
          where (rdb$description is not null)
          order by 1
          into :FieldName, :FieldDesc
      do
      begin
        if (FieldDesc <> '') then
        begin
          FieldName = ibec_Trim(FieldName);
          iOffs = ibec_fs_Position(FHBlobs);
          iLen = ibec_fs_WriteString(FHBlobs, FieldDesc);
          sParamName = ':h' || ibec_IntToHex(iOffs, 8) || '_' ||
ibec_IntToHex(iLen, 8);
          UpdStmt = 'UPDATE RDB$FIELDS' || ibec_Chr(13) || ibec_Chr(10)
||
                    'SET RDB$DESCRIPTION = ' || :sParamName ||
                    ibec_Chr(13) || ibec_Chr(10) ||
                    'WHERE (RDB$FIELD_NAME = ''' || FieldName || ''');';
          ibec_fs_Writeln(FHSQL, UpdStmt);
          ibec_fs_Writeln(FHSQL, '');
        end
      end
      ibec_fs_Writeln(FHSQL, 'COMMIT WORK;');
      ibec_fs_CloseFile(FHSQL);
      ibec_fs_CloseFile(FHBlobs);
    end
    commit;
  end;
```

8.25.2 IBECompare

IBECompare is a command-line tool to compare databases, scripts and table data. It is new to IBExpert version 2004.04.01.1. The current version (03/2005) is 2005.03.12.

IBECompare.exe can be found in the IBExpert root directory, and needs to be started from DOS:

```
c:\Program Files\HK-Software\IBExpert 2004>ibecompare
```

IBECompare offers the following options:

- **-C<config_file>** = config file
- **-O<output_file>** = output file (`Result.sql` if not specified)
- **-V<verbose_file>** = verbose file
- **-D** = compare database metadata and script
- **-T** = compare table data
- **-S** = silent mode
- **-s** = create a config file sample (`config_sample.ini`)

WARNING: All options are case-sensitive!

Example:

```
IBECompare -D -Cconfig.ini -OC:\Scripts\result.sql -Vlog.txt
```

In both cases (i.e. options `-D` or `-T`) IBECompare produces an SQL script file. It is necessary to specify an input settings file using the `-C` option.

You can obtain the template of this file starting IBECompare with the `-s` option (`IBECompare -s`). In this case IBECompare will create a `config_sample.ini` file within the current directory, which is simple and quick to modify.

It is also possible to create a settings file using *Save configuration* button in the IBExpert Tools menu / Database Comparer.

The following is an example of an `.ini` file, for comparing table data:

```
[MasterDB]
ConnectString=LOCALHOST:C:\MyData\Master.gdb
Username=SYSDBA
Password=masterkey
Charset=WIN_1251
ClientLib=gds32.dll
; Next item will be used while comparing tables
TableName=CUSTOMER

; Instead of MasterDB section you can use MasterScript section:
;[MasterScript]
;ScriptFile=D:\MyScripts\MyData.dql

[TargetDB]
ConnectString=MYSERVER:D:\Data\customer.gdb
Username=SYSDBA
Password=masterkey
Charset=WIN_1251
ClientLib=gds32.dll
; Next item will be used while comparing tables
TableName="Customer"
```

```
; Instead of TargetDB section you can use TargetScript section:
;[TargetScript]
;ScriptFile=D:\MyScripts\MyData.dql

[CompareObjects]
Domains=1
Tables=1
Views=1
Triggers=1
Procedures=1
Generators=1
Exceptions=1
Functions=1
Roles=1
Indices=1
Grants=1
Descriptions=1
PrimaryKeys=1
ForeignKeys=1
Uniques=1
Checks=1

[Options]
; Next items will be used while comparing tables
ProcessINSERTs=1
ProcessUPDATEs=1
ProcessDELETEs=1
```


Should the script generated by IBECompare include a

```
SET BLOBFILE 'xxx.lob';
```

command, it is necessary to execute the script using IBEScript or the IBExpert Script Executive.

`SET BLOBFILE` is a special extension of script language that allows insert or update blob values via script.

8.25.3 IBEExtract

`IBEEXtract.exe` can be found in the IBExpert root directory, and needs to be started from DOS. The current version (03/2005) is 2005.03.12.

Syntax:

```
IBEExtract database [options]
```

- **-U<user_name>** = user name ("SYSDBA" if not specified).
- **-P<password>** = password ("masterkey" if not specified).
- **-C<character_set>** = character set.
- **-O<output_file>** = output file ("Result.sql" if not specified).

- **-F<output_folder>** = output folder (for *Separate Files* mode; current directory, if not specified).
- **-G** = set generator values.
- **-D** = extract data.
- **-B** = extract blobs (please refer to blob fields for further information about blobs).
- **-S** = silent mode.
- **-V<verbose_file>** = verbose file.
- **-M<config_file>** = use config file.
- **-T** = generate `CREATE DATABASE` statement.
- **-N** = generate `CONNECT` statement.
- **-W** = include password into `CREATE DATABASE` or `CONNECT` statement.
- **-R** = extract object descriptions.
- **-A<integer_value>** = commit after `<integer_value>` records.
- **-Y** = extract computed fields separately.
- **-X** = extract privileges.
- **-L** = extract privileges only for selected objects.
- **-d** = date format (native InterBase/Firebird date format `<DD-MMM-YYYY>`, if not specified).
- **-f** = extract into separate files (new to IBExpert version 2004.9.12.1/IBEExtract version 2.02).
- **-s** = extract into separate files.
- **-r** = use `REINSERT` instead of repeated `INSERT`s.
- **-l** = client library file (`gds32.dll`, if not specified).
- **-z** = maximum size of resulting files in megabytes (new to IBExpert version 2004.9.12.1/IBEExtract version 2.02).

WARNING! All options are case-sensitive!

Example 1:

```
IBEExtract localhost:c:\mydata\mydatabase.gdb -OC:\scripts\result.sql -
USYSDBA -Pmasterkey -CWIN1251
```

Example 2:

```
IBEExtract "C:\IB Data\my.gdb" -O"My Script.sql" -V"Extract Log.txt"
```

Since IBExpert version 2003.11.6.1, the problem with extracting exceptions has been solved.

All options listed here can also be found in IBExpert under Tools / Extract Metadata.

8.25.4 IBEScript

IBEScript.exe can be found in the IBExpert root directory, and needs to be started from DOS. The current version (03/2005) is 2005.03.12.

Syntax:

```
IBEScript script_filename [options]
```

- **-S** = silent mode

- **-V<verbose_file>** = verbose output file. If <verbose_file> exists, IBEScript will overwrite it.
- **-v<verbose_file>** = verbose output file. If <verbose_file> exists, IBEScript will append message to this file.
- **-E** = display only error messages
- **-N** = continue after error.
- **-T** = write timestamp into log.
- **-D** = connections string (use it if your script does not contain CONNECT or CREATE DATABASE statements).
- **-P** = connection password (use only with -D option).
- **-U** = connection user name (use only with -D option).
- **-C** = character set (use only with -D option).
- **-L<1|2|3>** = SQL dialect (use only with -D option; 1 if not specified)
- **-i** = idle priority (new to IBExpert version 2004.9.12.1 / IBEScript version 2.02).

WARNING! All options are case-sensitive!

Since IBExpert version 2003.11.6.1 there is the added possibility to encrypt/decrypt scripts and to execute encrypted scripts. There are two possible ways to encrypt:

Encrypting without the password. In this case there is no possibility to decrypt an encrypted script but it is possible to execute this script with IBEScript.
Encrypting with the password. In this case it possible to decrypt the script and execute it with IBExpert if the correct password is specified.

The following options control the encrypting and decrypting:

- **-e** = encrypts a script file and create a file with the extension .esql if the output file is not specified (no execution will be performed).
- **-d** = decrypts an encrypted script file if it was encrypted with password (no execution will be performed).
- **-p<password>** = encrypt/decrypt password.
- **-o<file_name>** = output file name for encrypted and decrypted scripts.

Again: all options are case-sensitive!

Example 1:

```
IBEScript "C:\MyScripts\CreateDB.sql"
```

Example 2:

```
IBEScript C:\MyScripts\CreateDB.sql -S -UScriptLog.txt
```

Support for EXECUTE IBEBLOCK was implemented in IBEScript version 2.02 (released with IBExpert version 2004.9.12.1). This is unfortunately not available in the free version of IBEScript.

IBEScriptDll

New to IBExpert version 2004.12.12.1:

IBEScript.dll (for registered customers only)

For registered customers we've included the `IBEScript.dll` in the installation archive. You can use it in your applications to execute scripts from file or from a string buffer. There is a small demo application illustrating its use in the IBEScriptDll folder.. Please also refer to the IBEScriptDll Readme.txt.

To be allowed to distribute any of the IBExpert Modules (`ibexpert.exe`, `ibescript.exe`, `ibescript.dll`, `ibeextract.exe` and `ibecompare.exe`) together with your application, you need:

- IBExpert Site License, if the distribution is located only on computers in your own company.
- IBExpert VAR License, if the distribution is located on any computer outside your company.

If you are already an IBExpert customer, you can upgrade to a Site or VAR License and purchase the 24 month Extension Product.

See www.ibexpert.com PURCHASE area for details.

IBEScriptDll Readme.txt

1. `IBEScript.dll` exports the following functions:

- `ExecScriptFile` - executes script from file.
- `ExecScriptText` - executes script from string buffer.
- `CONNECT` - connects to the database if there is no `CONNECT` statement in the script.

2. Examples of the use of `ExecScriptFile` and `ExecScriptText` - see demo application in the IBEScriptDll folder.

3. Example using the `CONNECT` function:

```
procedure TForm1.Button2Click(Sender: TObject);
var
  Hndl : THandle;
  ESP : TExecuteScriptProc;
  CP : TConnectDBProc;
  s : string;
  Res : integer;
begin
  ErrCount := 0;
  StmtCount := 0;
  mLog.Lines.Clear;
  s := mScript.Text;
  if Trim(s) = '' then
  begin
```

```
    ShowMessage('Nothing to do!');
    Exit;
  end;
  try
    Hndl := LoadLibrary(PChar('IBEScript.dll'));
    if (Hndl > HINSTANCE_ERROR) then
    begin
      ESP := GetProcAddress(Hndl, 'ExecScriptText');
      CP := GetProcAddress(Hndl, 'Connect');
      if (@ESP <> nil) and (@CP <> nil) then
      begin
        Pages.ActivePage := tsOutput;
        Res := CP(PChar('db_name=localhost:c:\empty.fdb; pass-
word=masterkey; user_name=SYSDBA;'

+
                        'lc_ctype=win1251; sql_role_name=ADMIN;
sql_dialect=3;' +
                        'clientlib="c:\program
files\firebird\bin\fbclient.dll"'), @CEH);
        if Res = 0 then
          ESP(PChar(s), @HandleError, @BeforeExec, @AfterExec);
      end;
    end;
  finally
    if Hndl > HINSTANCE_ERROR then
      FreeLibrary(Hndl);
  end;
end;
```

8.26 InterBase and Firebird Command–Line Utilities

Several command-line tools are provided with InterBase/Firebird. They perform the same range of functions as the Server Manager and run on both UNIX and Windows platforms. Like the Server Manager, they can access servers on any platform that InterBase supports. The command-line tools include the following:

- GBAK
- GFIX
- GSEC
- GSTAT
- IBLOCKPR (Windows) GDS_LOCK_PRINT (Unix)
- IBMGR
- ISQL

The majority of the options provided by these command-line tools are also offered by IBExpert. Please refer to IBECompare, IBEExtract and IBEScript for further information.

8.26.1 GBAK and GSPLIT

(GBAK.EXE and GSPLIT.EXE)

GBAK is an InterBase/Firebird command line utility, which can be used to back up and restore databases. GSPLIT backs up and restores multiple file databases. Please refer to GBAK - Firebird Backup and Restore for further information.

The parameters and options offered by GBAK can be found in the IBExpert Database Backup and Database Restore menus.

GBAK – Firebird backup and restore

Many thanks to Stefan Heymann (www.destructor.de) for the following overview of options and examples.

GBAK is Firebird's/InterBase's command-line tool for online backup and restore of a complete database.

General Syntax:

```
gbak <options> -user <username> -password <password> <source> <destination>
```

Backup

For backups, <source> is the database you want to back up, <destination> is the file name of the backup file. The usual extension is .fbk for Firebird and .gbk for InterBase.

Only the SYSDBA or the database owner can perform a backup. For multi-file databases, specify only the name of the first file as the database name.

Restore

For restores, <source> is the backup file and <destination> is the name of the database that is to be built up from the backup file. You will have to specify the -C option for restore.

Options:

(Parts in square brackets are optional)

-b[ackup_database]	Back up. This switch is optional.	Backup only
-bu[ffers]	Set cache size for restored database	Restore only
-c[reate_database]	Restore (mandatory)	Restore only
-co[nvert]	Converts external tables to internal ta-	Backup

	bles	only
-e[xpand]	Creates an uncompressed backup	Backup only
-fa[ctor] n	Blocking factor for tape device	Backup only
-g[arbage collect]	Does not perform garbage collection (sweeping) during backup	Backup only
-i[nactive]	All indices will be restore as INACTIVE	Restore only
-ig[nore]	Ignores checksum errors while backing up	Backup only
-k[ill]	Does not create shadows that are defined in the backup	Restore only
-l[imbo]	Ignores Limbo transactions while backing up	Backup only
-m[etadata]	Only backs up metadata (schema). No table data will be stored	Backup only
-mo[de] read_write	Restores to a read/write database (This is the default)	Restore only
-mo[de] read_only	Restores to a read-only database	Restore only
-n[o_validity]	Does not restore validity constraints. So you can restore data that does not meet these constraints and could not be restored otherwise.	Restore only
-nt	Non-transportable format (use only when you know you will restore on same platform and database version)	Backup only
-o[ne_at_a_time]	Restores one table at a time. You can use this to partially restore databases with corrupt table data	Restore only
-ol[d_descriptions]	Old-style format	Backup only
-p[age_size] <size>	Sets page size of new database. <size> can be one of 1024, 2048, 4096, 8192. Default is 1024.	Restore only
-pa[ssword] <password>	Database password	
-r[eplace_database]	Restores over an existing database. This can only be performed by the SYSDBA or the owner of the database that is overwritten. Do NOT restore over a database that is in use!	Restore only

`-role <role>`	Connect as role	
`-se[rvice] <host-name>:service_mgr`	Backup: creates the backup file on the database server, using the Service Manager. Restore: creates the database from a backup file on the server, using the Service Manager.	
`-t[ransportable]`	Creates a transportable backup (transportable between platforms and server versions)	Backup only
`-u[ser] <username>`	Database user name	
`-use_[all_space]`	Normally, on restore, database pages will be filled to about 80 %. With the use_all_space option, database pages will be filled to 100 %. (Useful for read-only databases which will see no more modifications)	Restore only
`-v[erbose]`	Verbose output of what GBAK is doing	
`-y <filename>`	Redirect all output messages to `<filename>`. *NOTE*: the file *must* not exist before running GBAK!	
`-y suppress_output`	Quiet mode	
`-z`	Show GBAK version and server version	

Examples:

A "normal" backup:

```
gbak -v -t -user SYSDBA -password "masterkey" dbserver:/db/warehouse.fdb
c:\backups\warehouse.fbk
```

Backup with output to a logfile:

```
gbak -v -t -user SYSDBA -password masterkey -y c:\backups\warehouse.log
dbserver:/db/warehouse.fdb c:\backups\warehouse.fbk
```

A "normal" restore:

```
gbak -c -v -user SYSDBA -password masterkey c:\backups\warehouse.fbk
dbserver:/db/warehouse2.fdb
```

Restore to an already existing database:

```
gbak -c -r -v -user SYSDBA -password masterkey c:\backups\warehouse.fbk
dbserver:/db/warehouse.fdb
```

Create a read-only database:

```
gbak -c -v -mode read_only -use_all_space -user SYSDBA -password mas-
terkey c:\backups\warehouse.fbk c:\files\warehousedb.fdb
```

Multi-file backups:

Syntax for backup:

```
gbak [options] <database> <target file 1> <size 1> <target file 2> <size
2> ... <target file n>
```

NOTE: Do not specify a size for the last file. It will always be filled to take up what is left over, no matter how large. Size can be given in bytes (8192), kilobytes (1024k), megabytes (5m), or gigabytes (2g)

Syntax for restore:

```
gbak -c [options] <source file 1> <source file 2> ... <source file n>
<database>
```

Restoring to a multi-file database:

```
gbak -c [options] <source file> <db file 1> <size 1> <db file 2> <size 2>
... <db file n>
```


NOTE: Do not specify a size for the last database file. It can always grow unlimited to take up the rest. Size can be given in bytes (8192), kilobytes (1024k), megabytes (5m), or gigabytes (2g)

Restoring from a multi-file backup to a multi-file database:

```
gbak -c [options] <source file 1> <source file 2> ... <source file n> <db
file 1> <size 1> <db file 2> <size 2> ... <db file n>
```

8.26.2 GFIX

(GFIX.EXE)

GFIX is an InterBase/Firebird command-line utility, offering a number of options to validate and repair databases. These options are included in the IBExpert menu items Services / Database Validation and Database Properties.

The following articles are published here with the kind permission of Stefan Heymann (http://www.destructor.de/).

General Syntax

```
gfix [options] -user <username> -password <password> <database> [options]
```

Further information and examples can be found under the following subjects:

- Database Shutdown using GFIX
- Database Repair and Sweeping using GFIX

- GFIX - Miscellaneous parameters
- Using GFIX.

Database shutdown using GFIX

by Stefan Heymann.

Database Shutdown

When a database has been shut down, only SYSDBA and the database owner are able to connect to the database in order to perform administrative tasks.

Options:

`-at[tach] <seconds>`	Used with the `-shut` option. Waits `<seconds>` seconds for all current connections to end. If after `<seconds>` seconds there are still connections open, the shutdown will be cancelled.
`-f[orce] <seconds>`	Used with the `-shut` option. Waits `<seconds>` seconds for all connections and transactions to end. After this time, all connections and transactions are cancelled and the database is shut down. Use with caution.
`-o[nline]`	If a `-shut` operation is pending, it is cancelled. Otherwise, takes a database back online
`-sh[ut]`	Shut down database. Must be used together with `-attach`, `-force` or `-tran`.
`-tr[an] <seconds>`	Used with the `-shut` option. Waits `<seconds>` seconds for all running transactions to end. If after `<seconds>` seconds there are still running transactions, the shutdown will be cancelled.

Examples:

Shut down database, wait 60 seconds until all connections are closed:

```
gfix -user SYSDBA -password "masterkey" dbserver:/db/mydb.fdb -shut -
attach 60
```

Note that GFIX will terminate with an error if there are still connections open after 60 seconds.

Shut down database, force shutdown after 60 seconds:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -shut -force
60
```

Shut down database, force shutdown NOW:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -shut -force
0
```

Put database online again:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -online
```

Further examples can be found under Using GFIX and the subjects included under GFIX.

Database repair and sweeping using GFIX

by Stefan Heymann.

Options:

`-f[ull]`	Use with the `-v` option. Examines all records and pages and releases unassigned record fragments
`-h[ousekeeping] 0`	Switch off automatic sweeping
`-h[ousekeeping] <n>`	Set Sweep Interval to <n> transactions (default is 20000)
`-i[gnore]`	Ignores checksum errors during a validate or sweep
`-m[end]`	Marks corrupt records as unavailable so they are skipped on a subsequent backup
`-n[o_update]`	Use with the `-v` option. Examines all records and pages and reports errors but does not repair them
`-s[weep]`	Forces an immediate sweep
`-v[alidate]`	Check database for validity. At the same time, errors are reported and repaired

Examples:

Validate database:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -v -f
```

Sweep database now:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -s
```

Set sweep interval to 50000 transactions:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -h 50000
```

Switch off automatic sweeping:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -h 0
```

Further examples can be found under Using GFIX and the subjects included under GFIX.

GFIX – miscellaneous parameters

by Stefan Heymann.

Options:

-b[uffers] <pages>	Default cache buffers for the database will be set to <pages> pages
-c[ommit] <id>	Commits limbo transaction specified by the given <id>
-c[ommit] all	Commits all limbo transactions
-k[ill]	Drops shadows and unavailable shadows. Syntax is gfix -k (no database name).
-l[ist]	Display IDs of all Limbo transactions and what would happen to each transaction if you would use -t on it
-mo[de] read_write	Set mode of database to read/write (default). Requires exclusive access to database (shutdown)
-mo[de] read_only	Set mode of database to read-only. Requires exclusive access to database (shutdown)
-pa[ssword] <password>	Database password
-p[rompt]	Use with -l. Prompts for action.
-r[ollback] <id>	Rolls back limbo transaction specified by the given <id>
-r[ollback] all	Rolls back all limbo transactions
-s[ql_dialect] 1	Sets SQL dialect 1 for the database
-s[ql_dialect] 3	Sets SQL dialect 3 for the database
-t[wo_phase] <id>	Performs automated two-phase recovery for limbo transaction with the given <id>
-t[wo_phase] all	Performs automated two-phase recovery for all limbo transactions
-user <name>	Database username
-w[rite] sync	Enables Forced Writes
-w[rite] async	Disables Forced Writes
-z	Show GFIX and server version

Examples:

Set database to read-only:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -shut -attach
60g

fix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -shut -force 0
```

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -mode
read_only

gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -online
```

Set database to SQL dialect 3:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -sql_dialect
3
```

Enable forced writes:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -write sync
```

Disable forced writes:

```
gfix -user SYSDBA -password masterkey dbserver:/db/mydb.fdb -write async
```

Further examples can be found under Using GFIX and the subjects included under GFIX.

8.26.3 GSEC

(GSEC.EXE)

GSEC is an InterBase/Firebird command-line utility, which manages server security. It can be used to add, modify, and delete authorized users on the server. GSEC commands apply to the database server and not to individual databases, as with the majority of other command-line utilities.

All options offered by GSEC can be found in the IBExpert User Manager and Grant Manager.

Many thanks to Stefan Heymann (www.destructor.de) for the following overview of commands and options, and examples.

All database users are stored in the security database named security.fdb in the Firebird directory. There is at least one user, the system database administrator, SYSDBA.

After installation, the SYSDBA password is "masterkey". (Exception: Firebird 1.5 for Linux)

Only the first 8 characters of a password are significant. The password should not contain space characters.

Invoking GSEC:

GSEC can only be run by the SYSDBA.

To use GSEC for the local machine, use:

```
gsec -user sysdba -password <password> [options]
```

To use GSEC for a remote machine, use:

```
gsec -user sysdba -password <password> -database <databasename>
```

where <databasename> is the database name of the remote security.fdb database.

You can use GSEC as an interactive command line tool or give all commands on one command line.

Commands:

di[splay]	Displays all users
di[splay] <username>	Displays all information for the given user
a[dd] <username> -pw <password> [options]	Add a new user
mo[dify] <username> [options]	Modify user
de[lete] <username>	Delete user
h[elp]	Display help
?	Display help
q[uit]	Quit interactive mode
z	Display GSEC version number

If you don't want to invoke the interactive mode, you can enter all commands directly in the command line. To do that, precede the commands with a dash.

Options:

-pa[ssword] <password>	Password of the user who is performing the change
-user <username>	User name of the user who is performing the change
-pw <password>	Password of target user (or new password)
-fname <first name>	Target user's first name
-mname <middle name>	Target user's middle name
-lname <last name>	Target user's last name

Examples:

Add user Elvis Presley as user ELVIS, password is "Aaron":

```
gsec -user SYSDBA -password masterkey
GSEC> add elvis -pw Aaron -fname Elvis -lname Presley
GSEC> quit
```

Change password of user ELVIS to "chuck":

```
gsec -user SYSDBA -password masterkey
GSEC> modify elvis -pw chuck
GSEC> quit
```

Change password of SYSDBA on remote Linux server "harry" to "hamburg":

```
gsec -user SYSDBA -password masterkey -database
harry:/opt/firebird/security.fdb -modify sysdba -pw hamburg
```

Change password of SYSDBA on remote Windows server "sally" to "hannover":

```
gsec -user SYSDBA -password masterkey -database sally:"C:\Program
Files\Firebird\security.fdb" -modify sysdba -pw hannover
```

Change password of SYSDBA on remote server "jake" on TCP port 3051 to "london":

```
gsec -user SYSDBA -password masterkey -database
jake/3051:/opt/firebird/security.fdb" -modify sysdba -pw oberstein
```


Delete user Joe on local server:

```
gsec -user SYSDBA -password masterkey -delete joe
```

Notes:

On InterBase systems, the security database is named `isc4.gdb`.
There will be a warning when a new password is longer than 8 characters.

8.26.4 GSTAT

(`GSTAT.EXE`)

GSTAT is an InterBase/Firebird command-line utility, which can be used to display database statistics related to transaction inventory, data distribution within a database, and index efficiency.

All information offered by this tool can be found in the IBExpert Services menu item, Database Statistics.

8.26.5 IBLOCKPR (Windows) and GDS_LOCK_PRINT (Unix)

(`IBLOCKPR.EXE` on Windows) and `gds_lock_print` (UNIX)

These utilities display statistics for the InterBase lock manager.

8.26.6 IBMGR

(IBMGR.EXE)

IBMGR is a windows-based server management program, and includes the functionalities found in GSEC, GBAK and GFIX.

8.26.7 ISQL

ISQL is a program which can be used to run SQL queries on the database. ISQL supports data definitions and data manipulation commands as well as SQL scripts with multiple SQL commands within one script. It can be used to create and modify the database's metadata, insertion, alteration and deletion of data, data queries and the display of results (all this can be done in the IBExpert SQL Editor), adding and removal of user database rights (see IBExpert User Manager and Grant Manager) and execution of other database administrative functions.

ISQL commands end with ;. Each command must be explicitly committed using the commit statement.

9 IBExpert Services Menu

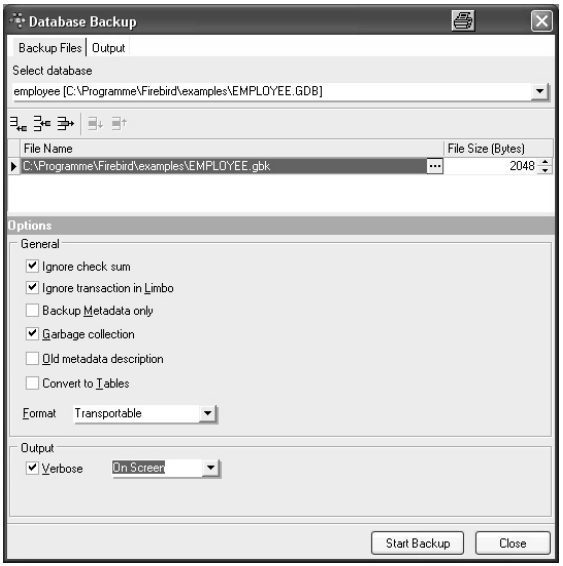
The IBExpert Services menu offers the following range of services:

- Database Backup
- Database Restore
- Server Properties/Log
- Activation Certificates
- Database Validation
- Database Statistics
- Database Properties
- Database Shutdown
- Database Online
- Communication Diagnostics

9.1 Backup Database

The IBExpert Services menu item Database Backup allows you to create a backup or copy of the database, saving it to file. This database copy may be kept simply for security reasons, or restored for the reasons detailed in "Why is a Database Backup and Restore Important?".

A database backup may be performed without having to disconnect the database; users may continue their work as InterBase/Firebird uses its multigenerational architecture to take a snapshot of the database at a moment in time the backup is requested. All information generated by committed transactions and present at this moment, is backed up.



First select the database to be backed up from the pull-down list of registered databases. Then select either an existing backup file name, or add a new backup file using

the Insert File icon (or [Ins] key). The [...] button to the right of this row allows you to find an existing file or specify the drive, path and backup file name for a new file. The suffixes .GBK and .FBK are traditionally respectively used for InterBase and Firebird backup files. A file size only needs to be specified when working with secondary files. All files in a multifile database are backed up (i.e. both secondary files and shadow files). InterBase/Firebird understands the links that exist with secondary database files and with shadows. Whereas the operating system backup works on a file-by-file basis, InterBase/Firebird always backs up all files in a database.

Backup Options:

Ignore check sum: If this option is checked, check sum errors in the database header pages, where the database connection properties are stored, are ignored in the backup. As InterBase and Firebird normally abort the backup when check sum errors are discovered, this is a way to force a backup when there are problems. Note that UNIX versions do not use check sums.

Ignore transactions in Limbo: If this option is checked, Transactions in Limbo, i.e. transactions, that can't be defined as executed or aborted, are ignored in the backup. Only those most recent, committed transactions are backed up. It allows a database to be backed up before recovering corrupted transactions. Generally in limbo transactions should be recovered before a backup is performed.

Backup Metadata only: If this option is checked, only the database's definition (i.e. the metadata, which provides an empty copy of the database) is saved. (If a database copy with certain data content is required, then use the IBExpert Script Executive.)

Garbage collection: If this option is checked, garbage collection is executed during the backup. By disabling this option, the backup can be speeded up considerably. (Refer to garbage collection for further information.)

Old metadata description: If this option is checked, old metadata descriptions are included into the backup database. This is included for compatibility reasons for older InterBase versions.

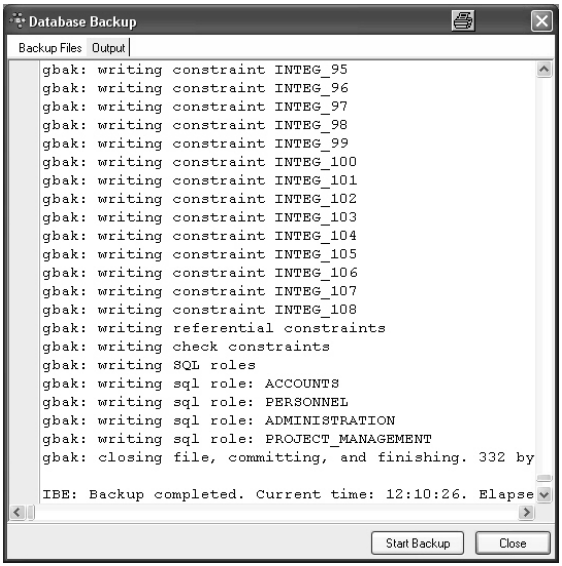
Convert to Tables: This option converts the database data to tables in the backup. This concerns external files. It is possible in InterBase/Firebird to create a table as an external file - this option converts them to internal database tables.

Format: Select the data format for the backup database file. *Transportable* is the recommended default option, as it allows a restore into different InterBase/Firebird Versions if wished, i.e. it saves the data and metadata to a generic format, as opposed to the option Non-Transportable. (Please note that when backing up and restoring, for example, from InterBase 4 to Firebird 1.5, stored procedures are restored as blobs, so that they may not initially work.)

Verbose: Checking *Verbose* provides a detailed protocol of the current database backup process, by writing step-by-step status information to the output log. Select the option *On Screen* or *Into File* (not forgetting to select or specify a file name for this protocol) before starting the backup. This option is useful if the backup is failing and the reason needs to be analyzed.

Then start the backup. If the protocol option *On Screen* was selected, the backup is logged on the Output page.

Using the IBExpert menu item Database / Database Registration Info, default backup file names, paths and drives may be specified if wished, along with default backup and restore options. This information may be specified when initially registering a database in IBExpert (see Register Database) or at a later date (see Database Registration Info).



In normal circumstances, the backup should run smoothly without any of the above options having to be checked. If however, corrupt or damaged data is suspected or problems have been encountered, alter the Format to Non-Transportable and check the options *Ignore Check Sum* and *Ignore Transactions in Limbo*. Although this will not provide the usual database compression, it does provide a complete copy of the database, which is important before starting to repair it.

It is also possible to validate the database using Services / Database Validation or GFIX, before retrying.

9.1.1 Why is a database backup and restore important?

Performing regular backups protects from hardware failures and data corruption, which cannot be fixed by the InterBase/Firebird maintenance tools. It is important to use the InterBase/Firebird backup and restore facilities even though most networks include a facility for data backup and restore across the network, because:

- Operating system backups require exclusive access to the database. The InterBase/Firebird backup runs parallel with concurrent database accesses by other users. InterBase/Firebird uses its multigenerational architecture to take a snapshot of the database at a moment in time for the backup. All information generated by committed transactions and present at this moment is backed up.

- All files in a multifile database are backed up. InterBase/Firebird comprehends the links between the different database files and shadows. The operating system backup processes files one after the other and saves them to the specified file or medium, so that all the various files are backed up in different versions and they cannot work together correctly anymore when restored. The InterBase/Firebird backup backs up all database files automatically.
- The different versions of InterBase/Firebird use different database file formats, so that it is impossible to copy a file directly from one operating system environment to the required format of another operating system environment. The InterBase/Firebird backup utility allows a transportable backup format, so that this file can be restored on any desired InterBase/Firebird platform.
  Please note: When backing up and restoring, for example, from InterBase 4 to Firebird 1.5, stored procedures are restored as blobs, so that they may not initially work.
- The InterBase/Firebird backup discards outdated data sets and index files, resulting in a smaller backup (please refer to garbage collection for more information).
- Empty pages are also automatically removed during a backup and restore, which reduces the total database size. The transaction number in the TIP is reset to zero (the total number of transactions that can be recorded in a TIP is approximately 1.3 billion!). The cache works with considerably more efficiency following a backup and restore as the pages are reordered.
  Please note: In Firebird 1.5 the new memory manager allows new data sets to automatically be stored in old pages, without first having to backup and restore.
- During an InterBase/Firebird backup the integrity and references for all database objects, e.g. domains, tables, indices, views, triggers, stored procedures, generators, exceptions, and permissions, are checked.
- Executing a backup and restore is the only way to subsequently alter fundamental parameters in the database structure, such as the page size and distribution across secondary files.
  It is therefore recommended to not only backup but also restore the database regularly (e.g. once a month).

9.1.2 Garbage collection

When performing a garbage collection, InterBase/Firebird does nothing other than remove outdated data sets and index files, which results in a smaller database. Outdated data sets are stored by InterBase/Firebird for the following reason: InterBase/Firebird are multigenerational databases. When a data set is altered, this alteration is stored in the database as a new copy. The old values remain in the database as a back version, which is the rollback protocol. If the transaction is rolled back after the update, the old value is ready to resume its function as the valid value. If the transaction is however committed, and not rolled back, this back version becomes superfluous. In databases with a lot of update operations this can result in a lot of garbage.

When garbage is collected in InterBase/Firebird, not only the out-of-date update values are deleted, but all outdated and deleted data set versions, based on the Transaction Inventory Page (TIP).

A garbage collection is only performed during a database sweep, database backup or when a SELECT query is made on a table (and not by insert, alter or delete). Whenever InterBase touches a row, such as during a SELECT operation, the versioning engine sweeps out any versions of the row where the transaction number is older than the

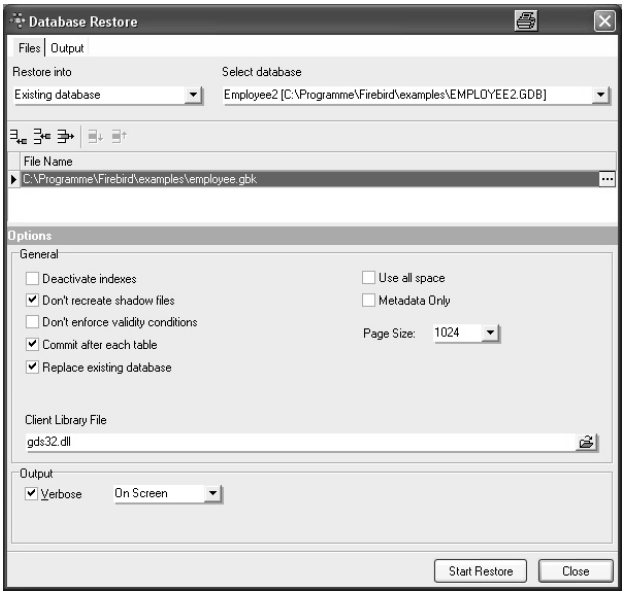
Oldest Interesting Transaction (OIT). This helps to keep the version history small and manageable and also keeps performance reasonable.

The sweep interval (i.e. at what interval (in number of transactions) a database sweep should be automatically conducted) for the garbage collection may be specified under the IBExpert menu item Services / Database Properties.

The garbage collection may be performed during 24 hour operation online without any problems (i.e. the server does not need to be shut down). Performance may however be slower during the database sweep which may not be desirable. If the sweep interval is specified at zero (0) (see Database Properties), the garbage collection is not performed automatically at all. It could then be carried out, for example, at night as a sweep or backup using GFIX and the `at` Windows command or the Linux `chron` command.

9.2 Restore Database

The IBExpert Services menu item Restore Database allows you to restore the database from a backed up file. Before restoring a backup file into a database, it is important to first disconnect the database! - Otherwise you could end up with a corrupt database should users try to log in and perform data operations during the restore.



The Files page allows the following specifications:

Restore into: Select to restore into the existing database, or create a new database. When restoring into the existing database, select it from the list of registered databases; if restoring to a new database, then set the database file name not forgetting the drive and path.

Specify the backup file name which is to be restored. The [...] button to the right of this row allows you to find an existing file name, drive, and path. The suffixes .GBK and .FBK are traditionally respectively used for InterBase and Firebird backup files.

The following restore options may be checked/unchecked as wished:

Deactivate indexes: If this option is checked, database indices are deactivated while restoring. This option is used to improve restore performance. If this option is not checked, InterBase/Firebird updates indices after all tables have been populated with the restored rows. This option may also be necessary if the database contains data with a unique index, but there are values in the table that are not actually unique. It can also be used when the field length in one or more tables is to be altered retrospectively; or when an index is simply not working due to some undiscovered inconsistencies.

Don't recreate shadow files: If this option is checked, shadow files are not recreated while restoring. *Restoring without Shadow* deletes the shadow definition; to restore it, it is necessary to recreate the shadow using the CREATE SHADOW statement (please refer to Creating a Shadow for further information). This option is sometimes required if the destination database does not support shadows, if you are migrating from an earlier version of InterBase where shadows were not supported, or if the machine where the shadow resides is not available.

Don't enforce validity conditions: When this option is checked, database validity conditions such as constraints on fields or tables are not restored. This option is useful if the validity constraints were changed after data had already been entered into the database. When a database is restored, InterBase/Firebird compares each row with the metadata; an error message is received if incompatible data is found. Once the offending data has been corrected, the constraints can be added back.

Commit after each table: If this option is checked, IB Manager commits work after restoring each table. This allows all those tables to be restored and committed where there is no corrupted data. It restores metadata and data for each table in turn as a single transaction and then commits the transaction. This option is useful if corrupt data is suspected in the backup file, or if the backup is not running to completion. Normally, InterBase/Firebird restores all metadata and then restores the data.

Should you encounter problems when restoring your database, deactivate this option and retry.

Replace existing database: If this option is checked the restored database replaces the existing one. Leaving this option unchecked provides a measure of protection from accidentally overwriting a database file.

Use All Space: This option should be checked when restoring the database onto a CD, as all (i.e. 100%) space is then used, as opposed to the usual 80% for databases which are subject to alterations and stored on hard drives.

Metadata Only: This option produces an empty copy of the database. It may also be used to restore the framework of a corrupt database, to allow analysis and repair work.

Client Library: This is new in version 2003.11.6.1 and is an added possibility to specify a client library which will be used while restoring. This option allows the user to specify whether he requires the InterBase or the Firebird client library for each IBExpert connection. The default client library is `gds32.dll`.

Page size: Database page size in bytes. This is the only option allowing the page size for an existing database to be altered.

Verbose: Check *Verbose* to receive a detailed protocol of the current database backup process, by writing step-by-step status information to the output log. The options *On Screen* or *Into File* (not forgetting to select or specify a file name for this protocol) need to be specified before starting the backup. This option is useful if the restore is failing, and the reason needs to be analyzed.

The restore can then be started. If the protocol option *On Screen* was selected, the backup is logged on the Output page.

Under normal circumstances, none of the above restore options should need to be specified. If inconsistencies between the metadata and the data itself are suspected, check the *Commit After Each Table*, *Deactivate Indexes*, and *Don't Enforce Validity Conditions* options.

Please note that InterBase/Firebird does not backup indices. It only backs up the index definition. When the database is restored InterBase/Firebird uses this definition to regenerate the indices.

Using the Database Registration dialog, default backup file names, paths and drives may be specified if wished, along with default backup and restore options. This information may be specified when initially registering a database in IBExpert (see Register Database) or at a later date (see Database Registration Info).

Empty pages are automatically removed during a backup and restore, which reduces the total database size. The transaction number in the TIP is reset to zero. The cache works with considerably more efficiency following a backup and restore as the pages are reordered. It is therefore recommended not only to backup but also to restore the database regularly (e.g. once a month).

In Firebird 1.5 the new memory manager allows new data sets to automatically be stored in old pages, without first having to backup and restore.

9.2.1 Database Shadow Files

Shadow files are an exact live copy of the original active database, allowing you to maintain live duplicates of your production database, which can be brought into production in the event of a hardware failure. These shadows are administrated in real time by the InterBase/Firebird server. They are used for security reasons: should the original database be damaged or incapacitated by hardware problems, the shadow can immediately take over as the primary database. It is therefore important that shadow files do not run on the same server or at least on the same drive as the primary database files. Shadow files are not normally used on Windows platforms, as the shadow file has to be on the same computer as the active database. These do work however on LINUX/UNIX.

InterBase allows up to 65,536 (216) database files, including shadow files. However the operating system used may have a lower limit on the number of simultaneous open files that the IBServer/FBServer can have. In some cases, the OS provides a means to raise this limit (refer to your OS documentation for the default open files limit, and the means to raise it).

Shadow files, as with the main database and secondary files, may not reside on networked or remote file systems (i.e. mapped drives on Windows and NFS files on UNIX).

The number of existing shadow files in a database may be ascertained using the IBExpert Services menu item Database Statistics, or using GSTAT (the shadow count is included in the database header page information).

Shadowing offers a number of advantages:

- It provides valuable protection of the database, in addition to the regular backups which should be maintained, and in addition to InterBase/Firebird's multigenerational architecture.
- If the original database is damaged, the shadow can be activated immediately, with little lost time.
- Shadowing runs automatically with little or no maintenance.
- You have full control over the shadow's configuration, including its use of hard disk space and distribution across other available devices.
- Creating a shadow does not require exclusive access to the database.
- Shadow files use the same amount of disk space as the database. As opposed to log files, which can grow well beyond the size of the database.
- Shadowing does not use a separate process. The database process handles writing to the shadow.

But there are also some limitations:

- Shadowing only helps to recover from certain types of problems. If a user error or InterBase/Firebird problem causes the database to be damaged beyond recovery, then the shadow is identically damaged. But if the database is accidentally deleted by the user, or a hardware problem on the primary server occurs, the shadow remains intact and can be used immediately.
- Shadowing is *not* replication. It is one-way writing, duplicating every write operation on the master database. Client applications cannot access the shadow file directly.
- The shadow cannot be used to rollback the database to a specific point in time. When the shadow is used to recover the database, everything up to the point where the original problem occurred is retrieved.
- Shadowing adds a small performance penalty to database operations. Every action on the database which modifies metadata or the data itself is mirrored in the shadow.
- Shadowing does not replace a careful security system within the operating system, but is one aspect or enhancement of the whole.
- Shadowing also works only for operations that go through the InterBase/Firebird database services manager (GDS), which processes all SQL and database requests.
- Shadowing can occur only to a local disk. Shadowing to a NFS file system or mapped drive is not possible. Shadowing to tape or other media is also not possible.

Tasks for Shadowing:

The main tasks in setting up and maintaining shadows are as follows (please refer directly to these subjects for further information):

Creating a shadow

(Source: InterBase® 7.1 Operations Guide)

Shadowing begins with the creation of a shadow, using the `CREATE SHADOW` statement. This statement has the following syntax:

```
CREATE SHADOW shadow_number
[AUTO | MANUAL] [CONDITIONAL] shadow_filename
```

The shadow number identifies a shadow set that collects the primary shadow file and any secondary files together. The most important function of the shadow number is to identify the shadow if you decide to drop it (please refer to Deleting a Shadow).

This can be performed without affecting users at all, as it does not require exclusive access. Before creating the shadow, the following should be considered:

- **Shadow location**: a shadow should be created on a different disk from the main database, as shadowing is intended as a recovery mechanism in case of disk failure. Therefore storing the main database and the shadow on the same disk defeats the whole purpose of shadowing!
- **Distributing the shadow**: a shadow can be created as a single-file (shadow file) or as multiple files (shadow set). To improve space allocation and disk I/O, each file in a shadow set may be placed on a different disk.
- **User access**: if a shadow becomes unavailable, user access to the database can be denied until shadowing is resumed, or access can be allowed (i.e. work can continue as normal) although any changes made during this period will obviously not be shadowed. Please refer to auto mode and manual mode for further information.
- **Automatic shadow creation**: To ensure that a new shadow is automatically created, create a conditional shadow. Please refer to conditional shadows for further information.

Please note: If the IBExpert Services menu item Restore Database dialog option, *Don't Recreate Shadow Files* is checked, shadow files are not recreated while restoring. This deletes the shadow definition; and to restore it, it is necessary to recreate the shadow using the `CREATE SHADOW` statement. This option is sometimes required if the destination database does not support shadows, if you are migrating from an earlier version of InterBase where shadows are not supported, or if the machine where the shadow resides is not available.

The following sections deal with the creation of shadows with various options:

- Creating Single-file or Multifile Shadows
- Auto Mode and Manual Mode
- Conditional Shadows

These options are not mutually exclusive, e.g. it is possible to create a single-file conditional shadow with the option manual mode.

Creating single–file or multifile shadows

(Source: InterBase® 7.1 Operations Guide)

To create a single-file shadow for the sample database `employee.gdb`, enter the following in the IBExpert SQL Editor:

```
CREATE SHADOW 1 '/usr/interbase/examples/employee.shd';
```

The name of the shadow file is `employee.shd`, and it is identified by the number `1`.
It is possible to verify that the shadow has been created by using the isql command:

```
SHOW DATABASE;
   Database: employee.gdb
   Shadow 1: '/usr/interbase/examples/employee.shd' auto
   PAGE_SIZE 4096
   Number of DB pages allocated = 392
   Sweep interval = 20000
```

The page size of the shadow is the same as that of the database.

A large database may be shadowed to a multifile shadow if wished, spreading the shadow files over several disks. Each file in the shadow set needs to be specified by name and size. This can be specified in two ways, the same as with multifile databases:

- Specify the page on which each secondary file starts
- Specify the length in database pages of each file.

You can specify both but this is redundant. If the information specified is inconsistent, InterBase/Firebird uses the length value in preference to the starting page value. In general, it is best to use *either* length values *or* starting page number to ensure consistency or legibility.

If the files are specified using the `LENGTH` keyword, do not specify the length of the final file, as InterBase/Firebird sizes the final file dynamically, as needed. Please refer to secondary files for further information.

The following example creates a shadow set consisting of three files. The primary file, `EMPLOYEE.SHD` is 10,000 database pages in length; the second file is 20,000 pages long, and the final file is left open, to expand as needed.

```
CREATE SHADOW 1 'employee.shd' LENGTH 10000
   FILE 'emp2.shd' LENGTH 20000
   FILE 'emp3.shd';
```

The second alternative is to specify the starting page of the files:

```
CREATE SHADOW 1 'employee.shd'
   FILE 'emp1.shd' STARTING AT 10000
   FILE 'emp2.shd' STARTING AT 30000;
```

Using the SHOW DATABASE command, the file names, page lengths or starting pages can be verified:

```
SHOW DATABASE;
   Database: employee.gdb
   Shadow 1: '/usr/interbase/examples/employee.shd' auto length 10000
   file /usr/interbase/examples/emp1.shd length 2000 starting 10000
   file /usr/interbase/examples/emp2.shd length 2000 starting 30000
   PAGE_SIZE 4096
   Number of DB pages allocated = 392
   Sweep interval = 20000
```

The page length for secondary files in the main database does not need to correspond to the page length for the secondary shadow files. As the database grows and its first shadow file becomes full, updates to the database automatically overflow into the next shadow file.

Auto mode and manual mode

(Source: InterBase® 7.1 Operations Guide)

A shadow database may become unavailable for the same reasons a database becomes unavailable (e.g. disk failure, network failure, or accidental deletion). If a shadow has been created in auto mode and suddenly becomes unavailable, database operations continue automatically without shadowing. If the shadow was created in manual mode, further access to the database is denied until the database administrator gives explicit instructions, as to how work is to be continued.

The benefits of auto mode and manual mode may be compared below:

Mode	*Advantage*	*Disadvantage*
Auto	Database operation is uninterrupted	Creates a temporary period when the database is not shadowed. The database administrator might be unaware that the database is operating without a shadow.
Manual	Prevents the database from running unintentionally without a shadow	Database operation is halted until the problem is fixed. Needs intervention of the database administrator.

Auto mode

The AUTO keyword can be used to create a shadow in auto mode:

```
CREATE SHADOW 1 AUTO 'employee.shd';
```

Auto mode is the default, so this does not necessarily need to be specified explicitly.

In auto mode, database operation is uninterrupted even though there is no shadow. To resume shadowing, it might be necessary to create a new shadow. If the original shadow was created as a conditional shadow, a new shadow is automatically created. Please refer to conditional shadows for further information.

Manual mode

The `MANUAL` keyword can be used to create a shadow in manual mode:

```
CREATE SHADOW 1 MANUAL 'employee.shd';
```

Manual mode is useful when continuous shadowing is more important than continuous operation of the database. When a manual-mode shadow becomes unavailable, further operations on the database are prevented. To allow work on the database to be resumed, the database owner or SYSDBA must enter the following command:

```
gfix -kill database
```

This command deletes metadata references to the unavailable shadow corresponding to the `database`. After deleting the references, a new shadow can be created if shadowing needs to be resumed.

Shadow information is kept in the metadata of the primary database file. If this file becomes unavailable for some reason, then the pointers to the shadow are also broken. In this situation, the database administrator can use the `-active` option in the GFIX utility to convert the original shadow into a new primary database.

Conditional shadows

(Source: InterBase® 7.1 Operations Guide)

A shadow may be defined so that if it replaces a database, the server creates a new shadow file, and thus allows shadowing to continue uninterrupted. This is termed a conditional shadow, and is specified using the `CONDITIONAL` keyword:

```
CREATE SHADOW 3 CONDITIONAL 'atlas.shd';
```

Creating a conditional file automatically creates a new shadow in either of two situations:

- The database or one of its shadow files becomes unavailable.
- The shadow takes over for the database due to hardware failure.

Activating a shadow

(Source: InterBase® 7.1 Operations Guide)

Should the main database become unavailable for whatever reason, the shadow can be activated, i.e. it takes over the main database and all users now access the shadow as

the main database. This activation may be defined to occur automatically or through the intervention of the database administrator.

Shadow information is kept in the metadata of the primary database file. If this file becomes unavailable for some reason, then the pointers to the shadow are also broken. To activate the shadow it is necessary to log in as SYSDBA or the database owner, and use GFIX with the `-activate` option, to convert the original shadow into a new primary database.

Important! The first step is to make sure the shadow is not active, i.e. if the main database has active transactions the shadow is active. Also check that the main database is unavailable. If a shadow is activated while the main database is still available, the shadow can be corrupted by existing attachments to the main database.

To activate a shadow, specify the path name of its primary file. For example, if database `employee.gdb` has a shadow named `employee.shd`, enter:

```
gfix -a[ctivate] shadow_name
```

The shadow name is the explicit path and name of the shadow's primary file.

Examples:

For a Windows NT server:

```
gfix -a F:\SHADOW\ORDENT\ORDERS.SHD
```

For any UNIX server:

```
gfix -a /usr/shadow/ordent/orders.shd
```

After a shadow is activated its name should be changed to the name of the original database. Then a new shadow can be created if shadowing needs to continue providing another disk drive is available.

Deleting a shadow

(Source: InterBase® 7.1 Operations Guide)

If a shadow is no longer needed, it can be stopped by simply deleting it. To stop shadowing, use the shadow number as an argument with the `DROP SHADOW` statement. For example:

```
DROP SHADOW 1
```

If you need to look up the shadow number, use the isql command `SHOW DATABASE`.

Important! `DROP SHADOW` deletes all shadow references from a database's metadata as well as the physical files on disk. Once the files have been removed from the disk, there is no way to recover them. However, as a shadow is merely a copy of an existing database, a new shadow will be identical to the dropped shadow.

Please note that when a database is dropped/deleted, all secondary and shadow files are also deleted. The complete structure and all the data is deleted permanently!

Adding files to a shadow/modifying a shadow

(Source: InterBase® 7.1 Operations Guide)

Shadow databases may consist of multiple files. As the shadow grows in size, files may need to be added to cope with the increase in space requirements.

To modify a shadow database or add a shadow file, first use the `DROP SHADOW` statement to delete the existing shadow, then use the `CREATE SHADOW` statement to create a multifile shadow.

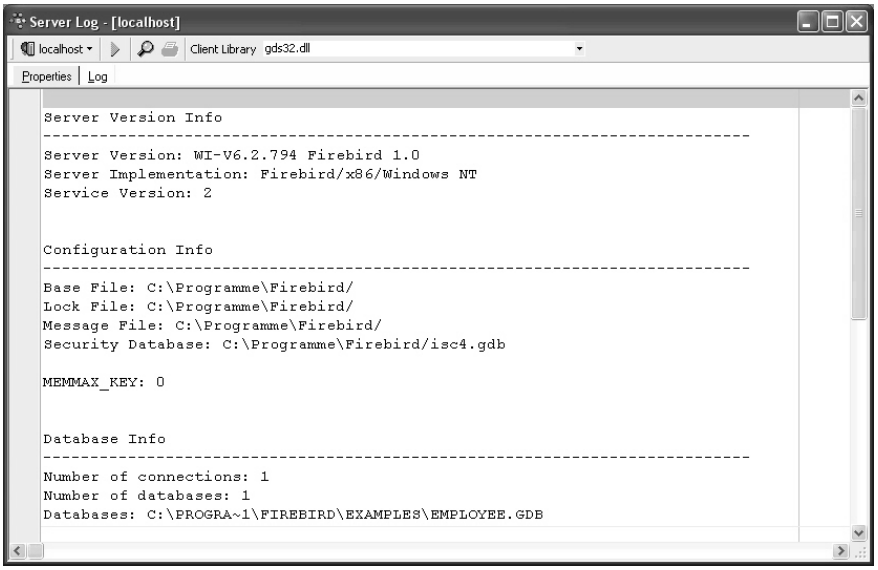
Example:

```
DROP SHADOW 2
CREATE SHADOW 3 AUTO CONDITIONAL
'F:\SHADOW\ORDENT\ORDERS.SHD' LENGTH 10000
FILE 'F:\SHADOW\OIRDENT\ORDERS2.SHD'
```

The page length allocated for secondary shadow files need not correspond to the page length of the database's secondary files. As the database grows and its first shadow file becomes full, updates to the database automatically overflow into the next shadow file.

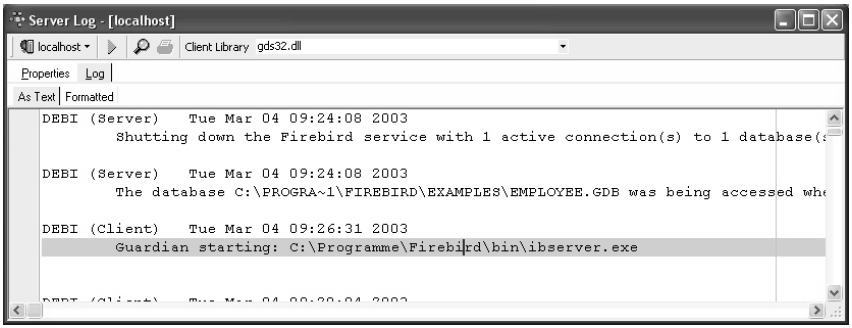
9.3 Server Properties/Log

The Server Properties page displays the following information:

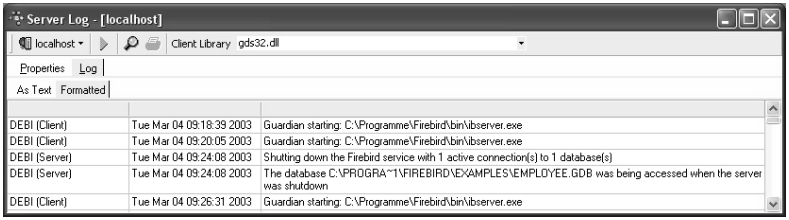


It includes server version information, configuration information and database information, particularly interesting, when working with remote and/or multiple connections.

The log can be started using the Retrieve (green arrow) icon. The log page displays information either as text:



or in a grid form:



The log may even be printed - the print preview can be opened using the magnifying glass icon.

9.4 Server Activation Certificates

This option is purely for Borland InterBase v 6.5. It allows new InterBase users to be registered or existing users to be removed directly in IBExpert, using the Borland InterBase certificate keys and IDs, without having to use IBConsole.

9.5 Database Validation

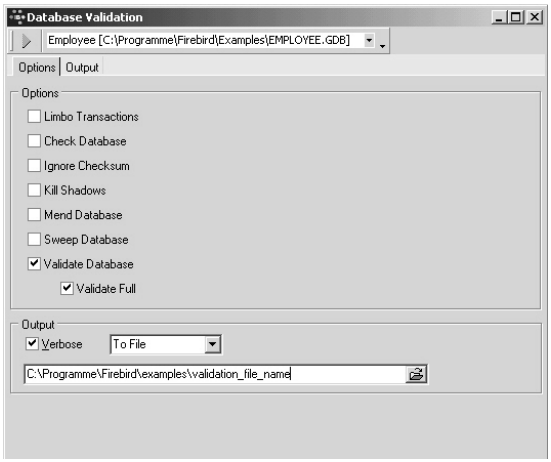
Database validation involves checking the database file to ensure that the various data structures retain their integrity and internal consistency. The validation process checks for three different types of problems:

- **Corrupt data structures**: for example, if a database row spans more than one page and the pointer that links the first page to the second is damaged or missing, there is a corrupt data structure. InterBase/Firebird is able to correct this situation, but the damaged row might be lost.
- **Misallocated data pages**: for example, a page can be used for transaction inventory, header information, data, blob pointers, or indices. If a page has been flagged as one type, but actually stores data of a another type, InterBase/Firebird detects the problem. However InterBase/Firebird cannot recover from this type of problem, so it will probably be necessary to restore from a backup.
- **Orphaned data pages**, which are automatically returned to the free space pool. By default, InterBase/Firebird does not completely fill data pages with records, to allow space for new records to be quickly inserted. As records are added and deleted, some pages are likely to end up with no active records on them. Older InterBase/Firebird versions do not automatically reallocate these pages to the free space pool.

The IBExpert Database Validation menu item offers those options also available in the InterBase/Firebird GFIX.

It is advisable to backup the database before validating. If possible it should also be shut down, so that the backup can be restored if necessary without any loss of transactions which may have been performed since the backup.

The Database Validation menu item can be found in the Services menu. It enables the database to be validated and verifies the integrity of data structures.



First select the registered database to be validated. The following options are none other than the GFIX parameters and may be specified as wished:

Limbo Transactions: If this option is checked, the database is checked for transactions in limbo, i.e. transactions, that can't be defined as executed or aborted. Please refer to transactions in limbo for further information.

Check Database: This option validates the database, but doesn't repair it.

Ignore Checksums: This option ignores all checksum errors. A checksum is a page-by-page analysis of data to verify its integrity. A bad checksum means that a database page has been randomly overwritten (for example, due to a system crash).

Kill Shadows: This option kills all unavailable shadow files.

Mend Database: This prepares a corrupt database for backup and repairs any database corruption if possible.

Sweep Database: This option can be checked to perform a database sweep (see database sweep for more information about sweeps).

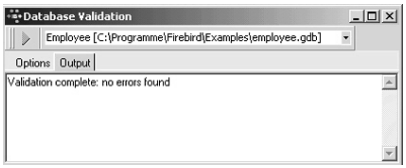
Validate Database: (default value). This option validates the database structure.

Validate Full: This validates record fragments. *Note*: This feature is not available in InterBase versions older than the version 6.

Output: Check Verbose to receive an extended report about the current database validation process. Select whether this report should be displayed on screen or saved to file (not forgetting of course to specify drive, path and file name).

Then start the database validation using the green arrow icon or [F9].

Output:



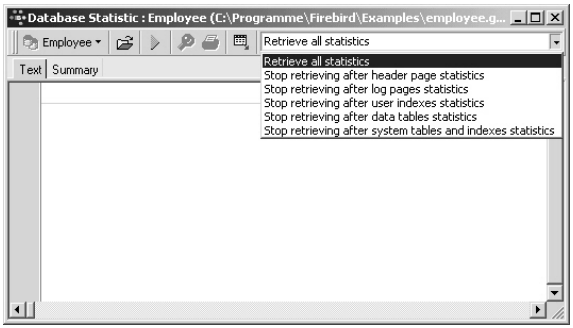
If no corruption is detected, a message is displayed informing that no database validation errors were detected. If corruption is detected that can be repaired, a report is displayed showing the number and types of errors found. Note that sometimes, irreparable database corruption is found, such as damage to the database header or space allocation tables.

Please refer to Database Corruption for further information concerning the recovery of corrupt databases.

9.6 Database Statistics

The IBExpert Database Statistics retrieves and displays important database statistical information, which can be exported to numerous file formats or printed. This menu item can be found in the IBExpert Services menu.

First select a registered database from the pull-down list on the toolbar, or alternatively open an existing statistics file to view and analyze.



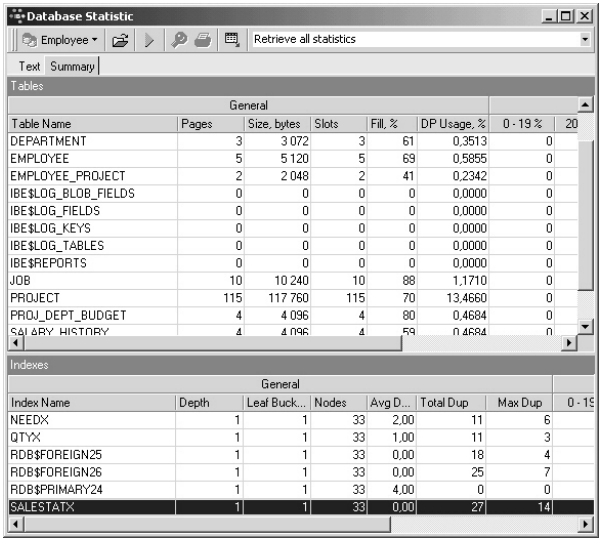
If wished, alter the default value *Retrieve all Statistics*, by selecting one of the following options:

- Stop retrieving after header page statistics
- Stop retrieving after log page statistics
- Stop retrieving after user indexes statistics
- Stop retrieving after data tables statistics
- Stop retrieving after system tables and indexes statistics

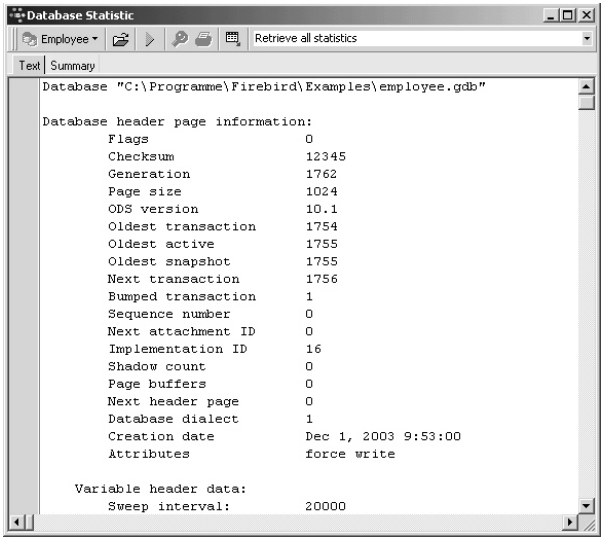
Since IBExpert version 2004.8.5 there is the added option to analyze average record and version length (Firebird 1.5, InterBase 7). Simply check the box below the toolbar.

Then simply click the Retrieve Statistics icon (green arrow) or press [F9] to start the retrieval process.

The database's statistical summary is displayed both in grid form:



as well as text:



The following information is displayed for all tables in the database: table name, pages, size (bytes), slots, fill (%), DP usage (%) and fill distribution (an optimal page fill is around 80%).

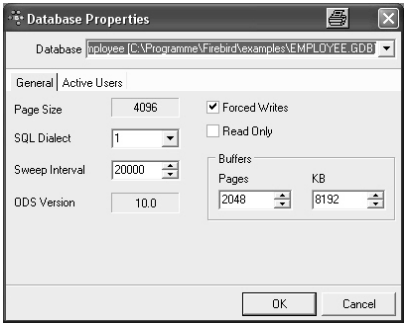
Below the table grid, an index grid displays the statistics for all indices for a selected table. The following information is displayed for indices: index name, fields, unique, active, sorting order, statistics, depth, leaf buckets, nodes, average data length, total dup and fill distribution.

The text summary provides certain additional information (see illustration above) as well as a statistical summary broken down by table, containing the information already mentioned in the grid summary.

This information can be exported (see Export Data) to save the information to file, or printed out.

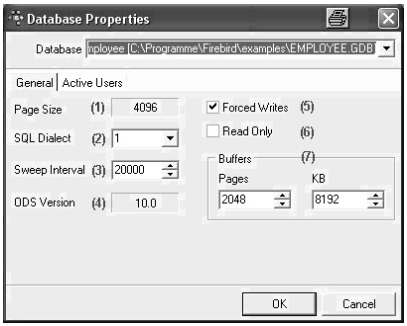
9.7 Database Properties

The Database Properties Editor can be started from the IBExpert Services menu. It can be used to specify certain properties and view others appertaining to the database specified in the Database pull-down list (in the upper part of the editor).



There are two tabs labeling the General page and the Active Users page.

9.7.1 General



The General page displays the following information for the selected database:

(1) Page Size: displays the current specified page size. The page size can only be altered by performing a database backup followed by a restore (IBExpert menu: Services / Restore Database) and redefining the database page size.

(2) SQL Dialect: shows which SQL dialect was specified at the time of database registration. This may be altered here, if wished (although watch out for possible dialect incongruencies, for example, the different date and time types).

(3) Sweep Interval: This displays the number of transactions which may be made in the database before an automatic garbage collection is executed by InterBase/Firebird. If this number is specified at zero (0) it is not performed automatically at all. It could then be carried out, for example, at night as a sweep or backup using GFIX and the `at` Windows command or the Linux `chron` command. Please refer to database sweep for further information.

(4) ODS Version: The ODS (= On-Disk Structure) version shows with which database version the database was created, e.g. InterBase 5 = ODS version 9, InterBase 6 = ODS version 10.0, InterBase 6.5 = ODS version 10.1, InterBase 7 = ODS version 11. Firebird versions start at ODS version 10.0.

(5) Forced Writes: This enables the forced writing onto disk mode. when committing. Please refer to forced writes for further information.

(6) Read Only: A database can be set to *Read Only* when, for example, saving the database onto a CD, or in the case of a reference or archive database. The *Read Only* property is forced in the TIP page, by preventing all insert, alter and delete commands.

(7) Buffers: Here it is possible to specify how much cache the database server should reserve. A good number of buffer pages is 10,000 (based on a 4K page size to allow 40MB cache).

The amount of buffers/cache reserved can be viewed in IBExpert here (default = 2,048). If this is increased the database can load considerably more pages. Please refer to buffers for details.

Buffers

The buffers/cache can be set using the IBExpert menu item Database Properties, found in the Services menu, or using the command-line utility GFIX. The amount of buffers/cache reserved can be viewed in IBExpert under Services / Database Properties. The IBExpert Performance Analysis also displays the number of data pages that are being held as cache on the server (from InterBase 6 onwards the standard is 2,048). This can be altered for the current database if wished.

If this is increased the database can load considerably more pages. For instance, it is much more efficient to load 10,000 pages, than loading 2,000 and then exchanging for new pages once the 2,000 have been loaded. The only limit to amount of cache is the physical size of the RAM (e.g. 10,000 x 4K page size). The total KB is calculated according to the current database page size. For an alteration to become effective, it is therefore necessary for all users to disconnect from the database and then reconnect. Buffers are only reserved if they are really necessary.

Database sweep / sweep interval

When a database is swept, all old invalid data is removed from the data pages, thus reducing the total size of the database and making room for new data sets.

A database sweep performs a garbage collection in the database, and is performed automatically during a database backup or when a SELECT query is made on a table (and not by INSERT; ALTER or DELETE). Furthermore database sweeps are, as standard, executed automatically after every 20,000 operations. With very consistent databases however a database sweep can be started unnecessarily and thus cost unnecessary performance losses during normal user processing. The default database sweep interval value of 20,000 (operations) can be overwritten using the IBExpert Services menu item Database Properties.

Under Database Properties / Sweep Interval the number of operations can be specified before a database sweep should be automatically performed. A database sweep or backup can be performed during 24 hour operation online without any problems (i.e. the server does not need to be shut down). This however does slow performance during the sweep which may not be desired.

If the sweep interval is specified at zero (0) it is not performed automatically at all. It could then be performed explicitly, for example, at night as a sweep or backup using GFIX and the at Windows command or the Linux chron command.

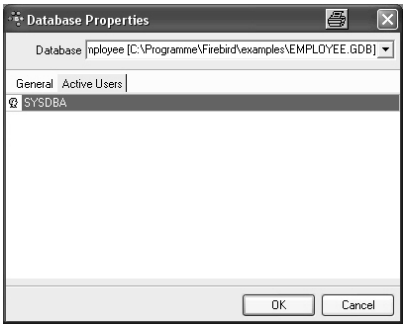
Forced writes

This enables the forced writing mode on disk. If the forced writes option is selected all data is saved immediately to disk, i.e. every time a commit is made everything is written to the hard drive, and then to the TIP (=Transactions Inventory Page).

Without forced writes the process is minimally quicker, but when working on a Windows platform, Windows decides what should be saved to file, where and when, and the data pages are saved to file last i.e. the TIP changes are written first, and then the data sets - which could possibly lead to inconsistencies. Firebird 1.5 has however made a number of improvements in this area, so that using forced writes for this reason is no longer quite so important.

Using forced writes is therefore recommendable in the case of instable systems (e.g. laptops). In normal circumstances with a secure system however, it should not be necessary to activate it.

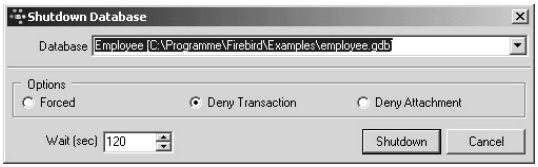
9.7.2 Active Users



This page displays those users logged in to the current database with an open attachment. If an application has several attachments, or a single user is connected more than once, this is also visible here. This is important should the database need to be shut down at short notice.

9.8 Database Shutdown

There are a few occasions when a database needs to be shut down. For example, when a new foreign key needs to be inserted the database should be shut down in order to avoid the annoying message "Object in use". A registered database can be shut down simply and quickly using the IBExpert Services menu item Database Shutdown.



Select the registered database which is to be shut down. Then select one of the following options, to specify how active transactions should be dealt with:

Forced: In this mode all transactions, that are still active at the stated time, are aborted regardless of their type or importance, and all users are forcefully disconnected. As InterBase/Firebird transactions function stably and securely, there are very few areas of application where this forced mode should not be used.

Deny new transactions: In this mode all transactions must be executed by the stated time. Any new transactions that are started are blocked. If there are any transactions that are still active by the stated time, the database shutdown is not executed.

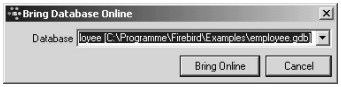
Deny new attachments: With this option all active user attachments must finish their work by the stated time. If some attachments are still active by the stated time, the database shutdown is not executed.

Wait: The period of time (in seconds) until the shutdown is executed can be specified here.

Then simply click *Shutdown* to shutdown the database. To bring the database back online, choose the IBExpert Services menu item Database Online.

9.9 Database Online

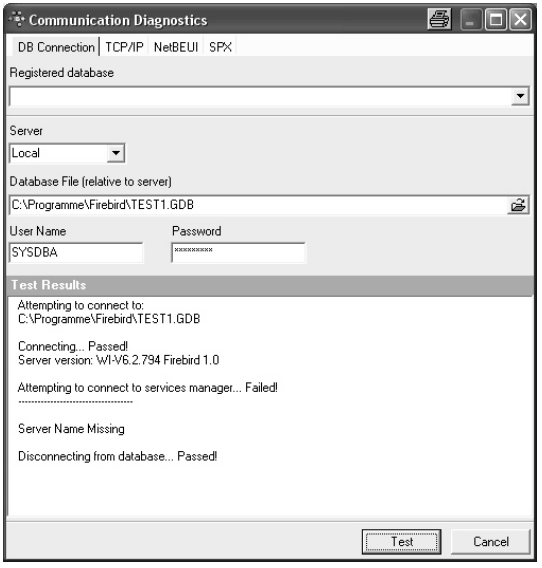
The IBExpert Service menu item Database Online is used to bring a database back online again after it has been shut down (please refer to Database Shutdown for further information).



Simply select a registered database and bring the database online.

9.10 Communication Diagnostics

The Communication Diagnostics dialog can be started from the IBExpert Services menu. It also appears automatically when registering a database and the *Test Connect* button is pressed. IBExpert's Communication Diagnostics delivers a detailed protocol of the test connect to a registered InterBase/Firebird server and the results:



This is particularly useful when attempting to connect to a remote database server, as detailed status information concerning the various steps taken to make the connection is displayed, indicating problem areas if the connection is not achieved. If using an alias path for a remote connection, please refer to the article "Remote database connect using an alias".

The following protocols are supported:

- TCP/IP (worldwide standard)
- SPX - which used to be used by Novell; now even Novell supports TCP/IP. a
- NetBEUI - which is not really a network protocol, it simply accesses the line. It is slow as it makes everything available everywhere and anyone can access the information. This is also purely a Windows protocol.

Should problems occur, switch to the relevant protocol page and test again.

The TCP/IP protocol offers the following services:

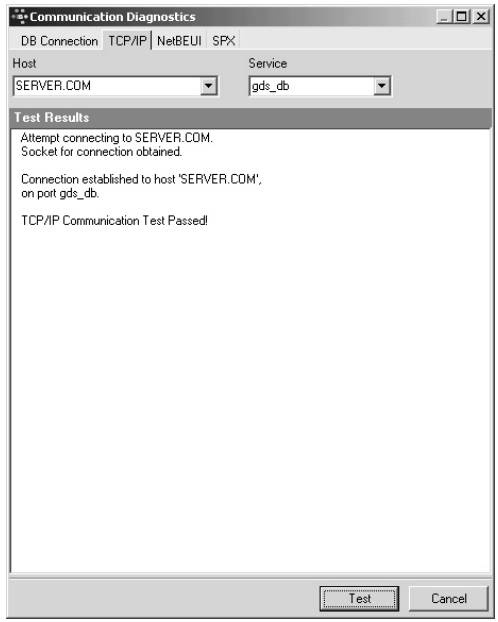
21 and FTP: Each port receives a name. With Firebird this is actually optional, with InterBase: `Win\System32\ drivers\etc\services -> ftp` (= the name for-) `21/tcp`.

3050: This is the standard port for InterBase and Firebird. However this is sometimes altered for obvious reasons of security, or when other databases are already using this port. If a different port is to be used for the InterBase/Firebird connection, the port number needs to be included as part of the server name. For example, if port number 3055 is to be used, the server name is `SERVER/3055`.

gds_db: For InterBase: `name = gds_db = 3050 / tcp` (a different port to the standard 3050 can be specified if wished). If this entry is nonexistent Firebird does not care; InterBase however does! The name `gds_db` has to be present.

Ping: can be used if the connection was unsuccessful and the reason is not known. This DOS command checks which input is correct, and works regardless of whether `InterBase.exe` or `Firebird.exe` is installed. The results show whether a database has been found, and at which address. This should, as a rule, always work unless of course the server uses a Firewall which does not allow a Ping to be answered. In this case, use the service FTP (as a rule the same as the 21 service).

Note: in DOS the `TRACERT` command lists the protocol route. TCP/IP intelligently takes another direction if one or part of the lines on the quickest route is blocked or down.


Communication Diagnostics
DB Connection
TCP/IP
NetBEUI
SPX
Host
SERVER.COM
Service
gds_db
Test Results
Attempt connecting to SERVER.COM.
Socket for connection obtained.
Connection established to host 'SERVER.COM',
on port gds_db.
TCP/IP Communication Test Passed!
Test
Cancel

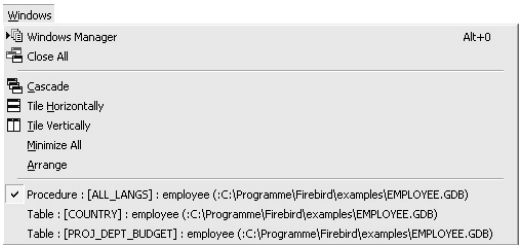
10 IBExpert PlugIns Menu

The IBExpert PlugIns menu is intended for user-specified menu items for third party components. Two Delphi PlugIn examples are delivered as part of IBExpert and can be found in the IBExpert/PlugIn directory. Should you have problems finding these files they can also be downloaded free of charge from the web: www.ibexpert.com/download (a direct link can be found under IBExpert Help /IBExpert Direct). You need to have Delphi, InterBase or Firebird and, of course, IBExpert installed.

Installation of the components is explained in detail in the `Readme.txt` files enclosed.

11 IBExpert Windows Menu

The IBExpert Windows menu offers a number of options to visually arrange all open windows in IBExpert.



Please note that all open windows are also displayed as buttons on the Windows bar (directly above the status bar), and in the DB Explorer on the Windows page (please refer to Windows Manager for further information).

11.1 Windows Manager

The DB Explorer Windows Manager can be opened using the IBExpert Windows menu item Windows Manager, by using the key combination [Alt + O], or simply by clicking on the Window tab heading directly in the DB Explorer.

For more information regarding this, please refer to DB Explorer / Windows Manager.

11.2 Close All

Close All is an option to close all open windows with one simple mouse click, ideal when closing all open work for one project or database, before beginning work on a new project or database, or finally finishing work for the day (...or night!).

11.3 Cascade / Tile / Minimize / Arrange

The IBExpert Windows menu offers the following options, for arranging all open windows:

- **Cascade** - all open windows are arranged one behind the other, in a cascading format, displaying the title bar of each window.
- **Tile Horizontally** - all open windows are displayed adjacently, one below the other.
- **Tile Vertically** - all open windows are displayed adjacently, one next to the other.
- **Minimize All** - this option minimizes all open windows simply and quickly with a single mouse click.
- **Arrange** - this option arranges the windows as currently viewed, e.g. all minimized windows are arranged in a horizontal row alongside each other.

If the SDI User Interface has been specified under Environment Options / User Interface, then only the Cascade option is offered here.

12 IBExpert Help

The IBExpert Help Menu offers a number of provisions to offer support for IBExpert.

Since IBExpert version 2004.2.26.1, there is a new context-sensitive help system. Pressing [F1] in any of the IBExpert forms now opens a new web-based Help page. It is also possible to download all Help Pages from http://www.ibexpert.info/documentation/documentation.zip and unzip this in the IBExpert main directory with subdirectories (there must be a new subdirectory called documentation). If a local Help document is available, it will be opened in the browser. Otherwise the browser will open the page from our web server. If you have any comments or questions please use our newsgroup (please see below).

The complete help files (beta version 1.1) are also available directly online: http://www.ibexpert.info/documentation/

The first view displays the Help structure. If you are looking for help about a specific subject use the SEARCH function, or the index.

To integrate the online Help Files into IBExpert itself, follow these five steps:

- Download the help file (http://www.ibexpert.info/documentation/documentation.zip)
- If you have an older version of IBExpert, delete the Help directory.
- Create a new directory: `Documentation` in the IBExpert main directory.
- Extract and copy the documentation.zip file into the `IBExpert\Documentation` directory.
- When you start IBExpert and press [F1] from any dialog, the DB Explorer or the SQL Assistant, it will open an html file in `C:\program files\HK-Software\IBExpert 2.0\Documentation\helpcontext` showing you the relevant help information.

Should you not be able to find a solution to your problem here, please use one of our newsgroups:

Username: **ibexpert**
Password: **ibexpert**

news://ibexpert.info/interbase.ibexpert.de German language
news://ibexpert.info/interbase.ibexpert.en English language
news://ibexpert.info/interbase.ibexpert.ru Russian language
news://ibexpert.info/interbase.ibexpert.fr French language

or send us an email to support@ibexpert.com or use our Bug Track System in the IBExpert Help Menu.

Should you have any comments or queries directly regarding the Help documentation, or wish to contribute you own articles, please contact documentation@ibexpert.com

Please also refer to our series of online tutorial films, demonstrating the more important IBExpert features, found on our website www.ibexpert.com on the DEMO page.

12.1 IBExpert Customer Area

New to IBExpert version 2005.3.12.1, this menu item allows all registered users of full versions (not the Trial Version or Personal Edition) direct access to the protected customer area, without having to search for their current registration keys.

Simply click the menu item, and IBExpert uses your registration keys to automatically access the online IBExpert Customer Area. This does nothing other than open a URL such as the following example:

http://1234567887654321:ibexpert@www.ibexpert.com/customer

where 1234567887654321 is a combination of Key A and Key B which is already stored in the registry. (There is no point testing the above link, as the keys quoted are for example only!).

Warning: Although this function works faultlessly with browsers such as Firefox, problems may be experienced with Windows Internet Explorer. In this case, it is necessary to access the protected customer area under http://www.ibexpert.com/download/ in the usual way, by inputting your customer keys and password, and then download the `customer_area.reg` to the local drive and then merge in regedit (Windows menu Start / Execute; type regedit, right-click menu item Merge and merge the files) .

Alternatively it is possible to create the following registry key manually:

- In Windows click the bottom left menu Start.
- Execute.
- Type "regedit" and enter (or click OK).
- HKEY_LOCAL_MACHINE ist the root-key. Open the folders, SOFTWARE, Microsoft, Internet Explorer, Main and FeatureControl.
- Here you need to add a new feature "FeatureControl".
- You should then add FEATURE_HTTP_USERNAME_PASSWORD_DISABLE and using the right-click menu in the empty right dialog area, select New and then Key, and type "`IExplore.exe`" in the input field.
- On the left you will now find a new folder, `IExplore.exe`, in the FeatureControl list. Highlight this, use the context-sensitive right-click menu to select New / DWORD value.
- Add new DWORD with name "`IExplore.exe`" and value "" (`"IExplore.exe"=dword:00000000`).

12.2 What Is New?

IBExpert 2005.03.12

1. Blob Editor:

- Added support for PNG (Portable Network Graphics) images.

2. Script Executive:

- Executing of `INSERT/UPDATE/EXECUTE PROCEDURE` statements `WITHOUT` parameters is up to 10 times faster now.

- Added support for following Firebird 2 features:

```
CREATE SEQUENCE
DROP SEQUENCE
ALTER SEQUENCE
```

- Fixed problem with the occasional hanging of the Script Executive.
- Extended syntax of `OUTPUT` command:

```
1.
output 'E:\data.sql'
  as insert into mytable commit after 1000;
  select * from IBE$$TEST_DATA where F_INTEGER < 3000;
  output;

2.
output 'E:\data.sql'
  as reinsert into mytable
  commit after 2000;
  select * from IBE$$TEST_DATA where F_INTEGER < 3000;
  output;

3.
output 'E:\data.sql'
  as execute procedure myproc;
  select * from IBE$$TEST_DATA where F_INTEGER < 3000;
  output;
```

`ASINSERT` option is available for compatibility.

3. Table Data Comparer:

- Now works up to 5 times faster. Many thanks to Nickolay Samofatov for useful hints and suggestions.

4. Database Comparer:

- Fixed some problems while comparing scripts with following declarations:

```
DECLARE VARIABLE MyVar CHAR;
WHERE ... CONTAINING
FOR EXECUTE STATEMENT ...
EXECUTE STATEMENT ... INTO ...
```

5. SP/Trigger Editor, SP Debugger:

- Added support for following Firebird 2 features:

```
DECLARE <cursor_name> CURSOR FOR ...
OPEN <cursor_name>
FETCH <cursor_name> INTO ...
CLOSE <cursor_name>
```

```
LEAVE <label>
NEXT VALUE FOR <generator>
```

6. SQL Editor:

- Fixed problem with error message display when executing `INSERTEX` statements.

7. Data Grid, Input Parameters Window:

- Fixed problem with input of date values when the system date format is `dd-MMM-yy`.

8. Database Designer:

- Added possibility to lock/unlock visual objects to protect them against casual modification of size and position. Use objects context-sensitive (i.e. right-click) menu to lock/unlock them.

9. IBEBlock:

- `SELECT ... EXPORT AS ...` implemented. Examples of usage:

```
1.
SELECT * FROM RDB$FIELDS
EXPORT AS HTML INTO 'E:\TestExport.html'
OPTIONS 'ColorShema=MSMoney; FontFace=Verdana';
```

Possible `ColorShemes` are `BW, Classic, ColorFull, Gray, MSMoney, Murky, Olive, Plain, Simple`.

```
2.
SELECT * FROM RDB$FIELDS
EXPORT AS XLS INTO 'E:\TestExport.xls' OPTIONS '';

3.
SELECT * FROM RDB$FIELDS
EXPORT AS TXT INTO 'E:\TestExport.txt'
OPTIONS 'OmitCaptions';

4.
SELECT * FROM RDB$FIELDS
EXPORT AS CSV INTO 'E:\TestExport.txt'
OPTIONS 'OmitCaptions; Delimiter=";"';

5.
SELECT * FROM RDB$FIELDS
EXPORT AS XML INTO 'E:\TestExport.xml'
OPTIONS 'Encoding=windows-1251; MemoAsText; StringAsText';
```

- `FOR ... DO` loops implemented. Examples of usage:

```
EXECUTE IBEBLOCK
RETURNS (I INTEGER)
```

```
AS
BEGIN
  FOR I = 0 TO 100 DO
    SUSPEND;
END
```

It is possible to use CONTINUE statement within FOR loop to proceed to the next iteration of FOR:

```
EXECUTE IBEBLOCK
RETURNS (I INTEGER)
AS
BEGIN
  FOR I = 0 TO 100 DO
  BEGIN
    IF (I < 20) THEN
      CONTINUE; -- SUSPEND will not be executed
    SUSPEND;
  END
END
```

- EXECUTE IBEBLOCK statement implemented. Using this statement you can call other IBEBlocks from the main block. Examples of usage:

```
1.
EXECUTE IBEBLOCK
AS
BEGIN
  ...
  MyFunc = 'EXECUTE IBEBLOCK (
              IntVal INTEGER)
            RETURNS (
              Square INTEGER)
            AS
            BEGIN
              Square = IntVal * IntVal;
            END';
  EXECUTE IBEBLOCK MyFunc (2) RETURNING_VALUES :Square;
  ...
END
```

```
2.
EXECUTE IBEBLOCK

AS
BEGIN
 ...
  MyFunc = ibec_LoadFromFile('C:\MyBlocks\Square.ibeblock');
  EXECUTE IBEBLOCK MyFunc (2) RETURNING_VALUES :Square;
  ...END
```

- Default values and comments for input/output parameters and variables implemented. Example:

```
EXECUTE IBEBLOCK (

  CodeDir VARCHAR(1000) = 'C:\MyBlocks\' COMMENT 'Path to my IBEBlocks',
  SQLDialect INTEGER = 3 COMMENT 'Database SQL Dialect')
RETURNS (
  TotalTime DOUBLE PRECISION = 0 COMMENT 'Total time spent')
AS
DECLARE VARIABLE MyVar INTEGER = 0 COMMENT 'Just a comment'
BEGIN
 ...END
```

Comments for input parameters will be displayed in `Description` column of `Request Input Parameters` form.
Comments for output variables will be used as column captions of the result dataset.
Comments for local variables are ignored.

- New examples added:

ODBC Access
Table Data Comparer using cursors
User forms in IBEBlock

Use following URL to download IBEBlock examples:
www.ibexpert.com/download/ibeblockex.zip

10. All console tools (IBEScript, IBECompare, IBEExtract) were updated. Current version is 2005.3.12.

11. Many other bug fixes and small improvements...

12. Distribution of IBExpert Modules

- To be allowed to distribute any of the IBExpert Modules (`ibexpert.exe`, `ibescript.exe`, `ibescript.dll`, `ibeextract.exe` and `ibecompare.exe`) together with your application, you need:

IBExpert Site License, if the distribution is located only on computers in your own company.
IBExpert VAR License, if the distribution is located on any computer outside your company.

If you are already an IBExpert customer, you can upgrade to a Site or VAR License and directly buy the 24 month Extension Product.

See www.ibexpert.com Purchase Area for Details.

Some functions of the new IBExpert Modules do not work on non-licensed computers, so you can only use them where your IBExpert License is valid.

Customers with a Site License are allowed to use them on every computer in their company. Simply copy the License file to the path, where the module (such as `ibescript.exe`) should run.

VAR License customers may also integrate these modules and the License file in their Software installation.

IBExpert 2005.02.12

1. SQL Editor:

- Added support for the `INSERTEX` command (importing data from a comma-separated values file).

2. Database Comparer:

- Fixed problem with incorrect updating of computed fields.
- Fixed problem with missing descriptions of view fields after recreating a view.

3. User Manager:

- Fixed problem with changing of password.
- Added Refresh button to refresh a list of users.

4. Database Designer:

- Fixed problem with missing `AS` keyword in SP/trigger bodies. It will be inserted automatically if it doesn't exist.

5. Log Manager:

- Now uses 64-bit ID's when working with SQL Dialect 3 databases.

6. Table Editor:

- Fixed problem with the maximum length of constraint names (was 27, now expanded to 31).

7. Editor Options:

- Added possibility to disable Code Insight while editing an object description.
- Added possibility to customize color of IBEBlock function names.

8. IBExpert Main Window:

- There was a problem with a second instance of IBExpert when one was disabled in options: it remained in the memory. It's fixed.

9. Services | Database Monitoring (InterBase 7.x):

- Added possibility to define different monitor queries for SQL Dialect 1 and SQL Dialect 3.

- Added possibility to commit/rollback transactions, shutdown attachments and cancel statements.

10. Data Grid:

- Fixed problem with the display of empty blob field values. Now IBExpert displays them as <NULL>.
- Added possibility to recalculate the number of filtered records automatically when the filter condition is changed.

11. Extract Metadata:

- Fixed problem with occasional incorrect script file names when "Limit file size" option is enabled.
- Fixed "out of memory" problem when extracting metadata of a database with several thousand triggers.
- Fixed problem with extracting charset of UDF parameters.

12. UDF Editor

- Fixed problem with parameter charsets.

13. SP Debugger:

- Fixed problem with restoring of debugger window properties.

14. EXECUTE IBEBLOCK:

- Fixed some problems with incorrect handling of BLOB and NUMERIC values.
- Added support for ROW_COUNT/ROWS_AFFECTED variables.
- Additional functions and examples were added. More info about IBEBlock may be found here: www.ibexpert.info/documentation

15. Code Insight:

- Now also supports IBEBlock constants and functions.

16. Environment Options | Tools:

- Added "Revoke existing privileges before autogranting" option. If this is enabled, existing privileges of an object (SP, trigger, view) will be deleted before granting it new privileges.

17. Many other bug fixes and small improvements...

18. Distribution of IBExpert Modules

- To be allowed to distribute any of the IBExpert modules (ibexpert.exe, ibescript.exe, ibescript.dll, ibeextract.exe and ibecompare.exe) together with your application, you need:
- an IBExpert Site License, if the distribution is located only on computers in your own company

- an IBExpert VAR License, if the distribution is located on any computer outside your company

If you are already an IBExpert customer, you can upgrade to Site or VAR License and purchase the 24 month Extension Product.

See http://www.ibexpert.com/ Purchase Area for details.

IBExpert 2004.12.12

1. IBEScript.dll (for registered customers only

For registered customers we've included the `IBEScript.dll` in the installation archive. You can use it in your applications to execute scripts from file or from a string buffer. There is a small demo application illustrating its use in the IBEScriptDll folder.. Please also refer to the IBEScriptDll Readme.txt.

To be allowed to distribute any of the IBExpert Modules (`ibexpert.exe`, `ibescript.exe`, `ibescript.dll`, `ibeextract.exe` and `ibecompare.exe`) together with your application, you need:

- IBExpert Site License, if the distribution is located only on computers in your own company.
- IBExpert VAR License, if the distribution is located on any computer outside your company.

If you are already an IBExpert customer, you can upgrade to a Site or VAR License and purchase the 24 month Extension Product.

See www.ibexpert.com PURCHASE area for details.

2. Table Editor:

- Added support for the InterBase 7.5 temporary tables feature.

3. SQL Editor:

- Fixed the problem with missing columns/parameters/variables while creating an SP/View from `SELECT`.

4. Table Data Comparer:

- The problem with the incorrect `WHERE` clause when a primary key consists of more than one column has been fixed.

5. Extract Metadata:

- Fixed the problem with the incorrect `GRANT` statement when there are `UP-DATE/REFERENCE` privileges on separate columns.
- The problem with incorrect file names when the *Limit File Size* option used has now been solved.
- Added support for the InterBase 7.5 temporary tables feature.

6. Database Explorer:

- Recent page: fixed the problem with sorting on *Last Used* column if the system short date format is `dd-MMM-yy`.
- Added support for InterBase 7.5 embedded user authentication. There is now a separate node for embedded users in the Database Explorer. It is possible to create, alter and delete embedded users using the DB Explorer context menu.
- Added option to activate/deactivate only selected procedures/triggers. Just select the required SP/triggers holding the [Ctrl + Shift] keys and choose the *Deactivate/Activate* item in the DB Explorer context menu.

7. User Manager:

- Added support for the InterBase 7.5 embedded user authentication.

8. Database Designer:

- Reverse Engineering. Added *Do not remove foreign keys marked as non-Generate* option. It is useful to prevent fake relationships from being deleted.
- Fixed the problem with saving modifications when adding a column with `Generate = FALSE`.
- Added the option to drag 'n' drop objects from the DB Explorer (Diagrams page) to subject areas to include them as members of this area. It is also possible to drag objects from the list of objects in the Subject Areas Manager.
- The problem with enabling the Save button following modification of generators and exceptions has been fixed.
- Generation of the *Update* script has been improved. Now IBExpert analyzes exceptions and procedures as well. Fixed the problem with updating the description of triggers.
- Fixed the problem with modification of *Notes* when the properties panel is undocked.
- The problem concerning the incorrect creation of primary key names has been fixed.

9. IBEExtract:

- Fixed the problem with quoting identifiers that correspond to the keywords.

10. Log Manager:

- Sometimes the bodies of logging triggers were empty. It's now fixed.
- Added the possibility to generate a log script for several tables simultaneously (Log Data tab / Log to Script). Just select the required tables holding the [Ctrl + Shift] keys.

11. EXECUTE IBEBLOCK:

- Fixed problem with using blob values in DML statements.

12. Console tools:

- IBEScript, IBEExtract and IBECompare were updated. Current version of all console tools is 2004.12.12.

13. Other minor bug fixes and small improvements...

IBExpert 2004.10.30

1. NEW: OLAP and Data Warehouse Technology in IBExpert:

Here is a brief description how to use the new IBExpert Data Analysis. Please refer to IBExpert Tools menu / Data Analysis for a more detailed documentation.

- Connect IBExpert to the `Employee` example database.
- Run the following SQL Statement `SELECT * FROM SALES`.
- On the Result Page click the Data Analysis Tool button.
- From left field lists choose `Cust_No` and drag ´n' drop it in the Dimensions.
- Select `Sales_Rep` and drag ´n' drop it in the Dimensions.
- Select `Ship_Date` and drag ´n' drop it in the Dimensions, but change *Alias Name* and *Display Name* to `Ship_Date_Month` and change *Wrap To* to Month.
- Select `Total_Value` and drag ´n' drop it in the Measures.
- Click the Build Cube icon or [F9].
- Now you can drag ´n' drop all Dimensions to the Columns Area on the top or in the Rows Area on the left.

To see what else is possible, take a look at Data Analysis or refer to http://www.pivotcube.com/.

We also plan to create a free runtime version where `*.cub` files can be opened also outside IBExpert.

So stay tuned ...

(this functionality is not available in the Personal Edition.)

2. Fixed the problem with incorrect support for XP themes that cause "list index of bounds (x)" error.

3. Data Grids:

Fixed the problem with including computed fields into `INSERT/UPDATE` statements using "Copy record as `INSERT/UPDATE`".

4. Table Editor:

There was a problem with adding/altering field descriptions introduced in 2004.09.12. It's fixed.

5. Database Designer:

- Sometimes there was an AV while creating an update script. It's fixed.
- IBExpert now uses templates (Options / Templates) to create foreign key names and names of check constraints.

6. IBEScript was updated. Current version is 2.1

7. IBEExtract was updated. Current version is 2.03

8. Some improvements were made to make initial databases opening faster while working with slow internet connections.

9. 50 MB Bug in Personal Edition is removed

10. Some other minor bug fixes and improvements...

IBExpert 2004.9.12

1. Since this version IBExpert stores all its files (`IBExpert.stg`, `IBExpert.tb`, `IBExpert.scm` etc.) in the user home directory (for example, `C:\Documents and Settings\<User_Name>\Application Data\HK-Software\IBExpert\`). Existing files will be moved to this directory automatically when the new version is started for the first time.

2. Data Grids:

- Added support for UNICODE. It is possible to display data in unicode, but there is no possibility to edit it directly in the grid. To edit data in unicode use the Form View or the modal editor connected with string cell. To display data as unicode click the *Display data as Unicode* button on the toolbar above the data grid or press F3. This feature is unfortunately not available in the IBExpert Personal Edition.
- The Form View has been completely redesigned. It now also displays field descriptions. It is also possible to select alternative layouts (classic or compact), the compact alternative for those who prefer a more compact and faster interface. Visual options now also include specification of Memo Height and Memo Word Wrap.

3. Database Explorer:

A separate node has been added for database indices. It is also possible to display system indices (indices for system tables). Use the IBExpert menu item Database Registration Info / DB Explorer / Additional / Show System Indices to enable/disable the display of system indices.

4. SP/Trigger Debugger:

There is now the added feature allowing initialization of Parameters/Variables using values in any data grid. Just drag and drop a cell value from any data grid onto the corresponding node in the Parameters/Variables list to initialize the variable with the value of the data cell. You can initialize multiple variables/parameters by holding the [Ctrl] key when dropping. In this case IBExpert searches for the corresponding parameter/variable (by name) for each field in the data record, and if the parameter/variable is found it will be initialized with the value of the field with the same name.

5. Database Designer:

Added Model Navigator to navigate models quickly. Use the corresponding tab in the SQL Assistant (Model Navigator) and of the Database Explorer (Diagrams).

SET NAMES, SET SQL DIALECT, CREATE DATABASE statements were removed from resulting CREATE DATABASE script. Use the model Prescript to specify necessary init statements.

6. IBExpert Database Menu:

Added *Recreate Database* feature. This drops the database, along with all its contents, and creates it again (after confirmation, of course) using the parameters of the database just dropped.

7. Extract Metadata:

- Additional mode added: Extract into separate files. (Please do not confuse it with the mode of the same name in previous versions of IBExpert; this was renamed "VCS files"!). The Separate Files mode extracts metadata (and data, if specified) into a set of files: two files with metadata (`_ibe$start_.sql` and `_ibe$finish_.sql`), files containing table data (one or more files for each database table) and a `runme.sql` file, that consists of a number of `INPUT <file_name>` statements in the correct order.
- A *Limit File Size* option has been added. This defines the maximum file size of the resulting script(s). When specified, and the maximum size is reached, IBExpert automatically creates the next file with suffixes 0001, 0002 etc.

8. IBEExtract:

- Added `-f` option (extract into separate files).
- Added `-z` option (maximum size of resulting files (in megabytes).
- The current version is 2.02.

9. IBEScript:

- Added `-i` options (idle priority).
- Support for EXECUTE IBEBLOCK implemented (unfortunately not available in the free version of IBEScript).
- Current version is 2.02.

10. IBECompare was updated to 1.45

11. EXECUTE IBEBLOCK:
In this version we introduce to you a new powerful feature - EXECUTE IBEBLOCK. What is IBEBLOCK? It is a set of DDL, DML and other statements that are executed on the server and on the client side and include some specific constructions applicable only in IBExpert or IBEScript (excluding free versions of products).

With EXECUTE IBEBLOCK you will be able:

- To work with different connections within the single IBEBLOCK at the same time.
- Move (copy) data from one database to another.
- Join tables from different databases.

- Compare data from different databases and synchronize them.
- Populate a table with test data using random values or values from other tables or even from other databases.
- ... and much more!

The syntax of IBEBLOCK is similar to that of stored procedures, but there are many important extensions. For example:

- You can use EXECUTE STATEMENT with any server, including InterBase 5.x, 6.x, 7.x.
- You can use one-dimensional arrays (lists) of untyped variables and access them by index.
- It isn't necessary to declare variables before using them.
- You can use data sets (temporary memory tables) to store data.
- ... and much more!

You can debug IBEBLOCKs in the same way as stored procedures and triggers.

IBEScript supports EXECUTE IBEBLOCK too.

For examples of using EXECUTE IBEBLOCK, please refer to IBEBLOCK Examples.

12. Changed Installer:

Now only the full Install version is uploaded, all tools from the old zip archives are installed with this installer in the IBExpert main directory.

13. Many minor bug fixes and small improvements...

IBExpert 2004.8.5

1. Database Designer:

- Now processes generators and triggers while generating the update database script.
- It also takes into account view dependencies while creating the script.
- The problem with incorrect definition of integer fields has been fixed (sometimes IBExpert included CHARACTER SET clause)
- There was an AV when copying selected objects twice and more. It's fixed.
- Fixed the problem with undo after modifying link points.

2. Stored Procedure Editor:

- Fixed the problem with missing user privileges in the stored procedure script.
- Lazy Mode: added the option to select domains as the data type for input/output parameters and variables. In this case IBExpert copies information from the domain definition to the native data type of the parameter/variable.
  It is also possible to drag 'n' drop domain from the Database Explorer.

3. Database Explorer:

- Added the possibility to store server info (server type, server name, server version, connection protocol) and client library name for database folders.

4. Database Statistics:

- A new feature has been added to analyze average record and version length (Firebird 1.5, InterBase 7).
- Some visual improvements.

5. Search Metadata:

- Added the possibility to search in object descriptions.
- Fixed the problem with AV when closing the window after disconnecting from the database.

6. Script Executive:

- There was a problem with the execution of `INPUT` statements: they are executed in another transaction. It's fixed.

7. Test Data Generator:

- Fixed the problem with generating integer values when min and max values were set to `-2147483648` and `2147483647` accordingly.
- Also fixed problems with generating `NUMERIC` and `TIME` values.

8. Data Grid:

- Added feature to calculate aggregate functions (`COUNT`, `SUM`, `MIN`, `MAX`, `AVG`) on numeric and datetime columns. You should click the Show summary footer button on the toolbar of the data view to display the summary footer. After this it is possible to select an aggregate function for each numeric/datetime column separately. IMPORTANT: all calculations are done on client side so do not use this feature on huge datasets with millions of records because IBExpert will fetch all records from the server to calculate aggregates.

9. Table Data Comparer:

- Now an error will be raised if there is no primary key defined for the reference table.

10. Shortcuts:

- Fixed the problem with missing shortcuts when choosing non-english localization in Options / Environment Options. The default shortcuts map is created automatically when IBExpert is started. Don't forget that it is possible to change most of the default shortcuts using [Ctrl + Alt + Shift + L] combination in the majority of the IBExpert windows.

11. Console tools:

- IBECompare was updated. Current version is 1.4.
- IBEScript was updated. Current version is 1.90
- IBEExtract was updated. Current version is 1.94

12. Many other bug fixes and small improvements...

IBExpert 2004.6.17

(Bug fixes and improvements since 2004.4.1)

1. IBExpert:

* IBExpert prevented Windows (2000/XP) to logoff/shutdown. It's fixed.

2. Log Manager:

* Added templates for data logging triggers.

  See Options / General Templates / Data Logging Triggers for more details.

3. Database Designer:

- Fixed problem with copying/pasting of model objects. Now it's possible to copy/paste between diagrams.
- IBExpert ignored the Generate property for views and table constraints. It's fixed.
- Added the option to specify font character set for model objects. See Model Options / General / Font Character Set.

4. Database Comparer:

- Fixed some problems with domains while comparing scripts.

5. SQL Editor:

- The problem with the displaying of error messages while executing a query in the background has been fixed.

6. SP Editor:

- Fixed problem with autogranting privileges for recursive procedures.
- Added Script page. It includes the `CREATE PROCEDURE` statement, stored procedure and parameter descriptions and `GRANT` statements.

7. Extract Metadata:

- Fixed problem with extracting computed blob fields.

8. SP Debugger:

- Fixed some problems with `EXECUTE STATEMENT` and `FOR EXECUTE STATEMENT`.
- The problem with conditional breakpoints (didn't work after Reset Procedure) has been fixed.
- Fixed problem with default values of variables (Firebird 1.5).
- Fixed some problems with handling of `DOUBLE PRECISION` values.

9. Database Explorer:

- Fixed the problem with drag 'n' dropping of domains from one database into another (incorrect length of VARCHAR domains).
- There was a problem with including computed fields into INSERT/UPDATE statements while dragging a table node into the code editor. It's fixed.

10. Query Builder:

- There was an AV after table deletion from the work area. It's fixed.

11. Table Editor:

- There was a problem with autogranting privileges for autoincrement triggers. It's fixed.

12. Table Data Comparer:

- Fixed the problem while comparing tables where all fields are included into the primary key.

13. Script Executive:

- Fixed a problem with stopping script execution after the first error in input file (INPUT sql_script_file).

14. Data View:

- The order of edit controls in the Form View now corresponds to the order of columns in the Grid View.

15. IBEScript was updated. Current version is 1.88.

16. IBEExtract was updated. Current version is 1.92.

17. IBECompare was updated. Current version is 1.3.

18. New languages added and other language files updated.

19. Many other minor bug fixes and small improvements.

IBExpert v. 2004.04.01.1

1. Database Comparer:

- Added a feature allowing script comparison.
- Added the option to store and load settings into/from file. It is possible to store settings into an external file and use it together with IBECompare (see below).

2. IBECompare is a command-line tool to compare databases, scripts and table data.

- IBECompare is a command-line version of corresponding IBExpert tools: Database Comparer and Table Data Comparer.
- Use `-D` option to compare database metadata and script.
  Use `-T` option to compare table data.
  In both cases IBECompare produces an SQL script file.
- It is necessary to specify an input settings file using the `-C` option. You can get the template of this file starting IBECompare with the `-s` option (`IBECompare -s`). In this case IBECompare will create a `config_sample.ini` file within the current directory. Also you can create a settings file using Save configuration button in IBExpert / Tools / Database Comparer.

3. Script Executive:

- Fixed the problem with non-standard terminators handling.
- Added support for the `EXECUTE BLOCK` statement (Firebird 2).

4. SQL Editor:

- Now displays real parameter types and offers different editors for different parameter types while executing queries with parameters.
- Added support for the `EXECUTE BLOCK` statement (Firebird 2).

5. SP/Trigger Debugger:

- Trace Into for `SELECT FROM SP` implemented.
- Fixed problem with `EXECUTE STATEMENT ... INTO :VAR` (value of `VAR` was never modified).
- Added support for default values of input parameters (Firebird 2).

6. Database Restore:

- Fixed the problem with the *Commit after each table* option. In previous versions IBExpert restored databases without this flag when the corresponding checkbox was checked.

7. Grant Manager:

- The problem with long object names (> 31 chars; IB 7) has been fixed.

8. Database Designer:

- Fixed the problem with extracting table and view descriptions while reverse engineering (they were always empty).

9. Database Registration Info / Additional:

- Added *Disable plan request in SQL Editor* and *Disable performance analysis* options.

10. Extract Metadata:

- It is now possible to extract table data into separate files (`TABLE_1.sql`, `TABLE_2.sql`, `TABLE_3.sql` etc.).

- Added support for default values of input parameters (Firebird 2).

11. IBEExtract was updated. Current version is 1.9.

12. IBEScript was updated. Current version is 1.86.

13. New Languages added and other language files updated.

14. Install path was changed, please copy all files from old IBExpert Directory to the new `\Program Files\HK-Software\IBExpert 2004` or just uninstall old version before installing newest version (if you use the IBExpert User Database, files do not need to be copied).

15. Many other bug fixes and small improvements...

IBExpert v. 2004.2.26.1

1. Data Grid:

- It is now possible to customize highlighting of NOT NULL columns captions. Use Options / Environment Options / Grid to customize color and font style of not null columns.

2. Script Executive:

- Added Save As button. Changed the behavior of the Save button.

3. Code Editors:

- Drag 'n' dropping from the DB Explorer and SQL Assistant has been greatly improved. Now, when you drag an object node(s) from DB Explorer or SQL Assistant IBExpert will offer you various versions of text to be inserted into the code editor.
- It is now possible to customize the highlighting of variables. Use Options / Editor Options / Colors to choose color and font style for variables.

4. Tools:

- Table Data Comparer has been implemented. It allows you to compare the data of two tables in different databases and get a difference script which will include corresponding `INSERT`, `UPDATE` and `DELETE` statements. The tables may have different names but they must have the same structure.

5. Project View:

- Added the possibility to sort items in alphabetical order. Use the right-click context menu item *Sort child nodes alphabetically*.

6. Extract Metadata:

- Added the *Decode domains* option. If enabled, the domain types will be inserted as comments just after the domain names. For example:

```
CREATE TABLE Z (
    B BOOL /* INTEGER DEFAULT 0 CHECK (VALUE IN (0, 1)) */ );
```

7. New Help System

- Pressing F1 in any of the IBExpert forms now opens a new web-based Help page. It is also possible to download all Help Pages from http://www.ibexpert.info/documentation/ibexpert_documentation.zip and unzip this in the IBExpert main directory with subdirectories. (There must be a new subdirectory called `documentation`). If a local Help document is available, it will be opened in the browser. Otherwise the browser will open the page from our web server. For any comments please use our newsgroups.

8. A lot of small bug fixes and improvements...

IBExpert v. 2004.1.22.1

1. Database Designer:

- Now supports working with generators.
- Fixed some problems with printing.

2. Data Grid:

- Added highlighting of mandatory (NOT NULL) fields while working with live queries. Captions of not null fields are in bold.

3. Table Editor:

- Added support for external tables.

4. Extract Metadata:

- Added possibility to extract `date/timestamp/time` values with ANSI-prefixes:

```
INSERT INTO MY_TABLE (DATE_FIELD, TIME_FIELD, TIMESTAMP_FIELD)
VALUES (date '01.01.2004', time '12:15:45',
        timestamp '01.01.2004 12:15:45');
```

5. Database Comparer was updated.

6. Log Manager:

- Added the option to *Allow comparing BLOBs in BEFORE UPDATE trigger.*

7. Other bug fixes and small improvements..

IBExpert v. 2003.12.18.1

1. Database Designer:

- This now displays borders of pages (printable parts) with dashed lines. You can customize the page options (size, headers and footers etc.) using the Print Preview form.
- Added possibility to define pre- and postscripts for your database model. The prescript will be inserted into model script just after the CREATE DATABASE or the CONNECT statement. The postscript will be added to the end of the model script.
- It is now possible to define pre- and postscripts for each table separately.
- Fixed the problem with the BOOLEAN data type.

2. Database Registration Info:

- It is now possible to execute sql scripts before and after connecting to the database and before and after disconnecting from the database.
- There is a new option on the Additional page - *Always prompt for a user name and password*. If this option is on IBExpert will display a login prompt dialog each time you try to connect to the database.

3. Script Executive, IBEScript:

- INSERTEX was improved, some problems with quoted identifiers while working with SQL dialect 3 have been fixed.
- Fixed the problem with incorrect error handling while executing SELECT statements.
- IBEScript was updated accordingly. The current version is 1.80.

4. Export Data Into Script:

- New feature allowing the export of data as a set of EXECUTE PROCEDURE statements.

5. Data Grid:

- Added the possibility to copy selected record(s) to clipboard as UPDATE statement(s). This will only work if there is a live query with a primary key.

6. Many small bug fixes and improvements...

IBExpert v. 2003.11.6.1

1. New Version Numbering:

- The version number now corresponds with the date of the build.

2. Table Editor:

- Indices page: fixed the problem with the incorrect display of indices sorting.
- Fields page: the field dependencies list now includes indices, primary and foreign keys.
- Constraints page: added support for the new Firebird feature - user-defined constraint index names.
- Added the possibility to change field default values. Because the server itself doesn't allow the alteration of field default values using ALTER TABLE we have implemented a kind of workaround:

First, IBExpert creates the temporary field with the new DEFAULT value:

```
ALTER table ADD IBE$$TEMP_COLUMN column_type DEFAULT new_default
```

Secondly, IBExpert copies the RDB$DEFAULT_SOURCE and RDB$DEFAULT_VALUE values of the newly created temporary field into RDB$DEFAULT_SOURCE and RB$DEFAULT_VALUE of the field which should be altered:

```
UPDATE RDB$RELATION_FIELDS F1
SET
F1.RDB$DEFAULT_VALUE  = (SELECT F2.RDB$DEFAULT_VALUE
                         FROM RDB$RELATION_FIELDS F2
                         WHERE (F2.RDB$RELATION_NAME = 'table')
                               (F2.RDB$FIELD_NAME ='IBE$$TEMP_COLUMN')),
F1.RDB$DEFAULT_SOURCE = (SELECT F3.RDB$DEFAULT_SOURCE
                         FROM RDB$RELATION_FIELDS F3
                         WHERE (F3.RDB$RELATION_NAME = 'table')
                               (F3.RDB$FIELD_NAME ='IBE$$TEMP_COLUMN'))
WHERE (F1.RDB$RELATION_NAME = 'table')
      (F1.RDB$FIELD_NAME = 'column')
```

After that IBExpert drops the temporary field:

```
ALTER TABLE table DROP IBE$$TEMP_COLUMN
```

3. Blob Viewer:

- Added syntax highlighting for SQL. This is useful if your blobs contain SQL queries.
- Added *As BLR* page. This allows blobs with the subtype 2 to be displayed (for example, RDB$PROCEDURE_BLR of RDB$PROCEDURES table) as BLR.

4. SP Debugger:

- Fixed problem with ROW_COUNT (Firebird).

5. Script Executive:

- OUTPUT command now supports extracting data into SQL scripts as a set of INSERT statements. Two options are available to control this: ASINSERT and INTO. Examples of usage:

```
OUTPUT 'C:\MyScripts\Data.sql';
        SELECT * FROM MY_TABLE;
        OUTPUT;
```

This will produce a set of the following INSERTs:

```
        INSERT INTO MY_TABLE (...) VALUES (...)
```

```
OUTPUT 'C:\MyScripts\Data.sql';
        SELECT * FROM MY_TABLE INTO "MyTable";
        OUTPUT;
```

This will produce a set of the following INSERTs:

```
INSERT INTO "MyTable" (...) VALUES (...)
```

- A full description of all IBExpert's extensions of script language can be found here: www.ibexpert.com/documentation under: Tools / Script Executive / Script Language Extensions.

6. Database Restore:

- It is now possible to specify a client library which will be used while restoring. Default client library is gds32.dll.

7. Database Designer:

- View Editor and Note Editor were redesigned. They are no longer modal.
- Fixed the problem with memory leaks while displaying foreign key marks.
- A lot of bugfixes...

8. Code Editors:

- Added the possibility to customize highlighting for double-quoted string. Use Options / Editor Options / Colors to customize this.
- Added the possibility to customize highlighting for conditional executing directives.
- Code Parameters: now IBExpert displays the list of fields to be inserted when you type the VALUES part of INSERT statements.
- Fixed some problems with Code Insight.

9. View Editor:

- Recreate Script now correctly restores descriptions of view fields.

10. Main Menu:

Added Grid menu item. It contains the following items:

- **Apply best fit:** Save grid data as... Saves grid data into TXT, XLS, HTML or XML formats. This works only with data set grids (field and index grids in the Table Editor, the parameters/variables grid in the Stored Procedure Editor when working in lazy mode), and doesn't work with SQL Assistant lists, the constraint list in Table Editor etc.
- **Copy current record to clipboard:** Copies the current record of any grid/list into the clipboard. Values are delimited with the tab character.
- **Copy all to clipboard:** Copies the content of any grid (including column captions) into the clipboard. Values are delimited with the tab character.

11. IBEScript:

Added possibility to encrypt/decrypt scripts and to execute encrypted scripts. There are two possible ways to encrypt:

- Encrypting without the password. In this case there is no possibility to decrypt an encrypted script but it is possible to execute this script with IBEScript.
- Encrypting with the password. In this case it possible to decrypt the script and execute it with IBExpert if there is correct password specified.

The following options control the encrypting and decrypting:

`-e` - encrypts a script and creates a file with extension "esql" if the output file is not specified.
`-d` - decrypts the encrypted script if it was encrypted with password.
`-p<password>` - password for encrypting/decrypting.
`-o<file_name>` - output file name.

The current version is 1.76.

12. IBEExtract:

- Fixed the problem with the extraction of exceptions.
- The current version is 1.85.

13. Database Comparer:

- Added the option to stop the comparison process.

14. User Database:

- Fixed the problem with using Firebird Embedded while working with the IBExpert User Database.

15. A lot of further bug fixes and small improvements...

IBExpert v.2.5.0.61

1. Script Executive, IBEScript:

- `NOFIELDNAMES` option is now obsolete. This means that there will be no column captions in the output file by default. If you wish to include column captions use `FIELDNAMES` option.
- Added the option to define non-printable characters as a delimiters. For example, if you need the [Tab] character as a delimiter use:

`DELIMITER #9` (decimal representation of tab char)

or

`DELIMITER $9` (hexadecimal representation of tab char)

- Added customization of the delimiter char for the `INSERTEX` command (`DELIMITER` option). If the `DELIMITER` option is missing a comma will be used as the delimiting character.
- Conditional executing of script statements has now been implemented. The syntax is the same as in Delphi:

```
{$IfExists <select_statement>}
{$Else}
{$EndIf}
```

At present IBExpert supports the following conditional directives:

```
$IfExists
$IfNotExists (shortened form is $IfNExists)
$Else
$EndIf
```

When the Script Executive (IBEScript) encounters a `$IfExists` or `$IfNotExists` directive it tries to execute specified `<select_statement>` and, depending on how many records this select returns (0 or >0), executes or skips the next block of statements.

For example, the following script tests the existence of `MY_TABLE` table in the database and, if `MY_TABLE` exists, a new field will be added into it. If there is no `MY_TABLE` table in the database it will be created.

```
{$IfExists SELECT RDB$RELATION_NAME FROM RDB$RELATIONS
           WHERE RDB$RELATION_NAME = 'MY_TABLE'}
ALTER TABLE MY_TABLE ADD NEW_FIELD INTEGER;
{$ELSE}
CREATE TABLE MY_TABLE (
    ID INTEGER NOT NULL,
    NEW_FIELD INTEGER);
{$ENDIF}
```

Conditional directives within statements will be ignored. This means that the following will not work:

```
CREATE TABLE MY_TABLE (
{$IfExists ...}
    ID INTEGER NOT NULL,
{$ENDIF}
    NEW_FIELD INTEGER);
```

To test existence of standard database objects you can use the following syntax:

```
{$IfExists|IfNotExists DOMAIN|TABLE|VIEW|TRIGGER|PROCEDURE|
                       EXCEPTION|GENERATOR|UDF|ROLE object_name}
```

For example:

```
{$IfExists TABLE MY_TABLE}
ALTER TABLE MY_TABLE ADD NEW_FIELD INTEGER;
{$ELSE}
CREATE TABLE MY_TABLE (
    ID INTEGER NOT NULL,
    NEW_FIELD INTEGER);
{$ENDIF}
```

- IBEScript has been updated accordingly. The current version is 1.72.

2. Table Editor:

- Added the possibility to drag 'n' drop fields from the Database Explorer tree and the SQL Assistant into the field list in the Table Editor. This allows you to quickly and easily copy field definitions from one table to another.

3. Database Designer:

- Added support of exceptions and stored procedures.
- Increased flexibility with regard to the customization of the table layout. It is possible to toggle on/off displaying of field name, field type, not null flag, field description and foreign key mark in any combination. It is also possible to display the table description instead of, or together with the table name.
- Copy/Paste features have been improved.
- Added a feature allowing the export of models into BMP/WMF. There is an *Export* button in the Database Designer local menu for this.
- IBExpert now stores printing options between sessions.
- Fixed some bugs with drawing of links.

4. Options / Environment Options / SQL Editor:

- Added the option *Get plan before statement execution.*

5. Tools / SP / Trigger Analyzer:

- Added the option to analyze views too.

6. SP / Trigger Debugger:

- Fixed some problems with the debugging of nested procedures.

7. SQL Editor:

- Added the option to quickly change transaction isolation level for a separate SQL Editor. There is a corresponding button on the SQL Editor toolbar which allows you to choose one of the following isolation levels: Snapshot, Read Committed, Read-Only Table Stability, Read-Write Table Stability.
- Execute in the background with own thread is also now possible.

8. Export Into Script:

- While exporting data as `INSERT` statements it's now possible to add a corresponding `CREATE TABLE` statement into the beginning of the script.

9. User Manager:

- Added AC (Active Connections) column into users list. It shows how many active connections a user has to the specified database. This works only with active databases.

10. Options / Environment Options / Associations:

- Added the option to associate IBExpert with grc-files (database models).

11. Many other bug fixes and small improvements...

IBExpert v.2.5.0.49

Minor bug fixes and improvements.

IBExpert v.2.5.0.47

1. Script Executive:

- It is now possible to execute only a selected part of the script. When there is a selected text within the script, only the selected part will be executed.
- Statements page: Added the options to mark/unmark for the execution of all or only selected statements. Use the popup menu in the statements grid for this.
- Fixed the bug that, in some cases, incorrectly interpreted the `SET TERM` statement.
- The `INSERTEX` command (import from CSV-files) now also allows a colon as delimiter.
- IBEScript is also updated. Current version is 1.62.

2. Dependencies Viewer:

- It is now possible to drop objects directly from the dependencies tree. Use the popup menu in the dependencies tree to drop selected objects. You can also multiselect objects for dropping.

3. Object Editors, Dependencies Page:

- Added the possibility to drop objects directly from the dependencies tree. Use the popup menu in the dependencies tree to drop selected objects. You can also multiselect objects for dropping.

4. Options / Environment Options / Associations:

- Added the option to associate IBExpert with `.fdb` and `.ib` files.

5. Database Registration Info / Additional:

- Added the option *Open database when IBExpert starts*.

6. Data Export / XML Export:

- Added the possibility to export string fields, memo fields and timestamp fields as text (instead of MIME encoded data). Also added an option to specify encoding charset for fields exported as text. Many thanks to Karsten Strobel for the corresponding improvements of the XML export component!

7. Extract Metadata:

- Added *Exclude IBExpert (IBE$*) objects*.

8. SP / Trigger parser:

- Fixed the problem with `EXECUTE STATEMENT`.
- Fixed problems with `ROW_COUNT`, `CURRENT_TRANSACTION` and `CURRENT_CONNECTION` variables.
- The problem with single line comments (Firebird) which begin from position > 0 has been fixed.

9. Trigger Editor:

- Fixed some problems with interpreting Firebird universal triggers type.

10. Trigger Debugger:

Added the possibility to initialize `NEW/OLD` variables from a table record:

- Start the Trigger Debugger (Debug Trigger button in the Trigger Editor).
- Go to the SQL Editor page.
- Modify offered statement if necessary (default is `SELECT * FROM trigger_table`).
- Click the Run button or [F9]) to execute the statement.
- Highlight the necessary record (you can use filters as in the regular SQL Editor).
- Choose *Init NEW variables* or [Ctrl + Shift + N] or *Init OLD variables* or [Ctrl + Shift + O] from the Debugger menu.

11. Stored Procedure Editor:

Added a feature to declare easily and quickly *unknown variables*. For example, you can write your procedure as

```
CREATE PROCEDURE MY_PROC
   AS BEGIN MY_VAR = 5;
     VAR_OUT = MY_VAR;
   SUSPEND;
END
```


and click the Compile button. Of course, this procedure is incorrect and you will get two *Unknown variable* error messages:

- Unknown variable: `MY_VAR`
- Unknown variable: `VAR_OUT`

Now just use the popup menu in the parser messages list and select the corresponding item for each unknown variable to obtain the correct procedure text:

```
CREATE PROCEDURE MY_PROC
   RETURNS (VAR_OUT INTEGER)
      AS DECLARE VARIABLE MY_VAR INTEGER;
      BEGIN MY_VAR = 5; VAR_OUT = MY_VAR;
   SUSPEND;
END
```

Added the possibility to easily and quickly remove *unused variables*. For example, you can write your procedure as:

```
CREATE PROCEDURE MY_PROC
   RETURNS (VAR_OUT INTEGER)
      AS DECLARE VARIABLE MY_VAR INTEGER;
      BEGIN VAR_OUT = 5;
   SUSPEND;
END
```

and click the Compile button. You will receive a parser message: - *Variable 'MY_VAR' is declared, but never used*. Just use the popup menu in the parser messages list to quickly remove a declaration of the unused variable.

12. Global:

Added [Shift + Ctrl + F12] shortcut. This displays a plain list of current database objects with the option to sort, filter, incrementally search objects and, of course, select the desired object.

13. Many other minor bug fixes and improvements...

12.3 Contents

The IBExpert online documentation can be viewed online under http://www.ibexpert.info/documentation/. It can be downloaded from http://www.ibexpert.info/documentation/documentation.zip. (For download instructions please refer to the IBExpert Help menu.)

The first view displays the Help structure. For a complete list of contents click on INDEX at the top of the page to display the index. If you are looking for help about a specific subject use the SEARCH function

In the meantime, should you be unable to find a solution to your problem here, please use one of our newsgroups (in English, German, French and Russian). Should you have any comments or queries directly regarding the Help documentation, or wish to contribute your own articles, please contact documentation@ibexpert.com.

12.4 Additional Help Files

This menu item has been included for third party help files, intended for those third party components included in the IBExpert PlugIns menu. Such Help files can be installed using the IBExpert Options menu: Environment Options / Additional Help.

The installed help files appear here as an additional menu item.

12.5 Product Home Page

The IBExpert Help menu item Product Home Page does none other than open the www.ibexpert.com homepage, which provides product information, news, support, downloads, plugins, purchase and a contact email, in English, German and Russian languages.

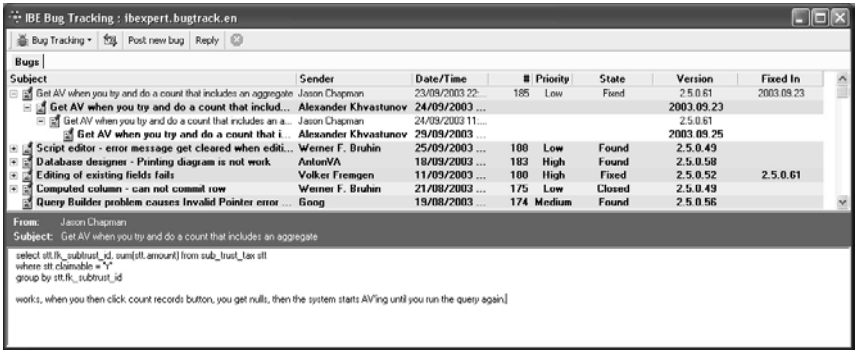
12.6 Send bug reports to

The IBExpert Help menu item Send Bug Reports To allows you to inform us at IBExpert of any bugs discovered or suggestions you may wish to make. The *From*, *To* and *Re* fields are automatically filled; it is merely necessary to type in the message, if possible with an example, in order to enable us to reproduce the operations leading to the problem, and send.

All bug reports can be followed in the Bug Track System.

12.7 Bug Track System

The IBExpert Bug Track System was introduced on the 28.04.2003 in version 2.5.0.38. It allows all users to post and follow all bugs discovered and their current status.



There are currently two bug track groups: English and Russian. Each bug reported receives a number and priority. It is also possible to follow the status (i.e. closed, found, fixed), follow correspondence (by clicking on the + button or using the [+] key), and view the IBExpert version and date including the fix.

If you want to post a bug directly from the Bug Track System (as an alternative to the IBExpert Help menu item Send Bug Reports To), it is first necessary to specify your signature. Simply click on the Configure Bug Tracking System icon, to spring to the Environment Options / IBExpert Bug Track window and input the required information.

Using either the Bug Track pull-down menu or the relevant icons in the toolbar, it is possible to reply to items and send and receive.

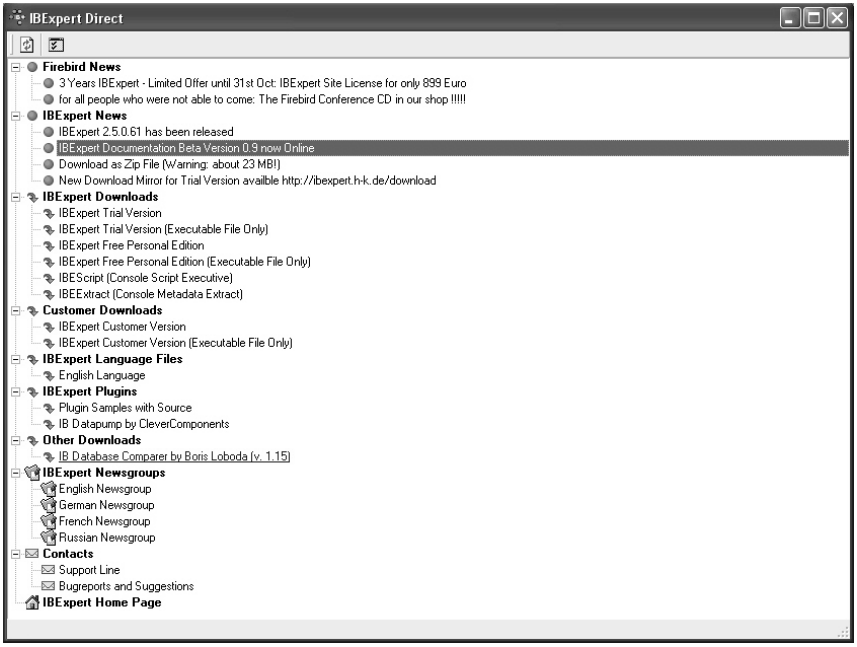
12.8 About

The IBExpert Help menu item About calls the so-called IBExpert splash screen, including the IBExpert logo and current installed version number, with a full copy of the software license on the second page (click the License tab).

Since October 2003 we have introduced a new version numbering system based on the date, as opposed to the more traditional version numbering system.

12.9 IBExpert Direct

The IBExpert Help menu item IBExpert Direct... provides all users with news concerning IBExpert, such as new versions, documentation, downloads, plugins, newsgroups, as well as contact addresses and a direct link to the IBExpert home page, http://www.ibexpert.com/.



The Configure IBExpert Direct icon opens the Options / Environment Options / IBExpert Direct dialog, where it is possible to specify how often the network should be polled for new items, and to configure a proxy server if wished.

12.10 Download Firebird / Purchase InterBase

These last three items in the IBExpert Help menu provide direct links to the software producers, for those wishing to purchase or download InterBase or Firebird.

- Download InterBase Open Edition - currently invalid.
- Buy Borland InterBase - opens the link: http://www.borland.com/interbase/. By clicking DOWNLOADS, it is possible to download the newest trial version (currently InterBase 7.5, November 29, 2004). By clicking PURCHASE, you can gain access to the online web shop, or search for your nearest retailer. Please refer to InterBase 7.5 trial version for further details.
- Download Firebird - opens the link: http://firebird.sourceforge.net/. Please refer to download and install Firebird for further details.

I SQL Language Reference

Here is some basic information regarding DDL, DML and stored procedure and trigger language. Refer to the InterBase SQL Language Reference handbook for detailed information.

Please also refer to the IBExpert Tools menu: Script Executive / Script Language Extensions for IBExpert's own invaluable extensions.

I.1 Firebird SQL

Every database management system has its own idiosyncrasies in the ways it implements SQL. Firebird adheres to the SQL standard more rigorously than any other RDBMS except possibly its 'cousin', InterBase®. Developers migrating from products that are less standards-compliant often wrongly suppose that Firebird is quirky, which is really not true at all.

The following excerpts have been taken from the *Firebird Quick Start Guide, © IBPhoenix Publications 2002, 2003.* Many thanks to Paul Beach (www.ibphoenix.com)!

I.1.1 String delimiter symbol

Strings in Firebird are delimited by a pair of single quote symbols `'I am a string'` (ASCII code 39 *NOT* 96). If you used earlier versions of Firebird's relative, InterBase®, you might recall that double and single quotes were interchangeable as string delimiters. Double quotes cannot be used as string delimiters in Firebird.

Source: *Firebird Quick Start Guide, © IBPhoenix Publications 2002, 2003.*

I.1.2 Double–quoted identifiers

Before the SQL-92 standard, it was not legal to have object names (identifiers) in a database that duplicated keywords in the language, were case-sensitive or contained spaces. SQL-92 introduced a single new standard to make any of them legal, provided that the identifiers were defined within pairs of double-quote symbols (ASCII 34) and were always referred to using double-quote delimiters.

The purpose of this "gift" was to make it easier to migrate metadata from non-standard RDBMSs to standards-compliant ones. The down-side is that, if you choose to define an identifier in double quotes, its case-sensitivity and the enforced double-quoting will remain mandatory.

Firebird does permit a slight relaxation under a very limited set of conditions: if the identifier which was defined in double-quotes 1) was defined as all upper-case, 2) is not a keyword and 3) does not contain any spaces, then it can be used in SQL unquoted as long as it is used in all upper-case.

Unless you have a compelling reason to define quoted identifiers, it is usually recommended that you avoid them. Firebird happily accepts a mix of quoted and unquoted identifiers - so there is no problem including that keyword which you inherited from a legacy database, if you need to.

Source: *Firebird Quick Start Guide, © IBPhoenix Publications 2002,2003*

I.1.3 Apostrophes in strings

If you need to use an apostrophe inside a Firebird string, you can "escape" the apostrophe character by preceding it with another apostrophe.

For example, this string will give an error:

```
'Joe's Emporium'
```

because the parser encounters the apostrophe and interprets the string as 'Joe' followed by some unknown keywords.

To make this a legal string, double the apostrophe character:

```
'Joe''s Emporium'
```

Notice that this is TWO single quotes, not one double-quote.

Source: *Firebird Quick Start Guide, © IBPhoenix Publications 2002,2003*

I.1.4 Concatenation of strings

The concatenation symbol in SQL is two 'pipe' symbols (ASCII 124, in a pair with no space between). In SQL, the '+' symbol is an arithmetic operator and it will cause an error if you attempt to use it for concatenating strings. The following expression prefixes a character column value with the characters 'Reported by:':

```
'Reported by: ' || LastName
```

Take care with concatenations. Be aware that Firebird will raise an error if your expression attempts to concatenate two or more char or varchar columns whose potential combined lengths would exceed the maximum length limit for a char or a varchar (32KB).

Please also refer to expressions involving NULL, for further information concerning concatenation in expressions.

Source: *Firebird Quick Start Guide, © IBPhoenix Publications 2002,2003*

I.1.5 Division of an integer by an integer

Firebird accords with the SQL standard by truncating the result (quotient) of an integer/integer calculation to the next lower integer. This can have bizarre results unless you are aware of it.

For example, this calculation is correct in SQL:

```
1 / 3 = 0
```

If you are upgrading from a RDBMS which resolves integer/integer division to a float quotient, you will need to alter any affected expressions to use a float or scaled numeric type for either dividend, divisor, or both.
For example, the calculation above could be modified thus in order to produce a non-zero result:

```
1.000 / 3 = 0.333
```

Source: *Firebird Quick Start Guide, © IBPhoenix Publications 2002,2003*

I.1.6 Expressions involving NULL

In SQL, NULL is not a value. It is a condition, or state, of a data item, in which its value is unknown. Because it is unknown, null cannot behave like a value. When you try to perform arithmetic on NULL, or involve it with values in other expressions, the result of the operation will always be NULL. It is not zero or blank or an "empty string" and it does not behave like any of these values.

Here are some examples of the types of surprises you will get if you try to perform calculations and comparisons with NULL:

```
1 + 2 + 3 + NULL = NULL
if (a = b) then
MyVariable = 'Equal'
else
MyVariable = 'Not equal';
```

will return 'Not equal' if both `a` and `b` are null.

```
if (a <> b) then
MyVariable = 'Not equal'
else
MyVariable = 'Equal';
```

will also return 'Not equal' if both `a` and `b` are null.

```
FirstName ||' ' || LastName
```

will return NULL if either `FirstName` or `LastName` is NULL.

Source: *Firebird Quick Start Guide, © IBPhoenix Publications 2002,2003*

I.2 DDL – Data Definition Language

DDL is the abbreviation for Data Definition Language.

The task of DDL is database definition, i.e. the predefinition and manipulation of the metadata.

Using different DDL commands, the database metadata can be created, altered and deleted. For example table structure, use of indices, the activation of exceptions and con-

struction of procedures can all be defined by DDL commands. DDL commands are a subarea of SQL; the range of the SQL language is composed of DDL and DML together.

Important: In SQL statements passed to DSQL, omit the terminating semicolon. In embedded applications written in C and C++, and in isql, the semicolon is a terminating symbol for the statement, so it must be included.

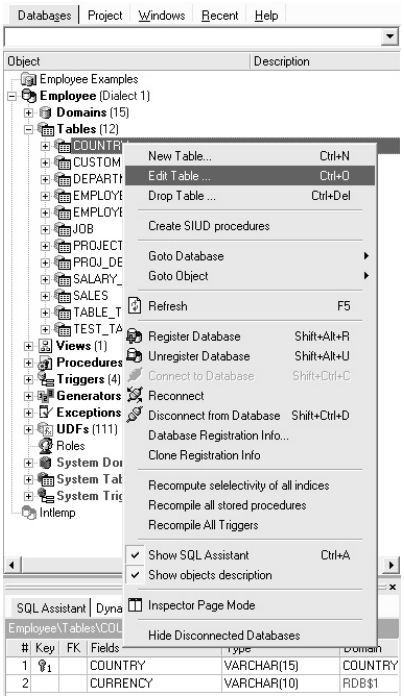
The source of all definitions included in this section is the Borland InterBase Language Reference.

I.2.1 ALTER

ALTER is the SQL command used to modify database objects, i.e. databases, domains, tables, views, triggers, procedures, generators, UDFs etc. can all be changed using the ALTER command.

The different versions of the ALTER command serve to extend or change an already defined structure, the type of alteration defined as an additional attribute of the command. This allows, for example, the metadata in already defined tables, stored procedures or triggers to be manipulated.

A database object can be altered in IBExpert using the DB Explorer right mouse button menu (Edit ...) or simply by double-clicking on the object to be altered.



Alterations can of course also be made directly in the SQL Editor.

I.2.2 COMMIT

The `COMMIT` command makes a transaction's changes to the database permanent. It is used to start all transactions.

`COMMIT` is used to end a transaction and:

- Write all updates to the database.
- Make the transaction's changes visible to subsequent `SNAPSHOT` transactions or `READ COMMITTED` transactions.
- Close open cursors, unless the `RETAIN` argument is used.

After executing a transaction with [F9] or the

icon, and all operations in the transaction have been successfully performed by the server, the changes to the database must be explicitly committed. This can be done using [Ctrl + Alt + C] or the

icon.

Of course, those competent in SQL can also enter the command directly in SQL Editor.

Syntax:

```
COMMIT [WORK] [TRANSACTION name] [RELEASE] [RETAIN [SNAPSHOT]];
```

WORK - an optional work used for compatibility with other relational databases that require it.

TRANSACTION name - Commits a transaction *name* to database. Without this option, COMMIT affects the default transaction.

RELEASE - Available for compatibility with earlier versions of InterBase/Firebird.

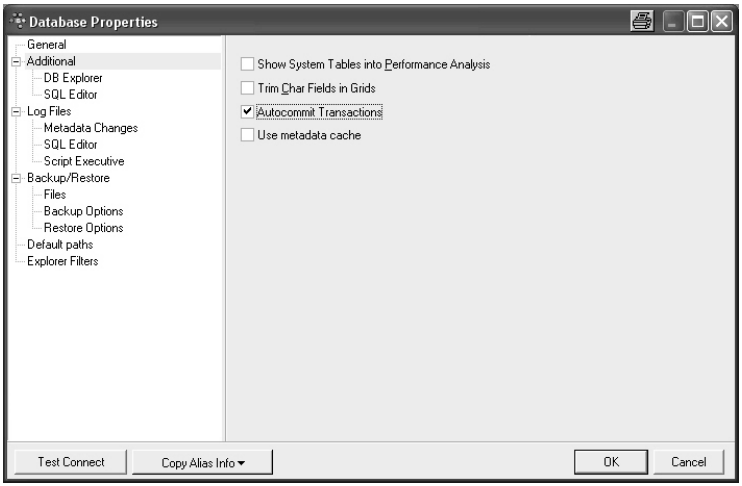
RETAIN [SNAPSHOT] - Commits changes and retains current transaction context.

The transaction name is only valid in an embedded SQL application using SQL or DSQL, where more than one transaction can be active at a time.

A transaction ending with `COMMIT` is considered a successful termination. Always use `COMMIT` or `ROLLBACK` to end the default transaction. *Tip*: after read-only transactions, which make no database changes, use `COMMIT` rather than `ROLLBACK`. The effect is the same, but the performance of subsequent transactions is better and the system resources used by them are reduced.

This statement is not valid inside a trigger, because a trigger is started automatically as part of a larger transaction, with other triggers perhaps firing after it. It is also not valid inside a stored procedure because the procedure might be invoked from a trigger.

In IBExpert it is possible to force all commands to be automatically committed, by checking the Autocommit Transactions box in the Database Properties dialog / Additional (menu item: Database / Database Registration Info...):



However, this is
NOT recommended, as it is all too easy to accidentally drop a database (instead of a database field for example), as the developer is no longer asked for confirmation, before committing.

I.2.3 CONNECT

A connection can be made to one or more existing databases using the CONNECT command.

The connection parameters can be specified in IBExpert using the menu item Database / Register Database. Here a specified connection may also be tested. the IBExpert menu item Services / Communication Diagnostics may be used to analyze connection problems. It delivers a detailed protocol of the test connect to a registered InterBase/Firebird server and the results. IBExpert also offers toolbar icons for connecting, reconnecting and disconnecting to a registered database.

The CONNECT statement initializes the database data structures and determines if the database is on the originating node (local database) or on another node (remote database). An error message occurs if InterBase/Firebird cannot locate the database. The CONNECT statement attaches to the database and verifies the header page. The database file must contain a valid database, and the on-disk structure (ODS) version number of the database must be recognized by the installed InterBase version on the server.

It is possible to specify a cache buffer for the process attaching to a database. In SQL programs, a database must first be declared with the SET DATABASE command, before it can be opened with the CONNECT statement. When attaching to a database, CONNECT uses the default character set (NONE), or one specified in a previous SET NAMES statement.

A subset of CONNECT features is available in ISQL (see syntax below). ISQL can only be connected to one database at a time. Each time the CONNECT statement is used to con-

nect to a database, previous attachments are disconnected. ISQL does not use SET DATABASE.

Syntax ISQL form:

```
CONNECT 'filespec' [USER 'username'][PASSWORD 'password']
   [CACHE int] [ROLE 'rolename']
```

SQL form:

```
CONNECT [TO] {ALL | DEFAULT} config_opts
   | db_specs config_opts [, db_specs config_opts...];
<db_specs> = dbhandle
   | {'filespec' | :variable} AS dbhandle
<config_opts> = [USER {'username' | :variable}]
   [PASSWORD {'password' | :variable}]
   [ROLE {'rolename' | :variable}]
   [CACHE int [BUFFERS]]
```

- **{ALL | DEFAULT}** - Connects to all databases specified with SET DATABASE; options specified with CONNECT TO ALL affect all databases.
- **'filespec' Database file name** - can include path specification and node. The filespec must be in quotes if it includes spaces.
- **dbhandle -** Database handle declared in a previous SET DATABASE statement; available in embedded SQL but not in isql.
- **:variable** Host-language variable specifying a database, user name, or password; available in embedded SQL but not in isql.
- **AS dbhandle** - Attaches to a database and assigns a previously declared handle to it; available in embedded SQL but not in isql.
- **USER {'username' | :variable}** String or host-language variable that optionally specifies a user name for use when attaching to the database. The server checks the user name against the security database. User names are case insensitive on the server. PC clients must always send a valid user name and password.
- **PASSWORD {'password' | :variable} -** String or host-language variable, up to 8 characters in size, that specifies password for a user listed in the security database, if used, for use when attaching to the database. The server checks the user name and password against the security database. Case sensitivity is retained for the comparison. PC clients must always send a valid user name and password.
- **ROLE {'rolename' | :variable**} String or host-language variable, up to 67 characters in size, which optionally specifies the role that the user adopts on connection to the database. The user must have previously been granted membership in the role to gain the privileges of that role. Regardless of role memberships granted, the user has the privileges of a role at connect time only if a ROLE clause is specified in the connection. The user can
  adopt at most one role per connection, and cannot switch roles except by reconnecting.
- **CACHE int [BUFFERS]** - Sets the number of cache buffers for a database (default is 75), which determines the number of database pages a program can use at the same time. Values for int:
  • Default: 256
  • Maximum value: system-dependent

This can be used to set a new default size for all databases listed in the CONNECT statement that do not already have a specific cache size, or specify a cache for a program that uses a single database. The size of the cache persists as long as the attachment is active. A decrease in cache size does not affect databases that are already attached through a server. Do not use the filespec form of database name with cache assignments.

Example:

```
CONNECT C:\DB01\DB01.GDB USER SYSDBA PASSWORD masterkey
```

In the above example a connection is made to the InterBase database DB01.GDB in the C:\DB01 directory on a Windows NT Server.

When making a connection to a UNIX server the path definitions need to be adapted accordingly:

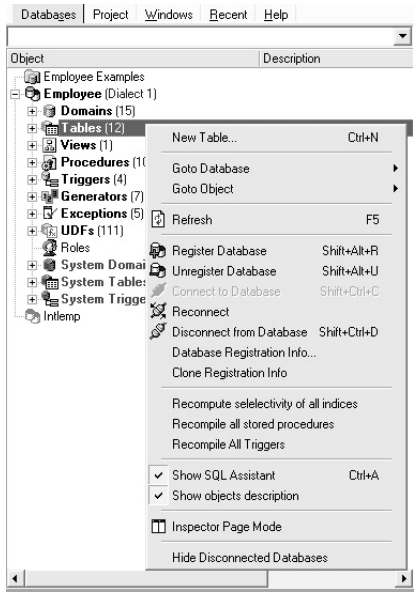
```
CONNECT /usr/db01/db01.gdb USER SYSDBA PASSWORD masterkey
```

If the user details are not specified when performing the CONNECT command, the relevant system variables for establishing the connection to the specified database are used. This can have the consequence, that if these variables have undefined values, a database connection is not made, and instead an appropriate error message appears.

I.2.4 CREATE

CREATE is the SQL command used to create database objects, i.e. databases, domain, tables, views, triggers, procedures, generators, UDFs etc. can all be defined using the CREATE command.

A database object can be created in IBExpert using the DB Explorer right mouse button menu (New ...), the Database menu, or the respective New Database Object icon.



It can of course also be created, by those who are competent in SQL, directly in the SQL Editor. The `CREATE` command syntax can be found under the respective subjects (e.g. Create Database, Create Domain, Create Table, etc.).

I.2.5 DECLARE EXTERNAL FUNCTION (incorporating a new UDF library)

In order to use an already defined or programmed UDF (User Defined Function) within an InterBase/Firebird database, this has to be explicitly declared using the `DECLARE EXTERNAL FUNCTION` command.

The `DECLARE EXTERNAL FUNCTION` command syntax is as follows:

```
DECLARE EXTERNAL FUNCTION name [datatype | CSTRING (int)

[, datatype | CSTRING (int) …]]
   RETURNS {datatype [BY VALUE] | CSTRING (int) | PARAMETER n}  [FREE_IT]
ENTRY_POINT <External_Function_Name>
   MODULE NAME <Library_Name>;
```

By declaring the UDF, the database is informed of the following for an existing UDF (`<External_Function_Name>`):

- **name** - Name of the UDF to use in SQL statements. It can be different to the name of the function specified after the `ENTRY_POINT` keyword.
- **datatype** - Data type of an input or return parameter. All input parameters are passed to a UDF by reference. Return parameters can be passed by value. It cannot be an array element.

- **CSTRING (int)** - Specifies a UDF that returns a null-terminated string `int` bytes in length.
- **RETURNS** - Specifies the return value of a function.
- **BY VALUE** - Specifies that a return value should be passed by value rather than by reference.
- **PARAMETER n** - Specifies that the nth input parameter is to be returned. Used when the return data type is a blob.
- **FREE_IT** - Frees memory of the return value after the UDF finishes running.
- **<External_Function_Name>** - Quoted string that contains the function name as it is stored in the library that is referenced by the UDF. The `entryname` is the actual name of the function as stored in the UDF library. It does not have to match the name of the UDF as stored in the database.
- **<Library_Name>** - Quoted specification identifying the library that contains the UDF. The library must reside on the same machine as the InterBase/Firebird server. On any platform, the module can be referenced with no path name if it is in. <InterBase/Firebird_home>/UDF or <InterBase/Firebird_home>/intl. If the library is in a directory other than <InterBase/Firebird_home>/UDF or <InterBase/Firebird_home>/intl, you must specify its location in InterBase/Firebird's configuration file (`ibconfig`) using the `EXTERNAL_FUNCTION_DIRECTORY` parameter. It is not necessary to supply the extension to the module name.

The UDF name in the database does not have to correspond to the original function name. The input parameters are basically transferred `BY REFERENCE`. In the case of the return parameters it is also possible to specify the form `BY VALUE`, using the optional `BY VALUE` parameter.

Note: Whenever a UDF returns a value by reference to dynamically allocated memory, you must declare it using the `FREE_IT` keyword in order to free the allocated memory.

To specify a location for UDF libraries in a configuration file, enter the following for Windows platforms:

```
EXTERNAL_FUNCTION_DIRECTORY D:\Mylibraries\InterBase
```

For UNIX, the statement does not include a drive letter:

```
EXTERNAL_FUNCTION_DIRECTORY \Mylibraries\InterBase
```

The InterBase/Firebird configuration file is called `ibconfig` or `firebird.conf` on all platforms.

Examples:

The following isql statement declares the `TOPS()` UDF to a database:

```
DECLARE EXTERNAL FUNCTION TOPS
   CHAR(256), INTEGER, BLOB
   RETURNS INTEGER BY VALUE
   ENTRY_POINT 'te1' MODULE_NAME 'tm1';
```

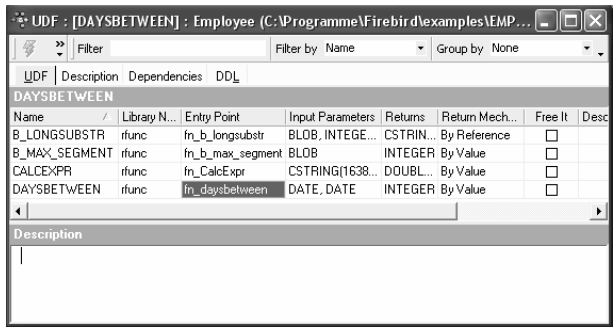
This example does not need the `FREE_IT` keyword because only cstrings, `CHAR` and `VARCHAR` return types require memory allocation.

The next example declares the LOWERS() UDF and frees the memory allocated for the return value:

```
DECLARE EXTERNAL FUNCTION LOWERS VARCHAR(256)
   RETURNS CSTRING(256) FREE_IT
   ENTRY POINT 'fn_lower' MODULE_NAME 'udflib';
```

In the example below (taken from the RFunc library) a function SUBSTR is declared, which calculates the substring of strings, from character i1 and length maximum i2:

```
DECLARE EXTERNAL FUNCTION SUBSTR
    CSTRING(256),
    INTEGER,
    INTEGER
RETURNS CSTRING(256)
ENTRY_POINT 'fn_substr' MODULE_NAME 'rfunc';
```



ENTRY_POINT

ENTRY_POINT is a term used in the declaration of an external function.

Syntax:

```
ENTRY_POINT <External_Function_Name>
```

The entry point is a text which specifies when the function should jump into a starting address from a DLL.

MODULE NAME

The DLL name of a UDF is entered as the last parameter when declaring an external function.

Syntax:

```
MODULE NAME <Library_Name>
```

It specifies in which UDF library the UDF can be found (<Library_Name>). Whether the file suffix needs to be entered or not, and how, is dependent upon the operating sys-

tem. For example, Linux requires the suffix .SO (Shared Object Library); in Windows .DLL (Dynamic Link Library).

RETURNS

RETURNS is a term used in the declaration of an external function. Here the output parameters are specified (i.e. data type and in which form).

Syntax:

```
RETURNS <Return_Type>
```

RETURN parameters can also be specified in the form BY VALUE, using the optional BY VALUE parameter.

I.2.6 DESCRIBE

Provides information about columns that are retrieved by a dynamic SQL (DSQL) statement, or information about the dynamic parameters that a statement passes. Available in gpre.

This feature is available in IBExpert in most object editors (please refer to Table Editor / Description for further information).

Syntax:

```
DESCRIBE [OUTPUT | INPUT] statement
   {INTO | USING} SQL DESCRIPTOR xsqlda;
```

- **OUTPUT [Default]** - Indicates that column information should be returned in the XSQLDA.
- **INPUT** - Indicates that dynamic parameter information should be stored in the XSQLDA
- **statement** - A previously defined alias for the statement to DESCRIBE. Use PREPARE to define aliases.
- **{INTO | USING} SQL DESCRIPTOR xsqlda** - Specifies the XSQLDA to use for the DESCRIBE statement.

DESCRIBE has two uses:

- As a describe output statement, DESCRIBE stores into an XSQLDA a description of the columns that make up the select list of a previously prepared statement. If the PREPARE statement includes an INTO clause, it is unnecessary to use DESCRIBE as an output statement.
- As a describe input statement, DESCRIBE stores into an XSQLDA a description of the dynamic parameters that are in a previously prepared statement. DESCRIBE is one of a group of statements that process DSQL statements.

Statement	*Purpose*
PREPARE	Prepares a DSQL statement for execution.

DESCRIBE	Fills in the XSQLDA with information about the statement.
EXECUTE	Executes a previously prepared statement.
EXECUTE IMMEDIATE	Prepares a DSQL statement, executes it once, and discards it.

Separate DESCRIBE statements must be issued for input and output operations. The INPUT keyword must be used to store dynamic parameter information.

Important: When using DESCRIBE for output, if the value returned in the sqld field in the XSQLDA is larger than the sqln field, you must:

- Allocate more storage space for XSQLVAR structures.
- Reissue the DESCRIBE statement.

Note The same XSQLDA structure can be used for input and output if desired.

Example:

The following embedded SQL statement retrieves information about the output of a SELECT statement:

```
EXEC SQL
   DESCRIBE Q INTO xsqlda
```

The following embedded SQL statement stores information about the dynamic parameters passed with a statement to be executed:

```
EXEC SQL
   DESCRIBE INPUT Q2 USING SQL DESCRIPTOR xsqlda;
```

I.2.7 DISCONNECT

The DISCONNECT command detaches an application from one or more databases, defined by its/their database handle, and frees the relevant sources. Available in gpre.

In IBExpert there is a toolbar icon to execute this command (or alternatively use the IBExpert menu item Database / Disconnect from Database).

Syntax:

```
DISCONNECT ;
```

- **ALL|DEFAULT** - Either keyword detaches all open databases.
- **dbhandle** - Previously declared database handle specifying a database to detach.

DISCONNECT closes a specific database identified by a database handle or all databases, releases resources used by the attached database, zeroes database handles, commits the default transaction if the gpre -manual option is not in effect, and returns an error if any non-default transaction is not committed.

Before using DISCONNECT, commit or roll back the transactions affecting the database to be detached.

Examples:

The following embedded SQL statements close all databases:

```
EXEC SQL
DISCONNECT DEFAULT;

EXEC SQL
DISCONNECT ALL;
```

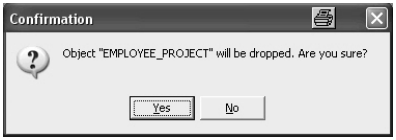
The following embedded SQL statements close the databases identified by their handles:

```
EXEC SQL
DISCONNECT DB1;

EXEC SQL
DISCONNECT DB1, DB2;
```

I.2.8 DROP

DROP is the SQL command used to delete database objects, i.e. databases, domains, tables, views, triggers, procedures, generators, UDFs etc. can all be deleted using the DROP command.

A database object can be dropped in IBExpert using the DB Explorer right mouse button menu (Drop ...). IBExpert requires confirmation of this command, as it is irreversible.



The DROP command can of course also be used directly in the SQL Editor. More information can be found under the respective subjects (e.g. Drop Database, Drop Domain, Drop Table, etc.).

Syntax:

```
DROP <database_object_type> <object_name>;
```

Example:

```
DROP TABLE Customer;
```

I.2.9 END DECLARE SECTION

Identifies the end of a host-language variable declaration section. Available in gpre.

Syntax:

```
END DECLARE SECTION;
```

The END DECLARE SECTION command is used in embedded SQL applications to identify the end of host-language variable declarations for variables used in subsequent SQL statements.

Example:

The following embedded SQL statements declare a section and single host-language variable:

```
EXEC SQL
   BEGIN DECLARE SECTION;
      BASED_ON EMPLOYEE.SALARY salary;

EXEC SQL
   END DECLARE SECTION;
```

I.2.10 EVENT

Please refer to the following:

- EVENT INIT statement
- EVENT WAIT statement

EVENT INIT

EVENT INIT is the first step in the InterBase two-part synchronous event mechanism:

- EVENT INIT registers an application's interest in an event.
- EVENT WAIT causes the application to wait until notified of the event's occurrence.

EVENT INIT registers an application's interest in a list of events in parentheses. The list should correspond to events posted by stored procedures or triggers in the database. If an application registers interest in multiple events with a single EVENT INIT, then when one of those events occurs, the application must determine which event occurred. The command EVENT INIT is only required by embedded SQL programmers, and not required when programming the BDE.

Events are posted by a POST_EVENT call within a stored procedure or trigger. The event manager keeps track of events of interest. At commit time, when an event occurs, the event manager notifies interested applications.

The EVENT INIT command is constructed as follows:

Syntax:

```
EVENT INIT request_name [dbhandle]
   [('string' | :variable [, 'string' | :variable …]);
```

- **request_name** - Application event handle.
- **dbhandle** - Specifies the database to examine for occurrences of the events; if omitted, `dbhandle` defaults to the database named in the most recent `SET DATABASE` statement.
- **`string'** - Unique name identifying an event associated with `event_name`.
- **:variable** - Host language character array containing a list of event names to associate with.

Example:

The following embedded SQL statement registers interest in an event:

```
EXEC SQL
  EVENT INIT ORDER_WAIT EMPDB ('new_order');
```

EVENT WAIT

Causes an application to wait until notified of an event's occurrence. Available in gpre.

Syntax:

```
EVENT WAIT request_name;
```

- **request_name** - Application event handle declared in a previous `EVENT INIT` statement.

`EVENT WAIT` is the second step in the InterBase/Firebird two-part synchronous event mechanism. After a program registers interest in an event, `EVENT WAIT` causes the process running the application to sleep until the event of interest occurs.

Examples:

The following embedded SQL statements register an application event name and indicate the program is ready to receive notification when the event occurs:

```
EXEC SQL
   EVENT INIT ORDER_WAIT EMPDB ('new_order');

EXEC SQL
   EVENT WAIT ORDER_WAIT;
```

I.2.11 EXECUTE

The `EXECUTE` command performs a specified SQL statement. The statement can be any SQL data definition, manipulation, or transaction management statement. Once it is prepared, a statement can be executed any number of times.

SQL commands can be executed using the [F9] key or following icon:

enabling the SQL code to be executed and tested before finally committing.

Should a part of the text have been highlighted, only the marked portion is executed, which often causes an error message. If the execution has been successful, the SQL can be committed using the respective icon or [Ctrl + Alt + C].

Syntax:

```
EXECUTE [TRANSACTION transaction] statement
   [USING SQL DESCRIPTOR xsqlda] [INTO SQL DESCRIPTOR xsqlda];
```

- **request_name** - Application event handle declared in a previous `EVENT INIT` statement.
- **TRANSACTION transaction** - Specifies the transaction under which execution occurs: This clause can be used in SQL applications running multiple, simultaneous transactions to specify which transaction controls the `EXECUTE` operation.
- **USING SQL DESCRIPTOR** - Specifies those values corresponding to the prepared statement's parameters should be taken from the specified XSQLDA. It need only be used for statements that have dynamic parameters.
- **INTO SQL DESCRIPTOR** - Specifies that return values from the executed statement should be stored in the specified XSQLDA. It need only be used for DSQL statements that return values.
- **xsqlda** - XSQLDA host-language variable.

Note If an `EXECUTE` statement provides both a `USING DESCRIPTOR` clause and an `INTO DESCRIPTOR` clause, then two XSQLDA structures must be provided.

`EXECUTE` carries out a previously prepared DSQL statement. It is one of a group of statements that process DSQL statements.

- **PREPARE** - Readies a DSQL statement for execution.
- **DESCRIBE** - Fills in the XSQLDA with information about the statement.
- **EXECUTE** - Executes a previously prepared statement.
- **EXECUTE IMMEDIATE** - Prepares a DSQL statement, executes it once, and discards it (please refer to `EXECUTE IMMEDIATE` statement for further information).

Before a statement can be executed, it must be prepared using the `PREPARE` statement. The statement can be any SQL data definition, manipulation, or transaction management statement. Once it is prepared, a statement can be executed any number of times.

Example:

The following embedded SQL statement executes a previously prepared DSQL statement:

```
EXEC SQL
   EXECUTE DOUBLE_SMALL_BUDGET;
```

The next embedded SQL statement executes a previously prepared statement with parameters stored in an XSQLDA:

```
EXEC SQL
   EXECUTE Q USING DESCRIPTOR xsqlda;
```

The following embedded SQL statement executes a previously prepared statement with parameters in one XSQLDA, and produces results stored in a second XSQLDA:

```
EXEC SQL
   EXECUTE Q USING DESCRIPTOR xsqlda_1 INTO DESCRIPTOR xsqlda_2;
```

EXECUTE IMMEDIATE

Prepares a dynamic SQL (DSQL) statement, executes it once, and then discards it. Available in gpre.

Syntax:

```
EXECUTE IMMEDIATE [TRANSACTION transaction]
   {:variable | 'string'} [USING SQL DESCRIPTOR xsqlda];
```

- **TRANSACTION transaction** - Specifies the transaction under which execution occurs.
- **:variable** - Host variable containing the SQL statement to execute.
- '**string'** - A string literal containing the SQL statement to execute.
- **USING SQL DESCRIPTOR** - Specifies that values corresponding to the statement's parameters should be taken from the specified XSQLDA.
- **xsqlda** - XSQLDA host-language variable:

`EXECUTE IMMEDIATE` prepares a DSQL statement stored in a host-language variable or in a literal string, executes it once and discards it. To prepare and execute a DSQL statement for repeated use, use `PREPARE` and `EXECUTE` instead of `EXECUTE IMMEDIATE`.

The `TRANSACTION` clause can be used in SQL applications running multiple, simultaneous transactions to specify which transaction controls the `EXECUTE IMMEDIATE` operation.

The SQL statement to execute must be stored in a host variable or be a string literal. It can contain any SQL data definition statement or data manipulation statement that does not return output.

`USING DESCRIPTOR` enables `EXECUTE IMMEDIATE` to extract the values of a statement's parameters from an XSQLDA structure previously loaded with appropriate values.

Example:

The following embedded SQL statement prepares and executes a statement in a host variable:

```
EXEC SQL
   EXECUTE IMMEDIATE :insert_date;
```

EXECUTE PROCEDURE

Calls a specified stored procedure. Available in gpre, DSQL, and isql.

In IBExpert a procedure can be executed in the Stored Procedure Editor or SQL Editor using the [F9] key or following icon:

Syntax SQL form:

```
EXECUTE PROCEDURE [TRANSACTION transaction]
   name [:param [[INDICATOR]:indicator]]
      [, :param [[INDICATOR]:indicator] …]
   [RETURNING_VALUES :param [[INDICATOR]:indicator]
      [, :param [[INDICATOR]:indicator] …]];
```

DSQL form:

```
EXECUTE PROCEDURE name [param [, param …]]
   [RETURNING_VALUES param [, param …]]
```

isql form:

```
EXECUTE PROCEDURE name [param [, param …]]
```

- **TRANSACTION transaction** - Specifies the `TRANSACTION` under which execution occurs.
- **name** - Name of an existing stored procedure in the database.
- **param** - Input or output parameter; can be a host variable or a constant.
- **RETURNING_VALUES: param** - Host variable which takes the values of an output parameter.
- **[INDICATOR] :indicator** - Host variable for indicating NULL or unknown values.

`EXECUTE PROCEDURE` calls the specified stored procedure. If the procedure requires input parameters, they are passed as host-language variables or as constants. If a procedure returns output parameters to a SQL program, host variables must be supplied in the `RETURNING_VALUES` clause to hold the values returned.

In isql, do not use the `RETURN` clause or specify output parameters. isql will automatically display return values.

Note: in DSQL, an `EXECUTE PROCEDURE` statement requires an input descriptor area if it has input parameters and an output descriptor area if it has output parameters.

In embedded SQL, input parameters and return values may have associated indicator variables for tracking NULL values. Indicator variables are integer values that indicate unknown or NULL values of return values.

An indicator variable that is less than zero indicates that the parameter is unknown or NULL. An indicator variable that is zero or greater indicates that the associated parameter is known and not NULL.

Examples:

The following embedded SQL statement demonstrates how the executable procedure, DEPT_BUDGET, is called from embedded SQL with literal parameters:

```
EXEC SQL
   EXECUTE PROCEDURE DEPT_BUDGET 100
     RETURNING_VALUES :sumb;
```

The next embedded SQL statement calls the same procedure using a host variable instead of a literal as the input parameter:

```
EXEC SQL
   EXECUTE PROCEDURE DEPT_BUDGET :rdno
     RETURNING_VALUES :sumb;
```

I.2.12 GRANT

GRANT is the SQL statement, used to assign privileges to database users for specified database objects. Grants can be assigned and revoked using the IBExpert Grant Manager, the relevant object editors' Grants pages, or the SQL Editor.

InterBase/Firebird offers the following access privileges at database object level:

Privilege	*Allows user to:*
SELECT	Read data.
INSERT	Write new data.
UPDATE	Modify existing data.
DELETE	Delete data.
ALL	Select, insert, update, delete data, and reference a primary key from a foreign key. (*Note*: does not include references or code for InterBase 4.0 or earlier).
EXECUTE	Execute or call a stored procedure.
REFERENCES	Reference a primary key with a foreign key.
role	Use all privileges assigned to the role (please refer to Role for further information).

PUBLIC is used to assign a set of privileges to every user of the database. Using the PUBLIC keyword does not grant the specified rights to stored procedures, only to all database users. Procedures need to be specified explicitly. *Please note*: PUBLIC is really public! This GRANT option enables all users to access and manipulate a database object with PUBLIC rights, even certain system files.

Table Interactions:

Many operations require that the user has rights to linked tables, in order for InterBase/Firebird to process updates.

- If foreign key constraints exist between two tables, then an UPDATE, DELETE or INSERT operation on the first table requires SELECT or REFERENCES privileges on the referenced table. *Tip:* Make it easy: if read security is not an issue, GRANT REFERENCES on the primary key table to PUBLIC. If you grant the REFERENCES privilege, it must, at a minimum, be granted to all columns of the primary key. When REFERENCES is granted to the entire table, columns that are not part of the primary key are not affected in any way. When a user defines a foreign key constraint on a table owned by someone else, InterBase/Firebird checks that the user has REFERENCES privileges on the referenced table. The privilege is used at runtime to verify that a value entered in a foreign key field is contained in the primary key table. You can grant REFERENCES privileges to roles.
- If there is a check constraint within a table, an UPDATE or INSERT operation also requires SELECT privileges on the same table.
- If a constraint includes one or more queries, an UPDATE or INSERT operation also requires SELECT privileges on the table or tables used in the SELECT.

IBExpert allows privileges to be granted on objects at the time of creation directly in the objects editor's Grants page (please refer to Table Editor / Grants page for further details). Dependencies upon or from other objects are also displayed in the individual object editors, to show visually any object interactions, which may need to be taken into consideration when assigning user permissions. Refer to Table Editor / Dependencies page for further information. All objects or a filtered selection of objects can be displayed and processed in the IBExpert Grant Manager.

Privileges can be granted to a role as well as to users or stored procedures, tables, views and triggers.

The GRANT statement can be used in gpre, DSQL and isql. The syntax is as follows:

```
GRANT privileges ON [TABLE] {tablename | viewname}
      TO {object|userlist [WITH GRANT OPTION]|GROUP UNIX_group}
   | EXECUTE ON PROCEDURE procname TO {object | userlist}
   | role_granted TO {PUBLIC | role_grantee_list}[WITH ADMIN OPTION];

<privileges> = ALL [PRIVILEGES] | privilege_list

<privilege_list> = {
     SELECT
   | DELETE
   | INSERT
   | UPDATE [(col [, col …])]
   | REFERENCES [(col [, col …])]
}[, privilege_list …]

<object> = {
     PROCEDURE procname
   | TRIGGER trigname
   | VIEW viewname
   | PUBLIC
}[, object …]
```

```
<userlist> = {
     [USER] username
   | rolename
   | UNIX_user
}[,userlist …]

<role_granted> = rolename [, rolename …]

<role_grantee_list> = [USER] username [, [USER] username …]
```

- **privilege_list** - Name of privilege to be granted; valid options are SELECT, DELETE, INSERT, UPDATE, and REFERENCES.
- **col** - Column to which the granted privileges apply.
- **tablename** - Name of an existing table for which granted privileges apply.
- **viewname** - Name of an existing view for which granted privileges apply.
- **GROUP unix_group** - On a UNIX system, the name of a group defined in /etc/group.
- **object** - Name of an existing procedure, trigger, or view; PUBLIC is also a permitted value.
- **userlist** - A user in the InterBase/Firebird security database (admin.ib or by default) or a rolename created with CREATE ROLE.
- **WITH GRANT OPTION** - Passes GRANT authority for privileges listed in the GRANT statement to userlist (please refer to GRANT AUTHORITY for further information).
- **rolename** - An existing role created with the CREATE ROLE statement
- **role_grantee_list** - A list of users to whom rolename is granted; users must be in the InterBase security database.
- **WITH ADMIN OPTION** - Passes grant authority for roles listed to role_grantee_list.

Important: In SQL statements passed to DSQL, omit the terminating semicolon. In embedded applications written in C and C++, and in isql, the semicolon is a terminating symbol for the statement, so it must be included.

To grant privileges to a group of users, create a role using the CREATE ROLE statement. Please refer to Role for details.

On UNIX systems, privileges can be granted to groups listed in /etc/groups and to any UNIX user listed in /etc/passwd on both the client and server, as well as to individual users and to roles.

Examples:

```
GRANT insert, update, delete
   ON customer
   TO Janet, John
   WITH GRANT OPTION;
```

or:

```
GRANT references
   ON customer
   TO PUBLIC;
```

If different levels of access are to be assigned to different objects and different people, separate GRANT statements have to be used.

This embedded SQL statement grants EXECUTE privileges for a procedure to another procedure and to a user:

```
EXEC SQL
   GRANT EXECUTE ON PROCEDURE GET_EMP_PROJ
   TO PROCEDURE ADD_EMP_PROJ, LUIS;
```

The following example creates a role called administrator, grants UPDATE privileges on table1 to that role, and then grants the role to user1, user2, and user3. These users then have UPDATE and REFERENCES privileges on table1:

```
CREATE ROLE administrator;
GRANT UPDATE ON table1 TO administrator;
GRANT administrator TO user1, user2, user3;
```

I.2.13 PREPARE

The PREPARE statement prepares a dynamic SQL (DSQL) statement for execution. Available in gpre.

The IBExpert SQL Editor toolbar icon Prepare Query or [Ctrl + F9] may also be used to prepare a query.

Syntax:

```
PREPARE [TRANSACTION transaction] statement
   [INTO SQL DESCRIPTOR xsqlda] FROM {:variable | 'string'};
```

- **TRANSACTION** - transaction name of the transaction under control of which the statement is executed.
- **statement** - Establishes an alias for the prepared statement that can be used by subsequent DESCRIBE and EXECUTE statements.
- **INTO xsqlda** - Specifies an XSQLDA to be filled in with the description of the select list columns in the prepared statement.
- **:variable | `string'** - DSQL statement to PREPARE; can be a host-language variable or a string literal.

PREPARE readies a DSQL statement for repeated execution by:

- Checking the statement for syntax errors.
- Determining data types of optionally specified dynamic parameters.
- Optimizing statement execution.
- Compiling the statement for execution by EXECUTE.

PREPARE is part of a group of statements that prepare DSQL statements for execution.

Statement	*Purpose*
PREPARE	Prepares a DSQL statement for execution.

DESCRIBE	Fills in the XSQLDA with information about the statement.
EXECUTE	Executes a previously prepared statement.
EXECUTE IMMEDIATE	Prepares a DSQL statement, executes it once, and discards it.

After a statement is prepared, it is available for execution as many times as necessary during the current session. To prepare and execute a statement only once, use EXECUTE IMMEDIATE.

STATEMENT establishes a symbolic name for the actual DSQL statement to prepare. It is not declared as a host language variable. Except for C programs, gpre does not distinguish between uppercase and lowercase in statement, treating "B" and "b" as the same character. For C programs, use the gpre -either_case switch to activate case sensitivity during preprocessing.

If the optional INTO clause is used, PREPARE also fills in the extended SQL descriptor area (XSQLDA) with information about the data type, length, and name of select list columns in the prepared statement. This clause is useful only when the statement to prepare is a SELECT.

Note: The DESCRIBE statement can be used instead of the INTO clause to fill in the XSQLDA for a select list.

The FROM clause specifies the actual DSQL statement to PREPARE. It can be a host language variable, or a quoted string literal. The DSQL statement to PREPARE can be any SQL data definition, data manipulation, or transaction-control statement.

Examples:

The following embedded SQL statement prepares a DSQL statement from a host variable statement. Because it uses the optional INTO clause, the assumption is that the DSQL statement in the host variable is a SELECT.

```
EXEC SQL
   PREPARE Q INTO xsqlda FROM :buf;
```

Note: The previous statement could also be prepared and described in the following manner:

```
EXEC SQL
   PREPARE Q FROM :buf;

EXEC SQL
   DESCRIBE Q INTO SQL DESCRIPTOR xsqlda;
```


I.2.14 REVOKE

REVOKE is the SQL statement, used to withdraw those rights already assigned to database users or objects for database objects. Rights can be revoked using the IBExpert Grant Manager, the relevant object editors' Grants pages, or the SQL Editor.

The following rules apply when revoking user privileges:

- Only the user who granted the privilege or the SYSDBA may revoke it.
- Revoking a privilege has no effect on any other privileges granted by other users. However, if multiple users have the ability to grant privileges, one user might have received a specific privilege from more than one source. If only one of them is revoked, the other remains in effect.
- If a privilege, which was originally granted using the WITH GRANT OPTION clause, is revoked, any subsequent users to which the same privilege had been granted in turn lose their privileges too.
- The ALL keyword can be used to revoke all granted privileges to an object, even if the user has not been granted all available privileges in the first place. REVOKE ALL however has no effect on the EXECUTE privilege, which must always be explicitly revoked.
- If a privilege is granted to all users using the PUBLIC option, this grant can only be revoked using the same PUBLIC option.

The SQL syntax is as follows:

```
REVOKE [GRANT OPTION FOR] privilege ON [TABLE] {tablename | viewname}
     FROM {object | userlist | rolelist | GROUP UNIX_group}
   | EXECUTE ON PROCEDURE procname FROM {object | userlist}
   | role_granted FROM {PUBLIC | role_grantee_list}};
<privileges> = ALL [PRIVILEGES] | privilege_list
<privilege_list> = {
     SELECT
   | DELETE
   | INSERT
   | UPDATE [(col [, col …])]
   | REFERENCES [(col [, col …])]
   }[, privilege_list …]
<object> = {
     PROCEDURE procname
   | TRIGGER trigname
   | VIEW viewname
   | PUBLIC
   }[, object …]
<userlist> = [USER] username [, [USER] username …]
<rolelist> = rolename [, rolename]
<role_granted> = rolename [, rolename …]
<role_grantee_list> = [USER] username [, [USER] username …]
```

- **privilege_list** - Name of privilege to be granted; valid options are SELECT, DELETE, INSERT, UPDATE, and REFERENCES.
- **GRANT OPTION FOR** - Removes grant authority for privileges listed in the REVOKE statement from userlist; cannot be used with object.
- **col** - Column for which the privilege is revoked.
- **tablename** - Name of an existing table for which privileges are revoked.
- **viewname** - Name of an existing view for which privileges are revoked.
- **GROUP unix_group** - On a UNIX system, the name of a group defined in /etc/group.

- **object -** Name of an existing database object from which privileges are to be revoked.
- **userlist** - A list of users from whom privileges are to be revoked.
- **rolename** - An existing role created with the `CREATE ROLE` statement.
- **role_grantee_list** - A list of users to whom `rolename` is granted; users must be in the InterBase security database (`admin.ib` by default).

For example, to revoke `INSERT` and `UPDATE` privileges from Janet and John:

```
REVOKE INSERT, UPDATE
   ON PROJ_DEPT_BUDGET
   FROM Janet, John
```

To revoke all privileges from every user, use the `PUBLIC` option, for example:

```
REVOKE ALL
   ON PROJ_DEPT_BUDGET
   FROM PUBLIC;
```

I.2.15 ROLLBACK

If a transaction's operations did not all complete successfully or satisfactorily, it is possible to roll back the transaction. A rollback restores the data to the state it was in before the transaction started. All changes made by insertions, updates and deletions are reversed.

The `ROLLBACK` is performed in IBExpert using the

icon or [Ctrl + Alt + R].

Rolling back can of course also be specified by issuing the following statement:

```
ROLLBACK [TRANSACTION name];
```

The transaction name is only required in embedded SQL applications using SQL or DSQL, where more than one transaction can be active at any one time.

It is important to note that when a transaction is rolled back, the changes performed by that transaction are not immediately deleted. Instead, InterBase flags the transaction associated with that entry as having been rolled back in the Transaction Inventory Page (TIP). Subsequent queries must then reconstruct the row using the version history.

When InterBase/Firebird performs a garbage collection or database sweep, the server detects that the row entry for the current version does not in fact contain the complete current version. It is then updated and the various data segments and version history relinked to ensure that the current version of the row is stored in the correct place, so that back versions do not need to be read each time.

I.2.16 SET

The SET statement includes the following:

- SET DATABASE statement
- SET GENERATOR statement
- SET NAMES statement
- SET SQL DIALECT statement
- SET STATISTICS statement
- SET TERM terminator
- SET TRANSACTION statement

SET DATABASE

The SET DATABASE command creates a so-called database handle when creating embedded SQL applications for a specified database. It is available in gpre.

As it is possible to access several databases with embedded SQL applications, the desired database can be explicitly specified with the aid of the handle. The SET DATABASE command is only required by embedded SQL programmers and is not necessary for programming the BDE.

SET DATABASE has the following syntax:

```
SET DATABASE DB_Handle =
 [GLOBAL | STATIC | EXTERN]
 [COMPILETIME] [FILENAME] "<db_Name>"
 [USER "UserName" PASSWORD "PassString"]
 [RUNTIME] [FILENAME] {"<DB_Name>"|:VarDB}
 [USER {"Name"| :VarName}
  PASSWORD {"Password"| :VarPassWord=};
```

- **DB_Handle** - This is the name of the database handle, defined by the application. It is an alias (usually an abbreviation) for a specified database. It must be unique within the program, follow the file syntax conventions for the server where the database resides, and be used in subsequent SQL statements that support database handles. For example, they can be used in subsequent CONNECT, COMMIT and ROLLBACK statements, or can also be used within transactions to differentiate table names when two or more attached databases contain tables with the same names.

The optional parameters GLOBAL, STATIC and EXTERN can be used to specify the validity range of the database declaration. Following rules apply for the validity range:

- **Global**: The database declaration is visible for all modules (default).
- **Static**: Limits the database declaration to the current module (i.e. limit the database handle availability to the code module where the handle is declared).
- **Extern**: References a global database handle in another module, rather than actually declaring a new handle.
- **Compiletime** - Identifies the database used to look up column references during preprocessing. If only one database is specified in SET DATABASE, it is used both at runtime and compiletime.

- **Runtime** - Specifies a database to use at runtime if different thatn the one specified for use during preprocessing. And if necessary, different standard users can be specified for both situations. InterBase/Firebird sets the same database for runtime and development time as standard, if the optional parameters `COMPILETIME` and `RUNTIME` are not used.
- **<DB_Name>** - Represents a file specificationfor the database to associate with `db_handle`. It is platform-specific.
- **:VarDB** - This is the host-language variable containing a database specification, user name, or password.
- **USER** and **PASSWORD** - Valid user name and password on the server where the database resided. Required for PC client attachments, optional for all others.

Example:

```
EXEC SQL
 SET DATABASE EMPDB = 'employee.gdb'
 COMPILETIME "Test.gdb"
 RUNTIME :db_runtime;
```

SET GENERATOR

The `SET GENERATOR` command sets a new start value for an existing generator.

The `SET GENERATOR` command syntax is composed as follows:

```
SET GENERATOR Gen_Name TO int_value;
```

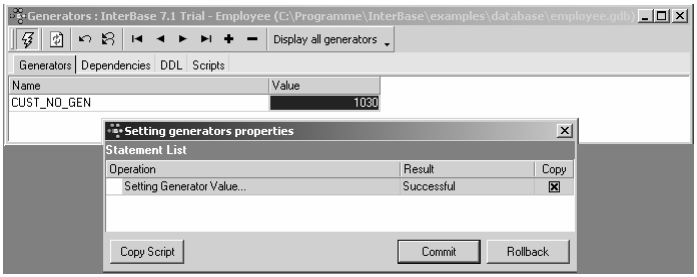
As soon as the function `GEN_ID()` enters or alters a value in a table column, this value is calculated from the `int_value` plus the increment defined by the `GEN_ID()` `step` parameter.

Example:

```
SET GENERATOR CUST_ID_GEN TO 1030;
```

Assuming that the `step` parameter in the function `GEN_ID()` is given the value `1`, the next customer would receive the customer number `1031`.

This statement can also be easily and quickly performed using IBExpert's Generator Editor (please refer to Alter Generator for further information):



SET NAMES

The SET NAMES statement specifies an active character set to use for subsequent database attachments. Available in gpre, and isql.

Syntax:

```
SET NAMES [charset | :var];
```

- **charset** - Name of a character set that identifies the active character set for a given process; default: NONE.
- **:var** - Host variable containing string identifying a known character set name. Must be declared as a character set name. SQL only.

SET NAMES specifies the character set to use for subsequent database attachments in an application. It enables the server to translate between the default character set for a database on the server and the character set used by an application on the client.

SET NAMES must appear before the SET DATABASE and CONNECT statements it is to affect.

Tip: Use a host-language variable with SET NAMES in an embedded application to specify a character set interactively.

Choice of character sets limits possible collation orders to a subset of all available collation orders. Given a specific character set, a specific collation order can be specified when data is selected, inserted, or updated in a column. If a default character set is not specified, the character set defaults to NONE.

Using character set NONE means that there is no character set assumption for columns; data is stored and retrieved just as it is originally entered. You can load any character set into a column defined with NONE, but you cannot load that same data into another column that has been defined with a different character set. No transliteration is performed between the source and destination character sets, so in most cases, errors occur during assignment.

Example:

The following statements demonstrate the use of SET NAMES in an embedded SQL application:

```
EXEC SQL
   SET NAMES ISO8859_1;

EXEC SQL
   SET DATABASE DB1 = 'employee.gdb';

EXEC SQL
   CONNECT;
```

The next statements demonstrate the use of SET NAMES in isql:

```
SET NAMES LATIN1;
   CONNECT 'employee.gdb';
```

SET SQL DIALECT

SET SQL DIALECT declares the SQL dialect for database access.

n is the SQL dialect type, either 1, 2, or 3. If no dialect is specified, the default dialect is set to that of the specified compile-time database. If the default dialect is different than the one specified by the user, a warning is generated and the default dialect is set to the user-specified value. Available in gpre and isql.

Syntax:

```
SET SQL DIALECT n;
```

where n is the SQL dialect type, either 1, 2, or 3.

SQL Dialect:	*Used for:*
1	InterBase 5 and earlier compatibility.
2	Transitional dialect used to flag changes when migrating from dialect 1 to dialect 3.
3	Current InterBase/Firebird; allows you to use delimited identifiers, exact NUMERICs, and DATE, TIME, and TIMESTAMP data types.

SET STATISTICS

SET STATISTICS enables the selectivity of an index to be recomputed. Index selectivity is a calculation, based on the number of distinct rows in a table, which is made by the InterBase/Firebird optimizer when a table is accessed. It is cached in memory, where the optimizer can access it to calculate the optimal retrieval plan for a given query. For tables where the number of duplicate values in indexed columns radically increases or decreases, periodically recomputing index selectivity can improve performance. Available in gpre, DSQL, and isql.

Only the creator of an index can use SET STATISTICS .

Note: SET STATISTICS does not rebuild an index. To rebuild an index, use ALTER INDEX.

Syntax:

```
SET STATISTICS INDEX name;
```

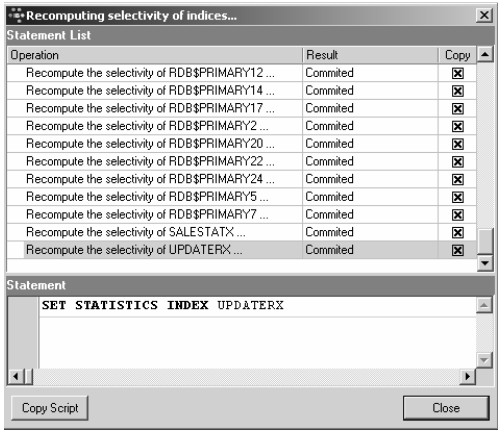
- **name** - Name of an existing index for which to recompute selectivity.

Example:

The following embedded SQL statement recomputes the selectivity for an index:

```
EXEC SQL
   SET STATISTICS INDEX MINSALX;
```

It is possible to recompute the selectivity for all indices using the IBExpert Database menu item Recompute Selectivity of all Indices.



SET TRANSACTION

`SET TRANSACTION` starts a transaction, and optionally specifies its database access, lock conflict behavior, and level of interaction with other concurrent transactions accessing the same data. It can also reserve locks for tables. As an alternative to reserving tables, multiple database SQL applications can restrict a transaction's access to a subset of connected databases. Available in gpre, DSQL and isql.

Important: applications preprocessed with the gpre `-manual` switch must explicitly start each transaction with a `SET TRANSACTION` statement.

Syntax:

```
SET TRANSACTION [NAME transaction]
   [READ WRITE | READ ONLY]
   [WAIT | NO WAIT]
   [[ISOLATION LEVEL] ]
   [RESERVING reserving_clause
      | USING dbhandle [, dbhandle …]];
<reserving_clause> = table [, table …]
   [FOR [SHARED | PROTECTED] {READ | WRITE}] [, reserving_clause]
```

- **NAME transaction** - Specifies the name for this transaction. Transaction is a previously declared and initialized host-language variable. SQL only.
- **READ WRITE [Default]** - Specifies that the transaction can read and write to tables.
- **READ ONLY** - Specifies that the transaction can only read tables.
- **WAIT [Default]** - Specifies that a transaction wait for access if it encounters a lock conflict with another transaction.

- **NO WAIT** - Specifies that a transaction immediately return an error if it encounters a lock conflict.
- **ISOLATION LEVEL** - Specifies the isolation level for this transaction when attempting to access the same tables as other simultaneous transactions; default: SNAPSHOT.
- **RESERVING reserving_clause** - Reserves lock for tables at transaction start.
- **USING dbhandle [, dbhandle ...]** - Limits database access to a subset of available databases; SQL only.

Examples:

The following embedded SQL statement sets up the default transaction with an isolation level of READ COMMITTED. If the transaction encounters an update conflict, it waits to get control until the first (locking) transaction is committed or rolled back.

```
EXEC SQL
   SET TRANSACTION WAIT ISOLATION LEVEL READ COMMITTED;
```

The next embedded SQL statement starts a named transaction:

```
EXEC SQL
   SET TRANSACTION NAME T1 READ COMMITTED;
```

The following embedded SQL statement reserves three tables:

```
EXEC SQL
   SET TRANSACTION NAME TR1
   ISOLATION LEVEL READ COMMITTED
   NO RECORD_VERSION WAIT
   RESERVING TABLE1, TABLE2 FOR SHARED WRITE,
      TABLE3 FOR PROTECTED WRITE;
```

I.2.17 WHENEVER

WHENEVER traps for SQLCODE errors and warnings. Every executable SQL statement returns a SQLCODE value to indicate its success or failure. If SQLCODE is zero, statement execution is successful. A non-zero value indicates an error, warning, or not found condition. Available in gpre.

If the appropriate condition is trapped, WHENEVER can:

- Use GOTO label to jump to an error-handling routine in an application.
- Use CONTINUE to ignore the condition.

WHENEVER can help limit the size of an application, because the application can use a single suite of routines for handling all errors and warnings.

WHENEVER statements should precede any SQL statement that can result in an error. Each condition to trap for requires a separate WHENEVER statement. If WHENEVER is omitted for a particular condition, it is not trapped.

Tip Precede error-handling routines with `WHENEVER … CONTINUE` statements to prevent the possibility of infinite looping in the error-handling routines.

Syntax:

```
WHENEVER {NOT FOUND | SQLERROR | SQLWARNING}
   {GOTO label | CONTINUE};
```

- **NOT FOUND** - Traps `SQLCODE = 100`, no qualifying rows found for the executed statement.
- **SQLERROR** - Traps `SQLCODE < 0`, failed statement.
- **SQLWARNING** - Traps `SQLCODE > 0 AND < 100`, system warning or informational message.
- **GOTO label** - Jumps to program location specified by label when a warning or error occurs.
- **CONTINUE** - Ignores the warning or error and attempts to continue processing.

Example:

In the following code from an embedded SQL application, three `WHENEVER` statements determine which label to branch to for error and warning handling:

```
EXEC SQL
   WHENEVER SQLERROR GO TO Error; /* Trap all errors. */

EXEC SQL
   WHENEVER NOT FOUND GO TO AllDone; /* Trap SQLCODE = 100 */

EXEC SQL
   WHENEVER SQLWARNING CONTINUE; /* Ignore all warnings. */
```

I.3 DML – Data Manipulation Language

DML is the abbreviation for Data Manipulation Language. DML is a collection of SQL commands that can be used to manipulate a database's data.

DML is part of the SQL language commands, which execute queries with database objects and changes to their contents. The various DML commands can be used to create, edit, evaluate and delete data in a database.

DML commands are a subarea of SQL; the range of the SQL language is composed of DML and DDL together.

I.3.1 SIUD

SIUD is the abbreviation for `SELECT`, `INSERT`, `UPATE`, `DELETE`, which are the four DML commands used for data manipulation.

SELECT

Retrieves a single row that satisfies the requirements of the search condition. The same as standard singleton SELECT, with some differences in syntax. Available in triggers and stored procedures.

```
<select_expr> = <select_clause> <from_clause>
[<where_clause>] [<group_by_clause>]
[<having_clause>]
[<union_expression>] [<plan_clause>]
[<ordering_clause>]
<into_clause>;
```

Description In a stored procedure, use the SELECT statement with an INTO clause to retrieve a single row value from the database and assign it to a host variable. The SELECT statement must return at most one row from the database, like a standard singleton SELECT. The INTO clause is required and must be the last clause in the statement.

The INTO clause comes at the end of the SELECT statement to allow the use of UNION operators. UNION is not allowed in singleton SELECT statements in embedded SQL.

Example:

The following statement is a standard singleton SELECT statement in an embedded application:

```
EXEC SQL
   SELECT SUM(BUDGET), AVG(BUDGET)
   INTO :TOT_BUDGET, :AVG_BUDGET
   FROM DEPARTMENT
   WHERE HEAD_DEPT = :HEAD_DEPT
```

To use the above SELECT statement in a procedure, move the INTO clause to the end as follows:

```
SELECT SUM(BUDGET), AVG(BUDGET)
   FROM DEPARTMENT
   WHERE HEAD_DEPT = :HEAD_DEPT
   INTO :TOT_BUDGET, :AVG_BUDGET;
```

INSERT

Adds one or more new rows to a specified table. Available in gpre, DSQL, and isql.

Syntax:

```
INSERT [TRANSACTION transaction] INTO object [(col [, col …])]
   {VALUES (val [, val …]) | select_expr};

<object> = tablename | viewname
```

```
<val> = {:variable | constant | expr
   | function | udf ([val [, val …]])
   | NULL | USER | RDB$DB_KEY | ?} [COLLATE collation]

<constant> = num | 'string' | charsetname 'string'

<function> = CAST (val AS datatype)
   | UPPER (val)
   | GEN_ID (generator, val)
```

Argument	*Description*
`expr`	A valid SQL expression that results in a single column value.
`select_expr`	A `SELECT` that returns zero or more rows and where the number of columns in each row is the same as the number of items to be inserted.

Notes on the `INSERT` statement:

- In SQL and isql, you cannot use val as a parameter placeholder (like "?").
- In DSQL and isql, val cannot be a variable.
- You cannot specify a COLLATE clause for Blob columns.

Important: In SQL statements passed to DSQL, omit the terminating semicolon. In embedded applications written in C and C++, and in isql, the semicolon is a terminating symbol for the statement, so it must be included.

Argument	*Description*
`TRANSACTION` *`transaction`*	Name of the transaction that controls the execution of the `INSERT`.
`INTO` *`object`*	Name of an existing table or view into which to insert data.
`col`	Name of an existing column in a table or view into which to insert values.
`VALUES` *`(val [, val …])`*	Lists values to insert into the table or view; values must be listed in the same order as the target columns.
`select_expr`	Query that returns row values to insert into target columns.

Description:

`INSERT` stores one or more new rows of data in an existing table or view. `INSERT` is one of the database privileges controlled by the `GRANT` and `REVOKE` statements.

Values are inserted into a row in column order unless an optional list of target columns is provided. If the target list of columns is a subset of available columns, default or NULL values are automatically stored in all unlisted columns.

If the optional list of target columns is omitted, the `VALUES` clause must provide values to insert into all columns in the table.

To insert a single row of data, the VALUES clause should include a specific list of values to insert.

To insert multiple rows of data, specify a select_expr that retrieves existing data from another table to insert into this one. The selected columns must correspond to the columns listed for insert.

Important:

It is legal to select from the same table into which insertions are made, but this practice is not advised because it may result in infinite row insertions.

The TRANSACTION clause can be used in multiple transaction SQL applications to specify which transaction controls the INSERT operation. The TRANSACTION clause is not available in DSQL or isql.

Examples:

The following statement, from an embedded SQL application, adds a row to a table, assigning values from host-language variables to two columns:

```
EXEC SQL
   INSERT INTO EMPLOYEE_PROJECT (EMP_NO, PROJ_ID)
      VALUES (:emp_no, :proj_id);
```

The next isql statement specifies values to insert into a table with a SELECT statement:

```
INSERT INTO PROJECTS
   SELECT * FROM NEW_PROJECTS
   WHERE NEW_PROJECTS.START_DATE > '6-JUN-1994';
```

UPDATE

Changes the data in all or part of an existing row in a table, view, or active set of a cursor. Available in gpre, DSQL, and isql.

Syntax SQL form:

```
UPDATE [TRANSACTION transaction] {table | view}
   SET col = val [, col = val …]
   [WHERE search_condition | WHERE CURRENT OF cursor]
   [ORDER BY order_list]
   [ROWS value [TO upper_value] [BY step_value][PERCENT][WITH TIES]];
```

DSQL and isql form:

```
UPDATE {table | view}
   SET col = val [, col = val …]
   [WHERE search_condition
   [ORDER BY order_list]
   [ROWS value [TO upper_value] [BY step_value][PERCENT][WITH TIES]]
```

```
<val> = {
   col [array_dim]
   | :variable
   | constant
   | expr
   | function
   | udf ([val [, val …]])
   | NULL
   | USER
   | ?}
   [COLLATE collation]

<array_dim> = [[x:]y [, [x:]y …]]

<constant> = num | 'string' | charsetname 'string'

<function> = CAST (val AS datatype)
   | UPPER (val)
   | GEN_ID (generator, val)
```

<expr> = A valid SQL expression that results in a single value.
<search_condition> = See CREATE TABLE for a full description.

Notes on the UPDATE statement:

- In SQL and isql, you cannot use val as a parameter placeholder (like "?").
- In DSQL and isql, val cannot be a variable.
- You cannot specify a COLLATE clause for Blob columns.

Argument	*Description*
TRANSACTION *transaction*	Name of the transaction under control of which the statement is executed.
table | *view*	Name of an existing table or view to update.
SET *col* = *val*	Specifies the columns to change and the values to assign to those columns.
WHERE *search_condition*	Searched update only; specifies the conditions a row must meet to be modified.
WHERE CURRENT OF *cursor*	Positioned update only; specifies that the current row of a cursor's active set is to be modified. • Not available in DSQL and isql.
ORDER BY *order_list*	Specifies columns to order, either by column name or ordinal number in the query, and the sort order (ASC or DESC) for the returned rows.
ROWS *value* [TO *upper_value*] [BY *step_value*]	• *value* is the total number of rows to return if used by itself. • *value* is the starting row number to return if used with TO. • *value* is the percent if used with PERCENT. • *upper_value* is the last row or highest percent to return.

[PERCENT][WITH TIES]	• If *step_value = n*, returns every nth row, or n percent rows. • PERCENT causes all previous ROWS values to be interpreted as percents. • WITH TIES returns additional duplicate rows when the *last* value in the ordered sequence is the same as values in subsequent rows of the result set; must be used in conjunction with ORDER BY.

Description:

UPDATE modifies one or more existing rows in a table or view. UPDATE is one of the database privileges controlled by GRANT and REVOKE.

For searched updates, the optional WHERE clause can be used to restrict updates to a subset of rows in the table. Searched updates cannot update array slices.

Important:

Without a WHERE clause, a searched update modifies all rows in a table.

When performing a positioned update with a cursor, the WHERE CURRENT OF clause must be specified to update one row at a time in the active set.

Note: When updating a blob column, UPDATE replaces the entire blob with a new value.

Examples:

The following isql statement modifies a column for all rows in a table:

```
UPDATE CITIES
   SET POPULATION = POPULATION * 1.03;
```

The next embedded SQL statement uses a WHERE clause to restrict column modification to a subset of rows:

```
EXEC SQL
   UPDATE PROJECT
   SET PROJ_DESC = :blob_id
   WHERE PROJ_ID = :proj_id;
```


DELETE

Removes rows in a table or in the active set of a cursor. Available in gpre, DSQL, and isql.

Syntax SQL and DSQL form:

Important: Omit the terminating semicolon for DSQL.

```
DELETE [TRANSACTION transaction] FROM table
   {[WHERE search_condition] | WHERE CURRENT OF cursor}
   [ORDER BY order_list]
   [ROWS value [TO upper_value] [BY step_value][PERCENT][WITH TIES]];
```

<search_condition> = Search condition as specified in SELECT.

isql form:

```
DELETE FROM TABLE [WHERE search_condition];
```

Argument	*Description*
TRANSACTION *transaction*	Name of the transaction under control of which the statement is executed; SQL only.
table	Name of the table from which to delete rows.
WHERE *search_condition*	Search condition that specifies the rows to delete; without this clause, DELETE affects all rows in the specified table or view.
WHERE CURRENT OF *cursor*	Specifies that the current row in the active set of cursor is to be deleted.
ORDER BY *order_list*	Specifies columns to order, either by column name or ordinal number in the query, and the sort order (ASC or DESC) for the returned rows.
ROWS *value* [TO *upper_value*] [BY *step_value*] [PERCENT][WITH TIES]	• *value* is the total number of rows to return if used by itself • *value* is the starting row number to return if used with TO • *value* is the percent if used with PERCENT • upper_value is the last row or highest percent to return • If *step_value* = n, returns every nth row, or n percent rows • PERCENT causes all previous ROWS values to be interpreted as percents • WITH TIES returns additional duplicate rows when the last value in the ordered sequence is the same as values in subsequent rows of the result set; must be used in conjunction with ORDER BY.

DELETE specifies one or more rows to delete from a table or updatable view. DELETE is one of the database privileges controlled by the GRANT and REVOKE statements.

The TRANSACTION clause can be used in multiple transaction SQL applications to specify which transaction controls the DELETE operation. The TRANSACTION clause is not available in DSQL or isql.

For searched deletions, the optional WHERE clause can be used to restrict deletions to a subset of rows in the table.

Important:

Without a WHERE clause, a searched delete removes all rows from a table.

When performing a positioned delete with a cursor, the WHERE CURRENT OF clause must be specified to delete one row at a time from the active set.

Examples:

The following isql statement deletes all rows in a table:

```
DELETE FROM EMPLOYEE_PROJECT;
```

The next embedded SQL statement is a searched delete in an embedded application. It deletes all rows where a host-language variable equals a column value.

```
EXEC SQL
   DELETE FROM SALARY_HISTORY
   WHERE EMP_NO = :emp_num;
```

The following embedded SQL statements use a cursor and the WHERE CURRENT OF option to delete rows from CITIES with a population less than the host variable, min_pop. They declare and open a cursor that finds qualifying cities, fetch rows into the cursor, and delete the current row pointed to by the cursor.

```
EXEC SQL
   DECLARE SMALL_CITIES CURSOR FOR
   SELECT CITY, STATE
   FROM CITIES
   WHERE POPULATION < :min_pop;

EXEC SQL
   OPEN SMALL_CITIES;

EXEC SQL
   FETCH SMALL_CITIES INTO :cityname, :statecode;
   WHILE (!SQLCODE)
      {EXEC SQL
         DELETE FROM CITIES
         WHERE CURRENT OF SMALL_CITIES;
      EXEC SQL
         FETCH SMALL_CITIES INTO :cityname, :statecode;}
EXEC SQL
   CLOSE SMALL_CITIES;
```


I.4 Stored Procedure and Trigger Language

The InterBase/Firebird procedure and trigger language includes all the constructs of a basic structured programming language, as well as statements unique to working with table data. The SQL SELECT, INSERT, UPDATE and DELETE statements can be used in stored procedures exactly as they are used in a query, with only minor syntax changes (refer to Using DML statements). Local variables or input parameters can be used for all of these statements in any place that a literal value is allowed.

Other statements that are specific to stored procedures include, among others, error handling and raising exceptions. Please refer to the relevant sections for further information.

Note that the string concatenation operator in InterBase/Firebird procedure and trigger language is || (a double vertical bar, or pipe), and not the + that is used in many programming languages. Please refer to concatenation of strings for further information.

I.4.1 Supported Firebird 2 features

Since IBExpert version 2005.03.12 the following Firebird 2 features are also supported:

- `DECLARE <cursor_name> CURSOR FOR ...`
- `OPEN <cursor_name>`
- `FETCH <cursor_name> INTO ...`
- `CLOSE <cursor_name>`
- `LEAVE <label>`
- `NEXT VALUE FOR <generator>`

I.4.2 Using DML statements

The SQL Data Manipulation Language (DML), consists primarily of the `SELECT`, `INSERT`, `UPDATE` and `DELETE` statements.

Statements that are not recognized or permitted in the stored procedures and trigger language include DDL statements such as `CREATE`, `ALTER`, `DROP`, and `SET` as well as statements such as `GRANT`, `REVOKE`, `COMMIT`, and `ROLLBACK`.

Wherever a literal value is specified in an `INSERT`, `UPDATE` or `DELETE` statement, an input or local variable can be substituted in place of this literal. For example, variables can be used for the values to be inserted into a new row, or the new values in an `UPDATE` statement. They can also be used in a `WHERE` clause, to specify the rows that are to be updated or deleted.

I.4.3 Using SELECT statements

InterBase/Firebird supports an extension to the standard `SELECT` statement, to solve the problem of what to do with the results when using a `SELECT` statement inside a stored procedure. The `INTO` clause appoints variables that receive the results of the `SELECT` statement. The syntax is as follows:


```
SELECT <result1, result2, ..., resultN>
FROM ...
WHERE ...
GROUP BY ...
INTO : <Variable1, : Variable2, ..., VariableN>;
```

The `INTO` clause must be the final clause in the `SELECT` statement. A variable must be given for each result generated by the statement. *Important*: this form of `SELECT` statement can generate only one row. Therefore the `ORDER BY` clause is unnecessary here.

To use a SELECT that generates more than one row within a stored procedure, use the FOR SELECT statement.

I.4.4 SET TERM terminator or terminating character

Normally InterBase processes a script step by step and separates two statements by a semicolon. Each statement between two semicolons is parsed, interpreted, converted into an internal format and executed. This is not possible in the case of stored procedures or triggers where there are often multiple commands which need to be successively executed, i.e. there are several semicolons in their source codes. So if CREATE PROCEDURE … was called, InterBase/Firebird assumes that the command has finished when it arrives at the first semi colon.

In order for InterBase(Firebird to correctly interpret and transfer a stored procedure to the database, it is necessary to temporarily alter the terminating character using the SET TERM statement. The syntax for this is as follows (Although when using the IBExpert templates this is not necessary, as IBExpert automatically inserts the SET TERM command):

```
SET TERM NEW_TERMINATOR OLD_TERMINATOR
```

Example:

```
SET TERM ^;
CREATE PROCEDURE NAME
   AS
     BEGIN
     <procedure body>;
   END^
SET TERM ;^
```

Before the first SET TERM statement appears, InterBase/Firebird regards the semicolon as the statement terminating character and interprets and converts the script code up until each semicolon.

Following the first SET TERM statement, the terminator is switched and all following semicolons are no longer interpreted as terminators. The CREATE PROCEDURE statement is then treated as one statement up until the new terminating character, and parsed and interpreted.

The final SET TERM statement is necessary to change the terminating character back to a semicolon, using the syntax:

```
SET TERM OLD_TERMINATOR NEW_TERMINATOR
```

(refer to above example: SET TERM ;^).

The statement must be concluded by the previously defined temporary termination character. This concluding statement is again interpreted as a statement between the two last termination characters. Finally the semicolon becomes the termination character for use in further script commands.

It is irrelevant which character is used to replace the semi colon; however it should be a seldom-used sign to prevent conflicts e.g. ^, and not * or + (used in mathematical formulae) or ! (this is used for "not equal": A!=B).

I.4.5 SUSPEND

SUSPEND is used in stored procedures; it acts as if it was a data set, i.e. returns the named data set visually as a result.

It suspends procedure execution until the next FETCH is issued by the calling application and returns output values, if there are any, to the calling application. It prevents the stored procedure from terminating until the client has fetched all the results. This statement is not recommended for executable procedures.

I.4.6 BEGIN and END statement

As well as defining the contents of the stored procedure, these keywords also delimit a block of statements which then executes as a single statement. This means that BEGIN and END can be used to enclosed several statements and so form a simple compound statement. A semicolon should not be placed after either of these words.

I.4.7 DECLARE VARIABLE

Please refer to local variables.

I.4.8 IF THEN ELSE

InterBase/Firebird supports simple assignment statements in the form:

```
variable = expression;
```

The variable can be an input or output parameter, or a local variable defined in a DECLARE VARIABLE statement. The expression needs to be concluded with a semicolon. The syntax for the IF statement is as follows:

```
IF <conditional_test>
THEN
<statements>;
ELSE
<statements>;
```


Any of the standard comparison operators available in SQL an be used (please refer to comparison operators for a full list).

The value can be a constant or one of the input parameters, output parameters or local variables used in the procedure.

If a single statement is placed after the THEN or ELSE clauses, it should be terminated with a semicolon. If multiple statements need to be placed after one of these clauses, use the BEGIN and END keywords as follows:

```
IF <conditional_test> THEN
BEGIN
<statement1>;
<statement2>;
...
<statementN>;
END
ELSE
etc.;
```

I.4.9 WHILE and DO

The `WHILE … DO` statement provides a looping capability. The syntax for this statement is as follows:

```
WHILE
<conditional_test>
DO
<statements>;
```

InterBase/Firebird evaluates the conditional test. If it is `TRUE`, the statements following the `WHILE` are executed. If it is `FALSE`, the statements are ignored. If only one statement is placed after the `DO` clause, it should be terminated with a semicolon. If multiple statements are used after one of these clauses, use the `BEGIN` and `END` keywords. Brackets need to be put around the conditional test.

I.5 Comparison Operators

Comparison operators for use in conditional clauses:

Conditional Test	*Description*
`value = value`	Equal to
`value < value`	Less than
`value > value`	Greater than
`value <= value`	Less than or equal to
`value >= value`	Greater than or equal to
`value !< value`	Not less than
`value !> value`	Not greater than
`value <> value`	Not equal to
`value != value`	Not equal to
`value LIKE value`	Wildcard search, use '%' for 0 or more characters and '_' for one character only
`value BETWEEN value AND value`	Within an inclusive range
`value IN (value, ...`	One of the elements in a list

value)	
value IS NULL	One of the elements in a list
value IS NOT NULL	One of the elements in a list
value CONTAINING value	Includes
value STARTING WITH value	Begins with

I.6 JOIN

In practice it seldom occurs that all relevant information can be found in a single database table. It is much more often the case that the data required is distributed across several tables and linked by relations. Indeed, information in a normalized database *should* be spread across multiple tables!

In a fully normalized database, the vast majority of tables have a primary key consisting of one or two columns only. If a referential integrity relationship exists, these primary key columns are replicated in other tables to ensure consistency in the data. These are the columns that allow you to establish logical links between these tables. When queries are performed, tables are commonly joined on these columns.

There is actually no restriction by design to the number of tables that may be joined. However the task of joining tables is exponential in relation to the number of tables in the join. The largest practical number of tables in a join is about 16, but experiment with your application and a realistic volume of data to find the most complex join that has an acceptable performance.

When you establish a join, InterBase/Firebird looks for matching values in the designated columns of each table. It does not care if a value appears once on one side of the join and multiple times on the other side, as is often the case.

In this instance, InterBase/Firebird joins each matching row in `TableB` to the single matching row in `TableA`, thereby creating what is known as a virtual row. Each `TableB` row can logically be linked to a single unambiguous row in `TableA`.

InterBase/Firebird also provides options for establishing a relationship where a value can appear on one side of the join instead of both. This is known as an `OUTER JOIN`, (please refer to `OUTER JOIN` for further information and examples).

The following statement selects from both `TableA` and `TableB` tables:

```
SELECT column_list
FROM TableA, TableB;
```

When you select from two or more tables, these tables are normally joined on a common column. For example, you might join `TableA` and `TableB` tables on the column that is common to each of them, the `TableA_ID`.

Theoretically it is not necessary to specify a join column. If you do not specify one, InterBase/Firebird performs a Cartesian product between the two tables, joining each

row in one table to each row in the other. So, for example, if the first table had 100 rows, and the second had 20, the result set would have 2000 rows. Such a join normally makes no sense because the row information in one table is not logically related to the row information in the other table, except where column and field values are shared between the tables.

InterBase/Firebird does not prevent you from establishing a meaningless join. You can issue an SQL statement that joins, for example, `Orders.PaymentMethod` with `Customer.Country`, and InterBase/Firebird processes the statement! But the result set is always empty because there are no matching values in either column.

JOIN syntax:

InterBase/Firebird currently supports two methods to link two or more tables via a common column:

- the traditional SQL syntax, and
- the SQL '92 syntax.

The traditional SQL syntax integrates the link in the `WHERE` clause:

```
SELECT <ColumnList>

     FROM Table1 Synonym1 , Table2 Synonym2

     WHERE Synonym1.JoinColumn = Synonym2.JoinColumn

     AND <Other_WHERE_Conditions> ;
```

The following example illustrates this syntax:

```
SELECT C.Name, C.Country, O.OrderID, O.SaleDate, O.TotalInvoice
     FROM Customer C, Orders O

     WHERE C.CustomerID = O.CustomerID
    AND C.Country != 'U.S.A.'

     ORDER BY C.Name, O.OrderID;
```

As opposed to traditional SQL syntax, the SQL 92 syntax detaches the link from the `WHERE` clause and relocates it in the `FROM` clause, i.e. that area, in which the tables to be used are defined:

```
SELECT <ColumnList>

     FROM Table1 Alias1 JOIN Table2 Alias2

              ON Alias1.Column = Alias2.Column

     WHERE <Where_Conditions> ;
```

Example:

```
SELECT C.Name, C.Country, O.OrderID, O.SaleDate, O.TotalInvoice

     FROM Customer C JOIN Orders O

         ON   C.CustomerID = O.CustomerID )

     WHERE C.Country != 'U.S.A.'

   ORDERBY C.Name, O.OrderID;
```

Either syntax can be used at any time; they are virtually interchangeable. The difference is that the SQL 92 syntax permits `OUTER JOIN`s, whereas the traditional syntax does not.

Specifying columns and rows:

When two or more tables are joined, rows can be included from either table in the result. It is also possible to specify `WHERE` conditions to limit the rows in either table that are considered for the join. For example, the following statement asks for customers in Florida who placed orders in 1994 with a total invoice of more than $5,000 for the order:

```
SELECT C.Name, C.City, O.SaleDate, O.TotalInvoice
FROM Customer C JOIN Orders O
ON C.CustomerID = O.CustomerID
WHERE C.State_Province = 'FL'
AND O.SaleDate BETWEEN '1/1/94' AND '12/31/94'
AND O.TotalInvoice > 5000;
```

Please refer to Joining more that two tables for further information.

I.6.1 INNER JOIN

When you join two tables, the result set includes only those rows where the joining value appears in both tables.

Syntax:

```
TableA JOIN TableB
```

The join applies to the table written to the left of the command.

For example, the following query joins `Stock` to `LineItem` to find out many orders included each stock item:

```
SELECT S.StockID, COUNT( L.OrderID )
FROM Stock S JOIN Lineitem L
ON S.StockID = L.StockID
GROUP BY S.StockID
```

From a theoretical standpoint, this is known as an INNER JOIN, but the INNER keyword is optional. What if you also want to include those stock items that have not yet been ordered, so that the result set shows all stock items. These items do not appear in the LineItem table at all. The solution lies in performing an OUTER JOIN. An outer join includes every column in one table and a subset of columns in the other table.

When your tables are linked in a referential relationship on a foreign key column, only the LEFT OUTER JOIN usually makes sense. For example, every order includes a customer from the Customer table. If you join Customer to Orders with a RIGHT OUTER JOIN, the result is the same as if you had performed an INNER JOIN.

I.6.2 OUTER JOIN

When you join two tables, the result set includes only those rows where the joining value appears in both tables.

There are three types of outer joins:

SQL92 syntax permits outer joins, whereas the traditional syntax does not.

Types of outer joins

- LEFT OUTER JOIN, which includes all rows from the table on the left side of the join expression
- RIGHT OUTER JOIN, which includes all rows from the table on the right side of the join expression.
- FULL OUTER JOIN, which includes all rows from both tables.

Syntax:

```
TableA LEFT OUTER JOIN TableB
```

The join applies to the table written to the left of the command.

```
TableA RIGHT OUTER JOIN TableB
```

The join applies to the table written to the right of the command.

When your tables are linked in a referential relationship on a foreign key column, only the LEFT OUTER JOIN usually makes sense. For example, every order includes a customer from the Customer table. If you join Customer to Orders with a RIGHT OUTER JOIN, the result is the same as if you had performed an INNER JOIN.

The following query modifies the preceding example to include all stock items, even the one that have not yet been ordered:

```
SELECT S.StockID, COUNT( L.OrderID )
FROM Stock S LEFT OUTER JOIN Lineitem L
ON S.StockID = L.StockID
GROUP BY S.StockID
```

Adding selection criteria:

If two tables are joined using an outer join, and there are also selection criteria in the table where the inclusion operator is placed, it would appear as first glance that you are asking two conflicting questions.

Consider the following query, which asks for the value of all orders placed by customers located in California, including those customers who might not have placed an order.

```
SELECT C.Name, SUM( O.TotalInvoice )
FROM Customer C LEFT OUTER JOIN Orders O
ON C.CustomerID = O.CustomerID
WHERE C.State_Province = 'CA'
GROUP BY C.Name;
```

On the one hand, the `LEFT OUTER JOIN` is asking InterBase/Firebird to include all customers in the result set, whether or not that customer has also placed any orders. On the other hand, the query is also asking InterBase/Firebird to limit the query to only those customers located in California.

InterBase/Firebird resolves this apparent conflict by always processing the `WHERE` clause before processing any outer joins. The `Customer` table is first limited to those customers in California, and this intermediate result is then joined to the Orders table to which of the California customers have placed orders.

I.6.3 Joining more than two tables

The SQL92 join syntax provides for joins that reference more than two tables. The trick is to establish the join with the first pair of tables, then join this product with the third table, and so on.

For example, the following query finds customers and the order details, where the order included a specific stock item:

```
SELECT C.Name, O.SaleDate, L.Quantity
FROM Customer C JOIN Orders O
ON ( C.CustomerID = O.CustomerID )
JOIN LineItem L
ON ( O.OrderID = L.OrderID )
WHERE L.StockID = '5313';
```


This syntax can be extended to any number of tables. You can even create a circular join. For example, the following statement asks for customers who have ordered products that were made by vendors in the same state as the customer. This query requires a series of joins from `Customer` to `Orders` to `LineItem` to `Stock` to `Vendors`, and another join from the Customer state to the Vendor's state.

```
SELECT DISTINCT C.Name, V.VendorName, C.State_Province
FROM Customer C JOIN Orders O
ON ( C.CustomerID = O.CustomerID )
JOIN LineItem L
ON ( O.OrderID = L.OrderID )
JOIN Stock S
```

```
ON ( L.StockID = S.StockID )
JOIN Vendors V
ON ( S.VendorID = V.VendorID )
AND ( C.State_Province = V.State_Province );
```

Note an important limitation in this `SELECT` statement: tables are added to the `JOIN` expression one at a time. You cannot reference columns from a table until the table has been joined to the expression. For example, the condition linking the `Customer` and `Vendor` tables on their `State` columns cannot be specified until the `Vendor` table has been added to the expression and correctly joined.

I.6.4 Self joins / reflexive joins

A self-join, also known as a reflexive join, is a join in which a table is joined to itself. It compares rows of data within a single table. For example, we could add another column to the `employee` table in the sample `employee` database that would contain the employee's manager number. Since managers are also stored in the `employee` table, we could create a self-join on the employee table to determine the name of each employee's manager.

```
SELECT e1.full_name AS Employee, e2.full_name AS Manager
FROM employee e1 JOIN employee e2
ON e1.mng_id = e2.emp_no;
```

II GLOSSARY

The majority of definitions can be found in the relevant IBExpert subject areas. This glossary includes a number of miscellaneous definitions that could not be allotted to individual IBExpert subjects.

If you are looking for a specific definition in the online documentation, please use the search function. Should you not be able to find the definition you are looking for, please contact documentation@ibexpert.com.

II.1 */Wildcard

The asterisk (*) or so-called wildcard is used, for example, when selecting all or any data (or data sets) meeting a certain condition.

Example:

```
SELECT * FROM EMPLOYEE
WHERE EMPLOYEE.PHONE_EXT='250';
```

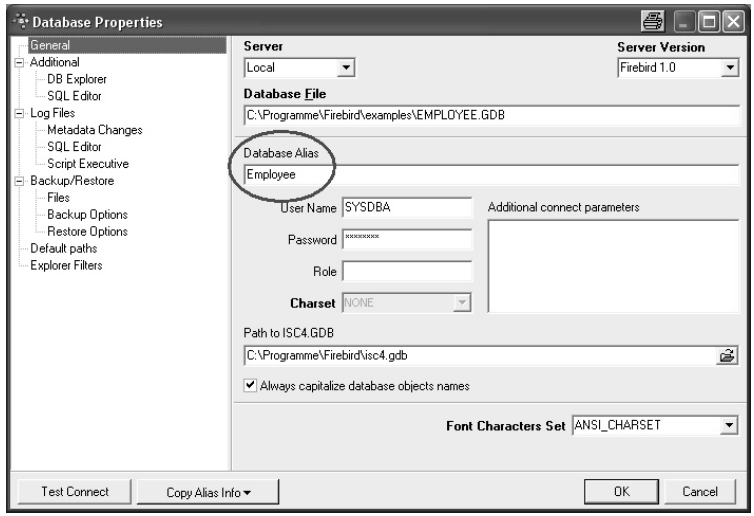
All data sets containing the value 250 in the PHONE_EXT column in the EMPLOYEE table are fetched.

II.2 Alias

An alias is a pseudonym. A database alias is a name chosen by the developer for day-to-day use, as a logical and preferable alternative to the often formally named gdb or fdb file, which is often named in accordance to internal company norms.

The alias indicates the location of the database tables. If the database is stored on a server, the alias also specifies the necessary connection parameters.

It is also used in SQL language to simplify input (saves repeatedly typing the same long database object and field names).



II.3 API (Application Program Interface)

API is the abbreviation for Application Program Interface, which is a set of routines, protocols, and tools for building software applications. A good API makes it easier to develop a program by providing all the building blocks. A programmer puts the blocks together.

Most operating environments, such as MS Windows, provide an API so that programmers can write applications consistent with the operating environment. Although APIs are designed for programmers, they are ultimately of advantage to users because they guarantee that all programs using a common API will have similar interfaces. This makes it easier for users to learn new programs.

Source: http://www.webopedia.com/

II.4 Application

An application is a program or group of programs designed for end users. Software can be divided into two general classes: systems software and applications software. Systems software consists of low-level programs that interact with the computer at a very basic level. This includes operating systems, compilers, and utilities for managing computer resources.

In contrast, applications software (also called end-user programs) includes database programs, word processors, and spreadsheets. Figuratively speaking, applications software sits on top of systems software because it is unable to run without the operating system and system utilities.

An application comprises the executing file, along with any other files, that a program needs to function fully. The word application is often used synonymously with the word program.

Source: http://www.webopedia.com/

II.5 ASCII

ASCII is an acronym for the American Standard Code for Information Interchange. Pronounced *ask-ee*, ASCII is a code for representing English characters as numbers, with each letter assigned a number from 0 to 127. For example, the ASCII code for uppercase M is 77. Most computers use ASCII codes to represent text, which makes it possible to transfer data from one computer to another.

Text files stored in ASCII format are sometimes called ASCII files. Text editors and word processors are usually capable of storing data in ASCII format, although ASCII format is not always the default storage format. Most data files, particularly if they contain numeric data, are not stored in ASCII format. Executable programs are never stored in ASCII format.

The standard ASCII character set uses just 7 bits for each character. There are several larger character sets that use 8 bits, which gives them 128 additional characters. The extra characters are used to represent non-English characters, graphics symbols, and mathematical symbols. Several companies and organizations have proposed extensions for these 128 characters. The DOS operating system uses a superset of ASCII called extended ASCII or high ASCII. A more universal standard is the ISO Latin 1 set of characters, which is used by many operating systems, as well as Web browsers.

Source: http://www.webopedia.com/

II.6 BDE (Borland Database Engine)

BDE is the abbreviation for the Borland Database Engine, the heart of Firebird/InterBase. IBExpert uses this database engine to access and retrieve data. It allows multiple sessions, each one being treated as a "virtual" user.

II.7 Client/Server

The main part of the database intelligence is contained in a server program (e.g. InterBase/Firebird). The operation is sent from the client to the server and is processed there, and the resulting data transferred back to the client.

Client-server architecture is a network architecture in which each computer or process on the network is either a client or a server. Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print servers), or network traffic (network servers).

Clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.

Another type of network architecture is known as a peer-to-peer architecture because each node has equivalent responsibilities. Both client/server and peer-to-peer architectures are widely used, and each has unique advantages and disadvantages.

Client-server architectures are also sometimes called two-tier architectures.

II.8 Comdiag

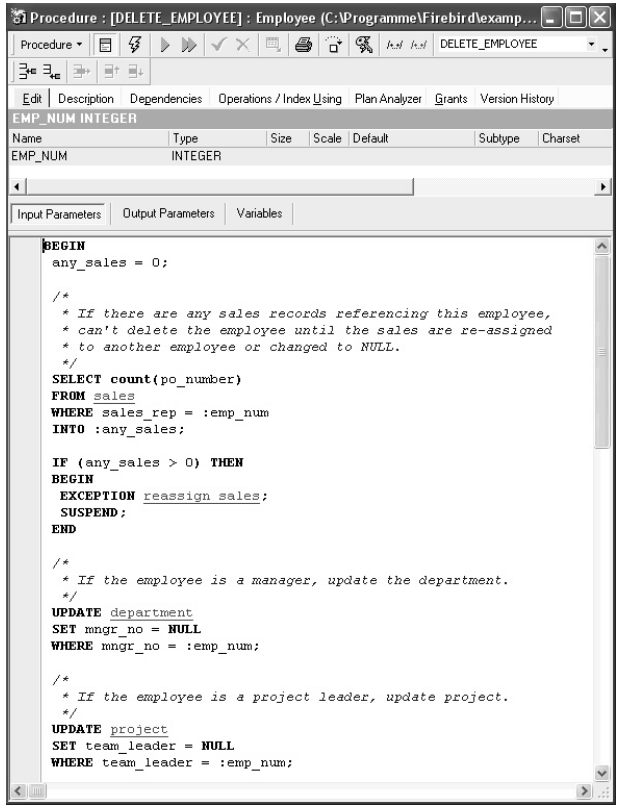
Comdiag is an InterBase/Firebird windows-based program to aid diagnosis of problems that may arise when connecting to InterBase/Firebird servers and the databases managed by those servers.

It validates all InterBase DLLs when connecting the server to the database and checks that the various protocol stacks are correctly installed and loaded.

II.9 Comments

Comments can be incorporated anywhere in an InterBase/Firebird ISQL script, as well as in the procedure body of a stored procedure. The following character sequences are used to determine a comment.

```
/* Comment */
```



Comments can span multiple lines, but a comment cannot be embedded in another comment.

They can also be incorporated in a Firebird script, determined by the following character sequence:

```
-- Comment
```

Comments introduced in this way in Firebird can only cover a single line, i.e. each new line must begin with --. Firebird however also understands the InterBase syntax.

II.10 Compile and Commit / Rollback

A transaction is committed, if all statements in the transactions were performed successfully and the whole transaction was completed without error. By committing a transaction, the instructions entered are interpreted and saved permanently to disk or cancelled. In IBExpert the

icon or [Ctrl + F9] can be used to perform this task. The *Compile* dialog shows whether the modifications, insertions or deletions are correct; the *Commit* button finally writes the alterations permanently to the database.

A transaction is rolled back, if the alterations are cancelled or revoked by the operator, or if an active transaction is perceived by another transaction to be "dead" and so set in a rolled-back condition. Rollback also aborts the compile actions, should errors have been reported or modifications be necessary.

II.11 Conditional Test

Conditional test is an expression that evaluates to logical TRUE or FALSE. If the statement TRUE, the statements in the THEN clause are executed; if FALSE, the statements in the optional ELSE clause are executed. Parentheses around the conditional test are required.

II.12 Constant

In programming, a constant is a value that never changes. The other type of values that programs use is variables, symbols that can represent different values throughout the course of a program.

A constant can be

- a number, such as `25` or `3.6`
- a character, such as `a` or `$`
- a character string, such as `"this is a string"`

Source: http://www.webopedia.com/

II.13 DBMS (Database Management System)

A collection of programs that enables you to store, modify, and extract information from a database. There are many different types of DBMSs, ranging from small sys-

tems that run on personal computers to huge systems that run on mainframes. The following are examples of database applications:

- computerized library systems
- automated teller machines
- flight reservation systems
- computerized parts inventory systems

From a technical standpoint, DBMSs can differ widely. The terms relational, network, flat, and hierarchical all refer to the way a DBMS organizes information internally. The internal organization can affect how quickly and flexibly you can extract information.

Requests for information from a database are made in the form of a query, which is a stylized question. For example, the query

```
SELECT ALL WHERE NAME = "SMITH" AND AGE > 35
```

requests all records in which the `NAME` field is `SMITH` and the `AGE` field is greater than `35`. The set of rules for constructing queries is known as a query language. Different DBMSs support different query languages, although there is a semi-standardized query language called SQL (structured query language). Sophisticated languages for managing database systems are called fourth-generation languages, or 4GLs for short.

The information from a database can be presented in a variety of formats. Most DBMSs include a report writer program that enables you to output data in the form of a report. Many DBMSs also include a graphics component that enables you to output information in the form of graphs and charts.

Source: http://www.webopedia.com/

II.14 DDE (Dynamic Data Exchange)

DDE is an acronym for Dynamic Data Exchange, an interprocess communication (IPC) system built into the Macintosh, Windows, and OS/2 operating systems. DDE enables two running applications to share the same data.

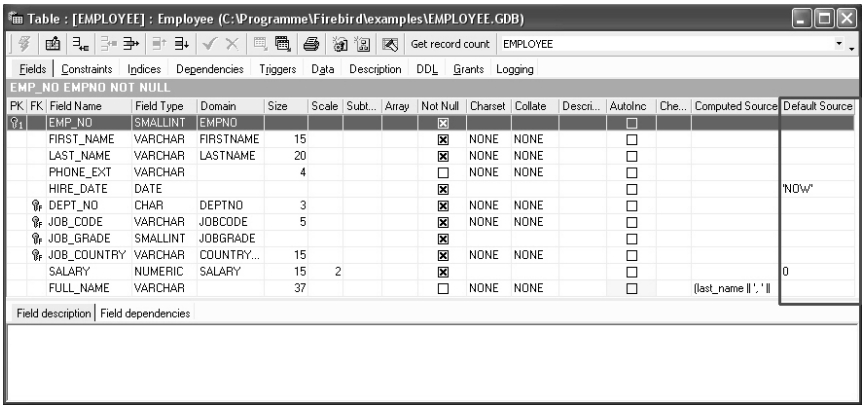
Although the DDE mechanism is still used by many applications, it is being supplanted by OLE, which provides greater control over shared data.

Source: http://www.webopedia.com/

II.15 Default

The `DEFAULT` parameter allows a standard value to be defined, should the user not enter a specific value. A `DEFAULT` value can be defined for a domain or a field. The default value predefined in the domain, can be overridden by the default value entry in the column/field definition following this domain.

In IBExpert it can be specified when creating a new table and fields or when creating a domain.



II.16 DLL (Dynamic Link Library)

DLL is the abbreviation for Dynamic Link Library. DLLs are library files with the suffix DLL. These are executable modules, containing source code or resources, which can access other DLLs or applications. DLLs enable multiple applications, source code and resource to be used collectively in a Windows environment.

II.17 Event

An action or occurrence detected by a program. Events can be user actions, such as clicking a mouse button or pressing a key, or system occurrences, such as running out of memory. Most modern applications, particularly those that run in Macintosh and Windows environments, are said to be event-driven, because they are designed to respond to events.

A database event can be anything relative to the rows in a table or values in fields. Coordinated and monitored by the Firebird/InterBase Event Manager.

II.18 Expression

An expression is a group of symbols that represent a value.

In programming, an expression is any legal combination of symbols that represents a value. Each programming language and application has its own rules for what is legal and illegal. For example, in the C language `x+5` is an expression, as is the character string `"MONKEYS"`.

Every expression consists of at least one operand and can have one or more operators. Operands are values, whereas operators are symbols that represent particular actions. In the expression

```
x + 5
```

`x` and `5` are operands, and `+` is an operator.

Expressions are used in programming languages, database systems, and spreadsheet applications. For example, in database systems, you use expressions to specify which information you want to see. These types of expressions are called queries.

Expressions are often classified by the type of value that they represent. For example:

- Boolean expressions : Evaluate to either `TRUE` or `FALSE`
- Integer expressions: Evaluate to whole numbers, like `3` or `100`
- Floating-point expressions: Evaluate to real numbers, like `3.141` or `-0.005`
- String expressions: Evaluate to character strings

Source: http://www.webopedia.com/

II.19 FBK Files

`FBK` is the standard suffix used for Firebird backup database file names.

This is not compulsory, in fact a Firebird or InterBase backup database may be named with any suffix. This standardization does however provide a certain conformity, of particular importance if a database is to be administrated long term by numerous people.

II.20 FDB Files

`FDB` is the standard suffix used for Firebird database file names. It is derived from the InterBase standard, `.GDB`.

This is not compulsory, in fact an Firebird or InterBase database may be named with any suffix. This standardization does however provide a certain conformity, of particular importance if a database is to be administrated long term by numerous people.

II.21 FTP (File Transfer Protocol)

FTP is an abbreviation of File Transfer Protocol, the protocol for exchanging files over the Internet. FTP works in the same way as HTTP for transferring web pages from a server to a user's browser and SMTP for transferring electronic mail across the internet in that, like these technologies, FTP uses the internet's TCP/IP protocols to enable data transfer.

FTP is most commonly used to download a file from a server using the internet or to upload a file to a server (e.g., uploading a web page file to a server).

Source: www.webopedia.com

II.22 GBK Files

`GBK` is the standard suffix used for Borland InterBase backup database file names.

This is not compulsory, in fact an InterBase or Firebird backup database may be named with any suffix. This standardization does however provide a certain conformity, of particular importance if a database is to be administrated long term by numerous people.

II.23 GDB Files

GDB is the standard suffix used for Borland InterBase database file names. It originates back to the days when the Interbase Corporation was still called Groton Database Systems.

This is not compulsory, in fact an InterBase or Firebird database may be named with any suffix. This standardization does however provide a certain conformity, of particular importance if a database is to be administrated long term by numerous people.

II.24 GRC Files

.GRC files are IBExpert Database Designer files.

II.25 HTML (HyperText Markup Language)

Short for HyperText Markup Language, the authoring language used to create documents on the World Wide Web. HTML is similar to SGML (Standard Generalized Markup Language), although it is not a strict subset.

HTML defines the structure and layout of a web document by using a variety of tags and attributes. The correct structure for an HTML document starts with <HTML><HEAD>(enter here what document is about), <BODY> and ends with </BODY></HTML>. All the information you'd like to include in your web page fits in between the <BODY> and </BODY> tags.

There are hundreds of other tags used to format and layout the information in a web page. Tags are also used to specify hypertext links. These allow web developers to direct users to other web pages with only a click of the mouse on either an image or word(s).

Source: http://www.webopedia.com/

II.26 HTTP (HyperText Transfer Protocol)

Short for HyperText Transfer Protocol, the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the web server directing it to fetch and transmit the requested web page.

The other main standard that controls how the World Wide Web works is HTML, which covers how web pages are formatted and displayed.

HTTP is called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it. This is the main reason that it is difficult to implement web sites that react intelligently to user input. This shortcoming of HTTP is being addressed in a number of new technologies, including ActiveX, Java, JavaScript and cookies.

Source: http://www.webopedia.com/

II.27 IDE (Integrated Development Environment)

Abbreviated as IDE, a programming environment integrated into a software application that provides a GUI builder, a text or code editor, a compiler and/or interpreter and a debugger. Visual Studio, Delphi, JBuilder, FrontPage and DreamWeaver are all examples of IDEs.

II.28 OAT (Oldest Active Transaction)

The Oldest Active Transaction (OAT) is the earliest transaction in the database, recorded by the versioning engine in the TIP (Transaction Inventory Page) that is currently active or open.

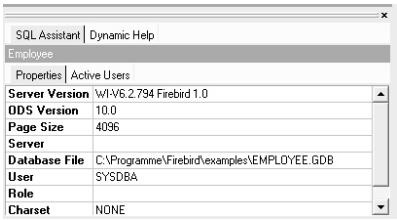
II.29 ODBC (Open DataBase Connectivity)

ODBC (pronounced as separate letters) is short for Open DataBase Connectivity, a standard database access method developed by the SQL Access group in 1992. The goal of ODBC is to make it possible to access any data from any application, regardless of which database management system (DBMS) is handling the data. ODBC manages this by inserting a middle layer, called a database driver, between an application and the DBMS. The purpose of this layer is to translate the application's data queries into commands that the DBMS understands. For this to work, both the application and the DBMS must be ODBC-compliant -- that is, the application must be capable of issuing ODBC commands and the DBMS must be capable of responding to them. Since version 2.0, the standard supports SAG SQL.

Source: http://www.webopedia.com/

II.30 ODS Version

ODS = On-Disk Structure.



The ODS version shows with which database version the database was created, e.g. InterBase 5 = 9, InterBase 6 = 10.0, InterBase 6.5 = 10.1, InterBase 7 = 11.

For more information about the InterBase On-Disk Structure, please refer to Ann Harrison's article, Space Management in InterBase.

II.31 OIT (Oldest Interesting Transaction)

The Oldest Interesting Transaction (OIT) is the earliest transaction in the database, recorded by the versioning engine in the TIP (Transaction Inventory Page) with a status other than committed. Every transaction prior to that one represents an unbroken chain of insertions and updates into the database.

II.32 OLAP (Online Analytical Processing)

Short for Online Analytical Processing, a category of software tools that provides analysis of data stored in a database. OLAP tools enable users to analyze different dimensions of multidimensional data. For example, it provides time series and trend analysis views. OLAP often is used in data mining.

The chief component of OLAP is the OLAP server, which sits between a client and a database management system (DBMS). The OLAP server understands how data is organized in the database and has special functions for analyzing the data. There are OLAP servers available for nearly all the major database systems.

Source: http://www.webopedia.com/

II.33 OLE (Object Linking and Embedding)

OLE is an abbreviation of Object Linking and Embedding, pronounced as separate letters or as *oh-leh*. OLE is a compound document standard developed by the Microsoft Corporation. It enables you to create objects with one application and then link or embed them in a second application. Embedded objects retain their original format and links to the application that created them.

Support for OLE is built into the Windows and Macintosh operating systems. A competing compound document standard developed jointly by IBM, Apple Computer, and other computer firms is called OpenDoc.

Source: http://www.webopedia.com/

II.34 Operand

In all computer languages, expressions consist of two types of components: operands and operators. Operands are the objects that are manipulated and operators are the symbols that represent specific actions. For example, in the expression

```
5 + x
```

x and 5 are operands and + is an operator. All expressions have at least one operand.

Source: http://www.webopedia.com/

II.35 Operator

An operator is a symbol that represents a specific action. For example, a plus sign (+) is an operator that represents addition. The basic mathematic operators are + addition, - subtraction, * multiplication, / division.

In addition to these operators, many programs and programming languages recognize other operators that allow you to manipulate numbers and text in more sophisticated ways. For example, Boolean operators enable you to test the truth or falsity of conditions, and relational operators let you compare one value to another. For example, the expression

```
x < 5
```

means x is less than 5. This expression will have a value of TRUE if the variable x is less than 5; otherwise the value of the expression will be FALSE.

Relational operators are sometimes called comparison operators. Expressions that contain relational operators are called relational expressions.

Source: http://www.webopedia.com/

II.36 PIP (Page Inventory Page)

The Page Inventory Page (PIP) is one of the ten page types defined in InterBase/Firebird. The PIP is used along with the pointer page for space management.

Every page in the database is represented by one bit in the PIP, this bit indicating whether the page is currently in use. PIPs occur at fixed intervals in the database, the interval being determined by the database page size. PIPs are never released.

For those interested in more detailed information, Ann Harrison's article, Space Management in InterBase, provides an in-depth insight into page types and their roles.

II.37 RDBMS (Relational Database Management System)

RDBMS is the abbreviation for Relational Database Management System and is pronounced as separate letters, a type of database management system (DBMS) that stores data in the form of related tables. Relational databases are powerful because they require few assumptions about how data is related or how it will be extracted from the database. As a result, the same database can be viewed in many different ways.

An important feature of relational systems is that a single database can be spread across several tables. This differs from flat-file databases, in which each database is self-contained in a single table. Almost all full-scale database systems are RDBMS's. Small database systems however, use other designs that provide less flexibility in posing queries.

From a technical standpoint, DBMSs can differ widely. In addition to the relational DBMS, there are also network, flat, and hierarchical DBMS's. These all refer to the way a DBMS organizes information internally. The internal organization can affect how quickly and flexibly you can extract information.

Source: http://www.webopedia.com/

II.38 Statement

A statement is the smallest unit of a program. Statements are separated in InterBase/Firebird by a semicolon.

A statement is an instruction written in a high-level language. A statement directs the computer to perform a specified action. A single statement in a high-level language can represent several machine-language instructions. Programs consist of statements and expressions.

Source: http://www.webopedia.com/

II.39 String

A string is a series of characters manipulated as a group. A character string differs from a name in that it does not represent anything -- a name stands for some other object.

A character string is often specified by enclosing the characters in single or double quotes. For example, `WASHINGTON` would be a name, but `'WASHINGTON'` and `"WASHINGTON"` would be character strings.

Source: http://www.webopedia.com/

II.40 TID (Transaction ID)

Each user performs transactions, and each transaction is given its own ID. The TIDs (Transaction IDs) are numbered sequentially, i.e. transaction ID 10 was started before the transaction with the ID 11.

The TIPs contain all transactional information in an array of bits, two per transaction, which indicate the state of the transaction. The transaction ID is an index into this array.

When the transaction number is allocated to a transaction, the user also receives a copy of the TIP (Transaction Inventory Page), which comprises the status of all transactions. If a data set is inserted or modified, the TID is entered next to the alteration. These simple rules are all that is needed to implement the InterBase/Firebird versioning.

A transaction can only see those transactions with a lower TID than its own. Furthermore, all other transactions that were still active at that point in time when the transaction was started, are invisible to the transaction.

The TIP copy, provided when the TID number is allocated, can be used to monitor the status of all other transactions at the point in time when the transaction was started. The only way to obtain a newer, more up-to-date TIP is to request a new TID.

For example: User A has a TID 10, user B has a TID 11 or higher. He could also have a TID 9 or lower, when his transaction was still active at the point in time when user A began his transaction with the TID 10. Otherwise he would not be able to alter the data set X. User B modifies the data set with his active transaction.

Now user A modifies data set X. When the transaction is posted, User A receives a deadlock error or an update conflict, providing the Transaction Isolation Level is set at *repeatable read*. this message informs user A that his modification cannot be carried out, as another user - in this case user B - has modified the data set. The programmer can decide at this point, how the program reacts to this situation.

II.41 TIP (Transaction Inventory Page)

The Transaction Inventory Page (TIP) is one of the ten page types defined in InterBase/Firebird.

Each and every user transaction is consecutively numbered, using the InterBase/Firebird Transactions Inventory Page (TIP) (also known as the Transaction Information Page). These transaction numbers are used by the InterBase/Firebird versioning engine to ensure that users always receive a consistent view of the database. It shows the status of each and every transaction in the database, and adheres to two main rules:

- Only those transactions are visible, whose ID <= own ID.
- Only those transactions are visible, which were already committed at the time the own transaction was started.

Transactions are shown with one of the following four status values:

Table: Values in the Transaction Information Pages

Status Code	*Description*
A	Transaction is active, or in process
C	Transaction was committed. The changes made by this transaction can be applied if necessary to show a consistent view of the database.
R	Transaction was rolled back. The changes made by this transaction should be ignored.
L	Limbo transaction. This transaction was part of an operation involving more than one database within an embedded SQL application.

For example, 1C = first transaction committed, 2A = second transaction is active, 3C = third transaction is rolled back, 4L = Transaction is in limbo (i.e. when a transaction is dependent upon another transaction in another database = two-phase commit). This information is important for the garbage collection.

The TIPs contain this information in an array of bits, two per transaction, that indicate the state of the transaction. The transaction ID (TID) is an index into this array.

Special transactions IDs

InterBase/Firebird tracks three special positions within the transaction history:

- The Oldest Interesting Transaction (OIT) is the earliest transaction in the database with a status other than committed. Every transaction prior to that one represents an unbroken chain of insertions and updates into the database.
- The Oldest Active Transaction (OAT) is the earliest transaction in the database that is currently active or open.
- The Next Transaction Number is the ID that is used for the next transaction that starts.
  You can find these numbers in the IBExpert Database Statistics display within Server Manager, or using the `gstat -h` command in isql.

When you start a transaction, InterBase/Firebird makes a copy of the TIP into the server memory cache assigned to your process, starting from the page holding the OIT and finishing with the page holding the OAT.

Whenever the database is backed up and restored, the transaction inventory is wiped out and the next transaction number is set to 1.

There is also a mechanism in the InterBase/Firebird server TIP page, to allow a local TIP page for each user. The local TIP page is generated the minute a new user presses the Execute [F9] key. Please refer to TID (Transaction ID) for further information.

The advantage of such a system is that older records are held ready. The disadvantage for users, who execute, but need a considerable time before finally committing is that the local TIP becomes very large, as it always begins at the oldest active transaction; so that it is possible using this technique, for one transaction to hold everything up and slow the transaction processing for everyone. If a system becomes increasingly slow with time, it is almost always due to the fact that TIP pages are being filled further and further with transaction information, because the first transaction has not been committed. 99% of local TIPs are held in the RAM, until there are no further pages free.

- *Note:* If you are only doing a `SELECT` in your transaction, you should always `COMMIT` to avoid creating an "interesting" transaction (transaction with a status code other than committed in the TIP).

All TIPs are of the page size defined when creating the database. 16,000 transactions fit, for example, onto a 4K page.

TIPs and Server Crashes:

If a server crashes or hangs during user transactions, the InterBase/Firebird server simply looks at the TIP, and rolls back all operations that were still active. This means that an InterBase/Firebird server can be rapidly restarted. As soon as the operating system is up and running, InterBase/Firebird is also up and running. Forced writes however influence the sequence in which is written:

- IBExpert Database Properties / Forced Writes - when committing InterBase/Firebird saves all data sets to the hard drive and then to the TIP.
- Without forced writes the process is minimally quicker, but on a Windows platform, Windows decides what should be saved to file, where and when; and the data pages are saved to file last i.e. the TIP changes are written first and then the data sets, which could possibly lead to inconsistencies.

Therefore forced writes are extremely important when working on a Windows platform. Without forced writes, the computer needs to be extremely secure.

II.42 Transaction

A transaction is a single task with a number of specific characteristics. An application can perform one or more operations, within the context of a transaction, each of which must be completed in sequence.

One of the main tools used by relational databases to maintain data integrity is the transaction. A transaction is a task with a number of specific characteristics:

- An application can perform one or more operations within the context of a transaction, each of which must be completed in sequence. An operation consists of, as a rule, one SQL statement, such as `SELECT`, `INSERT`, `UPDATE`, or `DELETE`.
- The changes performed by the transaction can be committed if all of the operations in the transaction are completed. Until the results of the transaction are committed, the changes made to the database are invisible to other users.
- A transaction can also be rolled back. In this case, as far as other database users are concerned, the data never changed.

Because of these characteristics, transactions ensure that complex operations on the database are performed completely. Transactions provide complete protection against operations not being completely processed, therefore ensuring data integrity.

A transaction can be in one of the following four states:

- in limbo
- Committed (please refer to Commit / Rollback)
- Rolled back (please refer to Commit / Rollback)
- Active

II.42.1 Transaction Number Column

For every table you create, including system tables, InterBase/Firebird maintains an extra column for the transaction number. When you insert or update a column as part of a transaction, the transaction number is written to this column, so that InterBase/Firebird knows which transaction is controlling that row of the table. Even when you delete a row as part of the transaction, the number is written to the row until the transaction is committed or rolled back, in case there is a problem, or in case the transaction is a lengthy one.

The InterBase/Firebird versioning engine uses this transaction number to ensure that each user receives a consistent view of the database at a moment in time. This is known as a
repeatable read. Please refer to versioning engine for further information.

II.42.2 Active Transactions

A transaction is active, if one of the following conditions is true:

- The transaction has not yet started.

- The transaction has started but not yet completed.
- The transaction has started, could not however complete successfully, due to for example, a system crash or communication problems etc.

The actual status of each transaction is recorded in the TIP (Transaction Inventory Page). In fact, the only alteration that occurs when a transaction is committed is the alteration to the status in the TIP from active to committed.

II.42.3 Transactions in Limbo

InterBase/Firebird's transaction mechanism, like most databases, can only handle transactions within a single database. However within an embedded SQL application, InterBase/Firebird can perform operations on more than one database at a time.

With a logical transaction that spans databases, InterBase/Firebird handles the operations within each database as separate transactions, and sequences them using a two-phase commit model, to ensure that both transactions complete or that neither completes. When InterBase/Firebird is ready to commit or rollback such a multidatabase transaction, it first changes the transaction status from active to limbo. It then performs the commit or rollback operation. Finally the transaction status is changed from limbo to committed.

Transactions in limbo are transactions that have been started by the PREPARE command within the framework of a two-phase commit. The transaction may or may not still be running. This transaction may become relevant at any point in time and all changes made so far may be committed or rolled back. Such alterations made by such transactions can neither be examined or ignored; they can neither be defined as executed or aborted. They can therefore not simply be removed from the database.

However for a database backup to be fully performed without aborting, such transactions in limbo need to be ignored in the backup. Only those most recent, committed transactions are backed up. It allows a database to be backed up, before recovering corrupted transactions. Generally in limbo transactions should be recovered before a backup is performed.

- *Note*: BDE clients use only single-database transactions, even if the client application accesses two or more databases. Embedded SQL and InterBase/Firebird API provide methods for programming distributed transactions.

II.43 Two–Phase Commit

A transaction spanning multiple InterBase/Firebird databases is automatically committed in two phases. A two-phase commit guarantees that the transaction updates either all of the databases involved or none of them - data is never partially updated.

In the first phase of a two-phase commit, InterBase/Firebird prepares each database for the commit by writing the changes from each subtransaction to the database. This subtransaction is the part of a multi-database transaction that involves only one database. In the second phase, InterBase marks each subtransaction as committed in the order that it was prepared.

If a two-phase commit fails during the second phase, some subtransactions are committed and others are not. A two-phase commit can fail if a network interruption or disk crash makes one or more databases unavailable. Failure of a two-phase commit causes in limbo transactions, i.e. transactions that the server does not know whether to commit or roll back.

It is possible that some records in a database are inaccessible due to their association with a transaction that is in a limbo state.

Note: The Borland Database Engine (BDE), as of version 4.5, does not exercise the two-phase commit or distributed transactions capabilities of InterBase/Firebird, therefore applications using the BDE never create limbo transactions.

II.44 Variable

A symbol or name that stands for a value. For example, in the expression

x+y

x and y are variables. Variables can represent numeric values, characters, character strings, or memory addresses.

Variables play an important role in computer programming because they enable programmers to write flexible programs. Rather than entering data directly into a program, a programmer can use variables to represent the data. Then, when the program is executed, the variables are replaced with real data. This makes it possible for the same program to process different sets of data.

Every variable has a name, called the variable name, and a data type. A variable's data type indicates what sort of value the variable represents, such as whether it is an integer, a floating-point number, or a character.

The opposite of a variable is a constant. Constants are values that never change. Because of their inflexibility, constants are used less often than variables in programming.

Source: http://www.webopedia.com/

III FAQs

Here we have attempted to list some of the more frequently asked questions regarding IBExpert. Should you not be able to find a solution to your problem under the links provided here or elsewhere within the IBExpert documentation, please contact one of our newsgroups:

Username **ibexpert**
Password **ibexpert**

news://ibexpert.info/interbase.ibexpert.de German language
news://ibexpert.info/interbase.ibexpert.en English language
news://ibexpert.info/interbase.ibexpert.ru Russian language
news://ibexpert.info/interbase.ibexpert.fr French language

or send an email to documentation@ibexpert.com or support@ibexpert.com
or use our bug track system in IBExpert

III.1 How do I connect to a database?

See Connect to Existing Database and Register Database.

If you are experiencing problems with a remote connection, please refer to Communications Diagnostics.

III.2 Why do I need to register a database?

See Register Database.

III.3 How do I create a new database?

See Create Database.

III.4 How do I use the SQL Editor?

See SQL Editor.

III.5 What is the Performance Analysis for?

See Performance Analysis.

III.6 What is the Query Plan?

See Plan Analyzer.

III.7 How can I optimize an SQL Statement?

See Optimizing an SQL Statement.

III.8 How do I debug a stored procedure?

See Debug Procedure.

III.9 Are there typical windows for all Object Editors?

See Database Objects.

III.10 How can I use the view and procedure version control?

See New View / Version History.

III.11 What is the Project View in the DB Explorer for?

See Project View.

III.12 What is the Recent list in the DB Explorer for?

See Recent List.

III.13 How do I use the integrated Report Manager?

See Report Manager.

III.14 Why can I not see the index statistics in the Table Editor?

Use the right-click menu directly on the Indices page in the Table Editor and select the menu item Show Statistics.

III.15 Why does the index selectivity/statistics not change?

See Recompute Selectivity of all Indices.

III.16 Indices do not seem to work on my newly installed application

See Recompute Selectivity of all Indices.

III.17 How can I integrate the online Help files into IBExpert?

Please refer to IBExpert Help menu.

III.18 Import CSV Files

Here are a few questions that have arisen with regard to importing CSV files.

1. In the examples a DB field gets the correct value if the imported data is numeric. Does truncation occur if it is not an integer?

INSERTEX itself doesn't truncate numeric values. Of course, if you're inserting numeric value into Integer fields the server will truncate it.

2. Can I import dates and if so what ASCII format does it accept for DATE or TIME-STAMP columns or do I need to perform my own external conversion of dates & times to a 32 bit integer?

You can import dates and INSERTEX accepts any date format known by the server. For example, 1.08.2004 or 1-AUG-2004.

3. If the imported string is longer than I specify for VARCHAR or CHAR does truncation occur?

Yes, it does.

IV Database technology–related articles

IV.1 Enterprise–wide data model

New technologies are not a universal remedy:
ways to achieve an enterprise-wide data model
Today almost all enterprises are fighting against a profusion of data, simultaneously suffering from a lack of useful information.

Applications have grown isolated and exist in their own more or less well-documented data and file world.

An important task of information management is to convert the multitude of data into a manageable amount of significant information.

"Information as a resource" has integrated itself in the series of terms that have become common knowledge for data users. This keyword is commonly used and everyone now considers information to be of equal importance to the classical production factors capital, human resources and plant.

Information management is an old hat which has finally been recognized and allocated its own organizational unit.

The persons appointed the responsibility for this information management are those who have so far been responsible for information systems: the DP or Organizational Manager.

As an additional admonition, these managers are then required by general management to also consider old data as a new resource, and treat it with the corresponding diligence.

This viewpoint may be exaggerated, however the impression is given in many enterprises that by appointing an Information Manager, enough has been done to keep up with the new trend, and it is now possible to return to day-to-day business with responsibilities for:

- Hardware and software selection and implementation,
- Design of a hardware and software architecture for centralized and decentralized applications,
- Provision of the infrastructure for information users in the various enterprise sectors,
- Maintenance of standards and procedures.

But is that really all that information management needs to do? It is indisputable that the strategic direction of Information Technology is a considerable complex task of information management, the tasks mentioned above having become considerably more complex than they ever were.

Information management has lost its way in the data-processing jungle. The technical range, with its overabundance of possibilities, has not just become more extensive and complex, but has also brought with it compatibility and integration problems due to the

lack of standardization; just consider the range of different network types, communication technologies, CIM products.

It's no wonder that information management can these days easily err in the data-processing jungle. But let's assume that the IT-technical world was different: strategically concise, tidier, clearly structured and without any technical problems.

What would then stop the enterprise from finally being able to fully utilize the longed-for possibilities to exchange all information as desired?

Everyone could then:

- within the realms of his authentication, independently
- use and alter others' information, create new information and make it available to others?

What is stopping them? This picture might be enticing, but unfortunately extremely deceptive.

Because even the most perfect technology cannot hide the fact that, although bits and bytes can be distributed as wished, their information content could still continue to be unknown, or at least be misinterpretable.

By now it should be clear, that today's information management insufficiently fulfils the fundamental tasks of tomorrow:

- Information planning and information strategy
- Design of an application architecture
- Planning software applications

These three fields of responsibility are closely linked together, as an expedient planning strategy of individual software applications needs to be based on a previously compiled applications architecture, designed for the future.

The application architecture itself will need to be based on the results of the information plan and strategy, so that this task can be regarded as, in the long-term, the central logical basis.

The following remarks will therefore be confined to this basic function. There are two aspects to information planning. It demands firstly that you deal with the information itself - specifically and in detail. And it needs the managerial functions that create and process the information. However the lynchpin remains the information itself.

We are still confused – but on a global level

So, initially the information is in the foreground. Information cannot be classified as such, until the data has been complemented by its semantic content, i.e. its meaning, thus becoming interpretable.

However the current situation in most enterprises still predominantly mirrors the conventional picture of data processing and not that of targeted information processing.

Applications systems that have grown isolated exist in their own world, where no one system is aware of the other, and which, at best, are only able to communicate via elaborate interfaces.

Data communication demands a common data appreciation though. However homonyms (terms with the same name but a different meaning) and synonyms (terms with different names but the same meaning) have become the order of the day in both application systems as well as in individual departments.

Applications, whose job it is to compile summaries and analyses, composed from base data from different operative systems, for planning purposes, or even as a tool to support enterprise decision making, find it extremely difficult to deliver reliable results.

Reliability can only be achieved, when it can be assured that the base data do not just have the same name, but also the same meaning.

As clear definitions and descriptions for the data meaning are still missing in many enterprises, it is right to doubt the informational value of many an analysis or report. This situation cannot however be improved by implementation of new technology, which serves no other purpose than to distribute the dubious data more quickly.

New technologies alone may even make this problem worse, by ingeniously helping to expand localized chaotic situations into global ones, based on the principle, "We are still confused, but on a global level".

Structuring data comprehensively and usefully

One of the most important tasks of information management is therefore to transform the multitude of existing data into a manageable quantity of meaningful information, in a structure that is both comprehensible and therefore usable for all information users.

This structure is the well-known data model. A data model is an illustration of the enterprise's information (or parts thereof) and their interrelations from a purely managerial point of view, independent of how they might be realized in the data-processing world.

These days the importance of such an enterprise-wide data model is almost indisputable and its design and maintenance should be a task for data management, which is an integral constituent of information management.

Unfortunately in reality, surprisingly few enterprises dare to venture the construction of such a model. One the reasons for this appears to be fear of the word "enterprise-wide", as it gives the impression of an impossibly huge and insurmountable task.

But there are in fact realistic and viable ways by which "enterprise-wide" can be approached step by step, without having the rug pulled out from under your feet. One of these methods leads to what should here be called "enterprise-wide data model", the other leads to the resulting "enterprise data model".

The construction of both models is based on the same theoretically established and empirically tested method, that of the data model, which however will not be gone into detail here.

Both models differ in their aim and, more than anything else, in their level of detail. Both models should enable information planning and information utilization globally across all projects, nevertheless each with a somewhat different specificity.

The enterprise-wide data model

The enterprise-wide data model corresponds to today's current established data model, and has the certainly extremely ambitious aim to achieve the following:

- A complete base of all information that the enterprise has to offer (including a professional data catalog), which is able to serve both as a detailed fundament of information and communication between departments, and aid with data processing.
- To provide a specification from which database structures can then be derived.
- To keep project interfaces small.

How is it possible to meet these high demands? Such a detailed data model cannot realistically be achieved in one simple step, but needs to be constructed from many small sub- data models.

Each single partial model results from a project, which applies methodical data analysis. Each project creates a project-related data model, confined to its own informational area. The terms and concepts used in this data model however need to be clearly defined and be valid for the total enterprise.

The enterprise-wide data model evolves from the bottom up, arising from the union of the single project results into one consolidated structure.

Practice shows that this method has the following advantages:

- Each project recognizes the benefits of data analysis itself as an aid.
- The resulting "project data model" can be utilized immediately.
- The project result has been achieved to the great level of detail required, yet with a manageable amount of effort.

Problems arise however with this method when consolidating the partial models. It often becomes apparent at the interface of two projects, that the supposed enterprise-wide denomination and definition of the data is only actually fully valid from the limited project viewpoint, and now needs to be synchronized with the other projects. Information streaming increases project effectiveness.

This fine-tuning can be an elaborate process, which also in addition needs to take into account the human factor, namely the danger of those involved mistaking their own contributions and efforts as their property.

The process is also elaborate, because alteration to names and structures could have an effect on the results of other projects (e.g. functional flow descriptions), and other projects may need to adapt their results accordingly.

It is only possible to minimize this project-related annoyance if:

- Each sub-project is adequately informed of the enterprise's strategy with regard to mutual information, and feels sufficiently obliged to comply.

- Each project is kept informed right from the beginning of the results of previous or progress of current projects, and is able to use these actively, thereby saving expenditure, and even more importantly, effort.

This method produces immediate results, as even the initial results of the first project are a step towards information organization, without which information management is powerless in the long-term.

However the enterprise-wide data model cannot be used as a basis for information planning until at least two years later, as it takes this long for the results of the individual projects to be delivered, quality-controlled and synchronized with each other.

The enterprise data model however demonstrates its benefits rapidly, because it is constructed as an independent assignment, detached from other projects and with a different target: that of the enterprise data model.

The enterprise data model primarily follows a different target direction to the enterprise-wide data model. It does not aim to achieve a detailed data catalog and will never represent a complete base of all data in the enterprise.

In contrast it should:

- Provide an summary of the enterprise's information at top level,
- Recognize information supply areas ("theme" databases), according to which this information can grouped and summarized,
- Be a decision aid, enabling projects to be defined more precisely at their initiation, and last but not least,
- Produce a result with which enterprise management can demonstrate to all information users that they take the term "information" seriously.

These goals of a collective consolidated structural summary of the enterprise cannot be attained by joining the individual project results, bottom-up, and then integrating them upwards. An enterprise data model can only be developed as an independent and self-contained project, with participation of all management levels, as summary and not detail information is required here.

By management we mean the specialist departments and sectors, as it is only here that data can be defined from a managerial point-of-view.

Data collation is achieved through interviews relevant for the level concerned and related professionally and technically.

Its goes without saying that the definition and description of this data will have a different quality to that of the enterprise-wide data model. In the enterprise data model relationships are identified clearly and precisely, always with the aim of simplification and rough abstraction. (Keep in mind that the objective here should not be a constant information refinement down to the last detail (i.e. an endless top-down process), but the construction of an approximate summary model that can continue to be maintained at this crude level.)

Project model with clear task definition

As the enterprise model is a self-contained investment, without direct implementation into a data application, its initial construction should be completed within a maximum 6 month time frame. The model can then be used immediately at project initiation.

The enterprise data model can be made immediately available and passed on to projects, along with a clear definition of which of the subject areas described belong to the project objectives.

Project boundaries can be defined and established in relationship to each other right from the start, and subproject intersections can at least be fixed at a rough level.

This anticipatory description of intersections considerably reduces later investment for project adjustments, as each project knows its own "data limits", and is also aware from the start which other information management project partners he will need to confer with for the fine-tuning phase.

The results of this project refinement are not included in the enterprise data model, but grow together to form their own independent enterprise-wide data model.

The enterprise-wide data model and enterprise data model are therefore not "either-or" models, but are, in the true sense of the word, "as-well-as" complements. But what do both these models have to do with future-oriented information planning and information strategy, are they not managed by data administration?

The problem today lies in the term administration, which has little to do with management and consequently with strategy and planning. Many enterprises have started constructing an enterprise-wide data model, but its use is still mainly limited to the unification of terms and information correlations. Compared to the alternative of prevailing data chaos this limitation is however still a huge step forward, which itself is worth a certain amount of effort.

However data and information planning require more, and open up in the long-term other perspectives. Information planning should achieve the goal of comparing the enterprise's current information supply situation with future visions and to assess them, in order to recognize supply deficits and to be able to simulate and optimize future supply situations. This however assumes that not only the enterprise's information is known, but also the functions that are connected to the use of this information. Principally only the comparison of a data model with the functions or functional areas that are necessary for the improvement or extensions of strategically important business areas, allows informational gaps and superfluous informational ballast to be detected. But what does this mean, in view of both the above-mentioned types of data model?

The enterprise-wide data model is only suitable for detail planning aspects, because of its level of detail; that is, when dealing with the concrete definition for contained single subject areas.

In contrast, this model is unsuitable for planning or strategic aspects. And yet it is the only model, in many cases, that is (at least in part) used by information management.

Foundation for theme databases

But who begins with strategic management tasks in other enterprise areas at the operative level, such as the Internal Revenue? Information management is often degraded to the level of dealing with nothing other than the daily business (enterprise-wide data model), instead of setting the basis for management tasks: the enterprise data model.

The enterprise data model offers management (and not just information management), for the first time and within a short period of time, a defined basis for their own information and informational areas (theme databases) which is comprehensible to all.

This model allows a meaningful and clear classification of information for the enterprise's functional areas and organizational units. It supports the construction of a more comprehensive model, with which a conscious design and simulation of future information management is made possible.

And there is one more advantage: the enterprise data model includes business management, in its target-oriented way of thinking, in information and informational correlations right from the beginning. Data model and information management become accessible to all concerned, becoming today's obligation to go on the "search for tomorrow's information".

$

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

How this book was produced

Foreword

This book has been created fully automatically using the Content Management System, PHPtree, developed by M² IT Design & Consulting. The editorial system for the input and editing of texts and illustrations, as well as all necessary data required technically for the creation of the book, are stored and processed by PHPtree in a Firebird database. The resulting PDF file can be processed and printed by any print shop.

PHPtree is an ideal Content Management System and editorial system for the creation and administration of web sites, documentation and other digital media. It can publish data from a wide range of databases (all SQL databases, AS/400, iSeries, ODBC, XML interfaces) or process such data in digital workflows. Individual design, customized functionalities and the ultimate medium to be used can all be specified and realized using templates.

The integrated parsing engine generates html, xml, wml and pdf documents, to name just a few, enabling PHPtree to be used as a genuine cross-media tool. For example, both a web site and a product catalog can be created from a single data source.

PHPtree

PHPtree is a database explorer, which can be accessed and operated from all popular web browsers. It organizes and displays data content clearly in a visual tree structure, and is simple to navigate and use. All the usual functions, such as copy, paste, duplicate, import, export etc., can be used on all objects.

PHPtree displays the content and the objects of the database dynamically. The hierarchical data structure can be simply and quickly altered, with the appropriate rights.

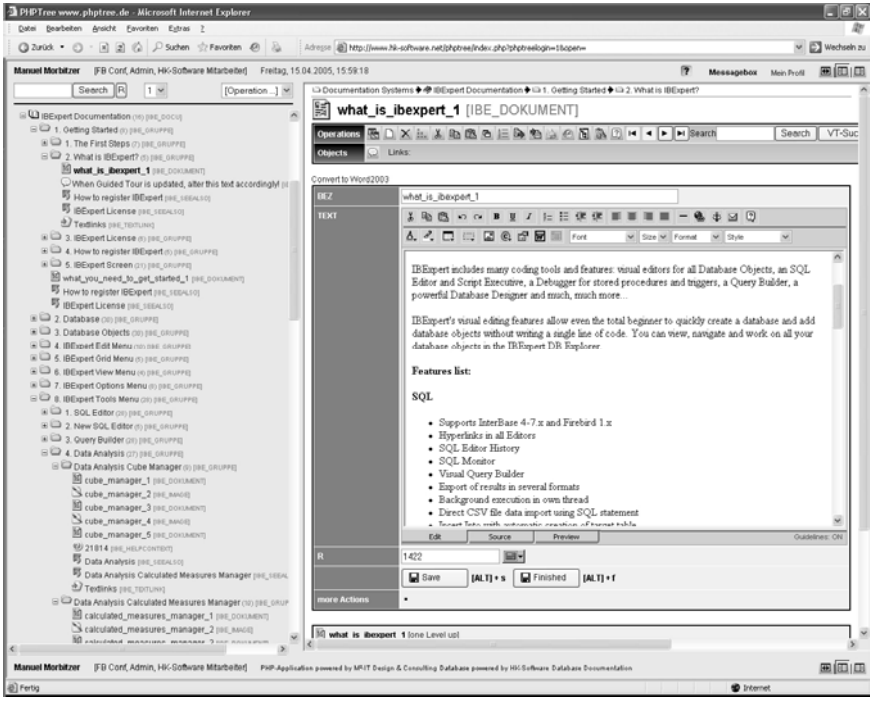
- Creation of media-independent documents, brochures, manuals, etc. with an easy-to-use editorial system
- Generation of any media (pdf, xml, html, wml, wordml, fop, …) by an extremely flexible parsing engine
- User-friendly and intuitive environment (customizable)
- Data can be easily moved per drag and drop
- Digital workflows
- Direct database access from a browser
- Platform independent
- Complete role administration
- Detailed rights administration for user groups and users, hereditary rights, additive and subtractive rights
- automatisms for rapid structuring of documents
- Plugins can be easily integrated (net structures, editorial and advertisement production, CMS and cross-media modules, document generation)
- and much more...

The editorial system

The editorial system is the part of PHPtree, which enables authors to edit and link texts, illustrations and other data formats as wished on a simple user interface, and format these using a range of

functions familiar from word processing (Word), such as, for example, cut and paste, font style etc. A WYSIWYG Editor has been implemented for this. The system can be accessed and operated from all popular web browsers, and can therefore be used on the internet or intranets.

Depending upon their user authorization, authors can administrate documents and pictures, classified in groups and objects, which are displayed in a clearly laid out tree structure (similar to the Windows file system). Objects (documents, pictures) can also be subsequently edited and altered. The entire tree structure can be viewed in the online documentation; in this book it can be viewed in a summarized form in the table of contents.



The above illustration depicts the editorial system. The documentation structure is displayed on the left-hand side. On the right-hand side the corresponding object for each element in the tree structure can be opened and edited. The illustration shows an open object for editing text.

Technical implementation

Both the online documentation and this book are constructed from the same data. All data is automatically stored media-independently, enabling publication in digital or print form. The format or layout is determined by templates, into which the data is inserted and which gives the texts and pictures their specific format or layout, and enables them to be processed for a specific medium. PHPtree provides a range of tools and plugins for the administration and management of templates. The structure hierarchy enables simple definition of how the data should be composed and with which structure this should then be combined with the templates to produce a finished document. The parsing kernel which undertakes the configuration of the data with the templates is extremely

flexible, as it can be constructed and configured individually using modules. It is also possible at this point to draw on data from any other desired data sources.

Demo

A selection of demo applications for digital and print media, along with further examples for catalogs can be found at www.phptree.de.

Contact:
M² IT Design & Consulting
Gerhard Stalling Strasse 47a
D-26135 Oldenburg
Germany

Tel +49 (0) 441 9507823
Fax +49 (0) 441 9507865
E-mail: info@m2-it.de