



Firebird 1.5.3 Release Notes

Helen Borrie (Collator/Editor)

23 Jan 2005 - Document version 153.06 - for Firebird 1.5.3 Release

Firebird 1.5.3 Release Notes

23 Jan 2005 - Document version 153.06 - for Firebird 1.5.3 Release
Helen Borrie (Collator/Editor)

Table of Contents

1. Firebird 1.5.3 Release Notes	8
This Edition	8
Previous Editions	8
General Notes	9
The Firebird 1.5 Binaries	9
Version Strings for Firebird 1.5 Releases	9
Documentation	9
2. New Features in Firebird 1.5	10
New Codebase, Better Optimization	10
Architecture	10
SQL Language	10
Installed Modules and Security	10
More Improvements	11
Trimming of Varchar fields for Remote Protocols	11
Multi-action Trigger Semantics	11
Enhancement to Named Constraints	12
Maximum Indexes per Table Increased	12
Pessimistic Locking	12
Security Database Connection Caching	12
Error-reporting Improvements	12
Services API on Classic for Linux	12
Changes in the Client Libraries	12
Windows clients	12
Linux clients	13
Renamed Files and Modules	13
All Platforms	13
All POSIX Platforms	13
32-bit Windows Platforms	14
Firebird 1.5.3 Point Release Additions	15
(1.5.3) Two ISQL Improvements	15
(1.5.3) Make Old Column Naming Convention available	15
(1.5.3) Security diagnostics added	15
(1.5.3) Closed an Endemic Security Hole	15
Firebird 1.5.2 Point Release Additions	15
(1.5.2) Performance improvement for permissions checking	15
(1.5.2) POSIX build and packaging changes	15
(1.5.2) POSIX improvements, FR # 1027636	16
(1.5.2) Changes to the standard ib_udf library declaration script	16
Firebird 1.5.1 Point Release Additions	16
(1.5.1) Introducing NPTL Builds for Higher Linuxen	16
(1.5.1) Services API is now fully supported on Classic	16
(1.5.1) GSTAT can now connect to localhost	17
(1.5.1) Character set NONE data now accepted "as is"	17
(1.5.1) Optional core dump on exceptions	18
(1.5.1) New collation added for Lithuanian language	18
(1.5.1) Small Win32 installation utility enhancement	18
3. Compatibility with Older Versions	19
On-disk Structure (ODS)	19
Firebird 1.0.n	19
InterBase® databases	19
File-names and Locations	20
Concurrently-running Servers	20

Reverting to Firebird 1.0.x	20
Linux Compatibilities	20
4. SQL Language Enhancements	21
Changes affecting All SQL	21
(1.5) Enhancement to single-line comment marker	21
(1.0) CURRENT_USER and CURRENT_ROLE	22
Data Definition Language (DDL)	23
New Data Types	23
(1.5) Enhancements to named constraints	23
(1.5) Multi-action triggers	24
(1.5) RECREATE VIEW	25
(1.5) CREATE OR ALTER { TRIGGER PROCEDURE }	26
(1.5) (1.0) Alter Trigger no longer increments the change count on tables	26
(1.5) NULLs in unique constraints and indices	26
(1.0) DROP GENERATOR	28
From Firebird v.1.0.x	28
Data Manipulation Language (DML)	28
(1.5) Expressions and variables as procedure arguments	29
(1.5) New constructs for CASE expressions	29
(1.5) SQL99-compliant Savepoints	31
(1.5) Explicit locking	33
(1.5) Improved Aggregate Handling	36
(1.5) ORDER BY clause can specify expressions and nulls placement	39
(1.5) SELECT FIRST 0.. is Now Valid	40
Other Firebird 1.0.x Features	42
Stored Procedure and Trigger Language (PSQL)	43
(1.5) EXECUTE STATEMENT	43
(1.5) New Context Variables	45
(1.5) Enhancements to Exception Handling in PSQL	47
(1.5) LEAVE BREAK statement	49
(1.5) Valid PLAN statements can now be included in triggers	50
(1.5) Empty BEGIN..END blocks	50
(1.5) Declare and define local variable in single statement	50
New Reserved Words	50
5. Miscellaneous Enhancements	52
Character Sets	52
V. 1.5	52
V. 1.5 (Binary Collations only)	52
V. 1.0.x	53
New ISQL Features	53
"readline" Capability in the ISQL Shell	54
External Functions (UDFs)	54
In the ib_udf Library	54
In the fbudf library	55
6. New Configuration Files	56
The Firebird Root Directory	56
Precedence Trail for Locating the Firebird Root	56
Server Configuration File--firebird.conf	57
Parameters	57
Database File Aliasing	68
Aliases.conf	68
Connecting using an aliased path	69
7. Firebird 1.5 Project Teams	70
"The Field Test Heroes"	72
8. INSTALLATION NOTES	73
Windows 32-bit Installs	73
READ THIS FIRST!	73

READ THIS NEXT!	75
Using the Win32 Firebird Installer	77
Installing Superserver from a zip kit	78
Installing Embedded Server from a Zip Kit	80
Other Win32 Issues	81
POSIX Platforms	82
READ THIS FIRST	82
Installing on Linux	82
Testing your Linux installation	84
Utility Scripts	86
Linux Server Tips	86
Uninstalling on Linux	87
Solaris	88
MacOS X	88
FreeBSD	88
Debian	88
9. Configuring the Port Service on Client and Server	89
How the server sets the listening port	89
Using the -p switch override	90
Syntax for TCP/IP	90
Syntax for WNet redirection	91
Classic on POSIX: the inetd or xinetd daemon	91
Using a configuration file parameter	92
Setting up a client to find the service port	92
Using the connection string	92
Using port syntax with database aliases	93
Using a copy of firebird.conf	94
Location of Firebird artifacts on clients	94
Configuring the Services File	94
Locating the services file	94
10. Further Information	96
Firebird Development	96
Lists and Newsgroups	96
Newsgroup Mirrors	96
Paid Support	97
Sponsorship	97
Tools and Drivers	97
Database Admin Tools	97
Drivers and Components	97
Documentation	98
11. Bugfixes and Additions since Release 1.0	100
Improvements	100
Release 1.5, 1.5.1 and 1.5.2 Bugs Fixed in v.1.5.3	111
Release 1.5 and 1.5.1 Bugs Fixed in v.1.5.2	113
Release 1.5 Bugs Fixed in v.1.5.1	114
Old Bugs Fixed	116
Known Issues at v.1.5.1	132

List of Tables

4.1. How TPB settings affect explicit locking	34
7.1. Firebird Development Teams	70

Chapter 1

Firebird 1.5.3 Release Notes

This Edition

- The Firebird 1.5.3 sub-release introduces a number of retrospective fixes to bugs that became apparent and were fixed in the Firebird 2 tree during the pre-alpha and alpha phases of the Firebird 2 development.
- Minor enhancements added since v.1.5.2 are described in [Firebird 1.5.3 Point Release Additions](#).
- Note the addition of a new optional compatibility parameter to `firebird.conf`. For details of this parameter--*OldColumnNaming*--refer to the section [Compatibility Parameters](#).
- Fixes to bugs introduced in v.1.5, v.1.5.1 and v.1.5.2 are appended to the feature list in the section [Release 1.5.2 Bugs Fixed in v.1.5.3](#).
- A further list of fixes is prepended to the [Old Bugs Fixed](#).

Previous Editions

Firebird 1.5.2 Release Notes

- Minor enhancements added since v.1.5.1 are described in [Firebird 1.5.2 Point Release Additions](#).
- Fixes to bugs introduced in v.1.5 and v.1.5.1 are appended to the feature list in the section [Release 1.5.1 Bugs Fixed in v.1.5.2](#).
- A further list of fixes is prepended to the [Old Bugs Fixed](#).

Firebird 1.5.1 Release Notes

Point release, Firebird 1.5.1, 14 July 2004 - Document version 151.00

- Minor enhancements added since v.1.5 were described in [Firebird 1.5.1 Point Release Additions](#).
- Fixes to bugs introduced in v.1.5 are appended to the feature list in the section [Release 1.5 Bugs Fixed in v.1.5.1](#).
- A further list of fixes is prepended to the [Old Bugs Fixed](#).
- A list of (currently one only) [Known Issues](#) is included.
- Some errata were corrected.

Firebird 1.5 Release Notes

Initial release, Firebird 1.5, 5 February 2004 - Document version 150.08

General Notes

The Firebird" database engine has been developed by an independent team of voluntary developers from the InterBase® source code that was released by Borland under the InterBase Public License v.1.0 on 25 July 2000.

Development on the Firebird 2 codebase began early in Firebird 1 development, with the porting of the Firebird 1 C code to C++ and the first major code-cleaning. Firebird 1.5 is the first release of the Firebird 2 codebase. It is a significant milestone for the developers and the whole Firebird project, but it is not an end in itself. As Firebird 1.5 goes to release, major redevelopment continues toward the next point release on the journey to Firebird 2.

Firebird 1.0.x continues in active maintenance, with important bug-fixes and enhancements being back-ported from 1.5.

The Firebird 1.5 Binaries

The Firebird binaries can be downloaded through the Firebird website [Downloads area](#). If you are looking for unusual builds that are not linked there, you might wish to plough through Firebird's [Sourceforge project download area](#).

Version Strings for Firebird 1.5 Releases

Win32: "WI-V1.5.0.nnnn Firebird 1.5"

Linux: "LI-V1.5.0.nnnn Firebird 1.5"

and so on, where "nnnn" is the build number.

Documentation

These release notes are currently your only source of free up-to-date documentation of changes that happened since Firebird 1.0.x. Please refer to the Documentation section for locations of recommended documentation.

Chapter 2

New Features in Firebird 1.5

New Codebase, Better Optimization

This release was built from code ported from the original C to C++, a process begun by Mike Nordell back in 2000. Extensive code cleanup and bug-fixing has continued, along with new memory management and language enhancements. Not least, during the v.1.5 development process, the SQL query optimizer has undergone enhancements and fixes at the hands of Arno Brinkman and others, resulting in reported speed improvement of 30 to 60 percent and more.

Architecture

Two significant new additions for Windows platforms are *Classic* server and *embedded* server.

Note that

- There has not been a Classic server on Windows for nearly eight years. This one can utilize multiple processors, something which still eludes the Windows Superserver. Though usable, Classic should be regarded as experimental.
- Embedded server is a dll that merges a single client attachment with a Firebird Superserver for building very quick and efficient stand-alone and briefcase applications.

SQL Language

Several important new language features have been added since version 1.0.x, including the SQL-92 conditional expression functions CASE, COALESCE and NULLIF. For syntax of these and other new language implementations, please refer to the Language Enhancements section later in this document.

Installed Modules and Security

If you have been using Firebird 1.0.x until now, you will notice big changes in the names of modules and the rules for accessing and locating them. Following are some highlights; for detailed information on installation, disk layout and configuration, see the relevant section in this document.

1. Most modules and constants have been renamed. In most cases, the new names involve some variant of “firebird” or “fb”. For example, the API library is now located in a shared library named “fbclient.dll” on Windows and “libfbclient.so” or “libfbembed.so” on other platforms.

The exception that breaks the rule is the security database, formerly named “isc4.gdb”, which is

now called “security.fdb”.

2. External files used by the server (UDF libraries, BLOB filters, character set libraries, external tables) are now subject to levels of filesystem protection that, in some cases, default to a level that will be different to what you had under Firebird 1.0.x or InterBase.
3. The new server configuration file, *firebird.conf*, that replaces *ibconfig* (Windows) and *isc_config* (other platforms) contains several new configurable features along with improved self-documentation and organisation.
4. A database-aliasing feature comes in 1.5. Now you can optionally “soft-code” the database location into your application code using your choice of alias to replace the path string. Actual path locations are stored in a text file, *aliases.conf*. The main purpose of aliasing, however, is to protect your physical paths from being maliciously “sniffed” on the wire.
5. The default (and past practice) on Windows server platforms makes it that the *localsystem* user runs the program that installs the Firebird service at system start-up. This could be a serious security vulnerability if the Firebird server should be hacked, since it provides a window through which the hacker can access the entire machine. The 1.5 version of this program (*instsvc.exe*) now accepts a Windows user log-on for the service installation. It is strongly recommended that you create a Firebird user for this purpose and make use of the new logon feature if your server is connected to the Internet in any way.

More Improvements

Trimming of Varchar fields for Remote Protocols

Work was resumed and completed on this tricky feature for the 1.5 client and varchars now cross the wire right-trimmed to actual length plus two bytes.

Note

As it is the client that requests the server to trim varchars, the Firebird 1.5 client (*fbclient.dll* or *libfbclient.so*) will trim, even if connected to a pre-1.5 server version. If you use an old client, you will not get trimming, even if you are connected to a 1.5 or later server.

Multi-action Trigger Semantics

Now you can write conditional insert/update/delete actions in one Before or After trigger to have the one trigger cover all DML actions for that trigger's phase. This cuts down the composition and maintenance of triggers without deprecating the ability to have multiple triggers per phase.

Enhancement to Named Constraints

Indexes that enforce named integrity constraints may now be named with user-defined identifiers.

Caution

If you use this feature, your database will not be downgradable to v.1.0.x or InterBase®.

Maximum Indexes per Table Increased

Now-in both Release 1.0 and this release-the maximum number of indexes you can define for a table has been increased from 64 to 256.

Pessimistic Locking

For the rare times when you need to impose a pessimistic lock, this release adds syntax to place a “reader's lock” on rows as they are output to the client. Use with care.

Security Database Connection Caching

Connection to the security database is cached in Superserver builds. It means that `security.fdb` is loaded when the first connection is made and stays attached until all client connections are gone.

Error-reporting Improvements

Where possible, error messages report the cause of SQL errors at a more detailed level. It is **IMPORTANT** to note that you will encounter bizarre messages if you use an old `interbase.msg` or `firebird.msg` file.

Services API on Classic for Linux

Limited support for the Services API is now available on Classic server on Linux. Services available from `gbak` (backup/restore) and `gfix` (validate database, shutdown/online, etc.) work. Others (`gstat`, server logs, etc.) were not tested and are probably non-functional.

Changes in the Client Libraries

Windows clients

The client library is now named “fbclient.dll”. All server utilities (`gbak`, `gfix`, etc) use only the client library `fbclient.dll`. Connect new applications to `fbclient.dll`, without requiring `gds32.dll` (recommended).

“Compatibility” Client

For compatibility with existing applications, it is possible to generate a “clone” of fbclient.dll with the name “gds32.dll” using a new utility named *instclient.exe*. For exact details, see the installation section and any late installation notes distributed with your Windows kit.

Linux clients

The Superserver client library is now named “libfbclient.so”. For compatibility with existing applications, a symlink “libgds.so” is installed that points back to libfbclient.so. The local client library for embedded applications connecting to Classic server has been renamed to libfbembed.so.

Renamed Files and Modules

All Platforms

Platform	Module	Firebird 1.0	Firebird 1.5	Special notes
All	Environment variables	INTERBASE INTERBASE_LOCK INTERBASE_MSG INTERBASE_TMP	FIREBIRD FIREBIRD_LOCK FIREBIRD_MSG FIREBIRD_TMP	Points to the installation root path Points to location of lock file Points to location of message file Points to a directory for sort space
All	Security database	isc4.gdb	security.fdb	
All	Message file	Interbase.msg	firebird.msg	
All	Server log file	interbase.log	firebird.log	
All	ODS version	10	10.1	New ODS (10.1). doesn't cause any incompatibilities with previous ODS but version is not upgraded automatically. Firebird 1.0 and 1.5 both can serve ODS 10.0 and 10.1 DBs. Nevertheless, backup/restore is still the recommended procedure for migrating DBs to a different version of the server.

All POSIX Platforms

Platform	Module	Firebird 1.0	Firebird 1.5	Special notes
Linux	Configuration file	isc_config	firebird.conf	
Linux	Client library	libgds.so	libfbclient.so libfbembed.so	Thread-safe remote client and TCP/IP local loopback client for Superserver Local client (single-user, non-thread-safe) for Classic
Linux	Client library symlink for compatibility	N/A	libgds.so	
Linux	Classic server binary	gds_inet_server	fb_inet_server	
Linux	Classic lock manager	ib_lock_mgr	fb_lock_mgr	
Linux	Superserver control	ibmgr.bin	fbmgr.bin	
Linux	Superserver binary	ibserver	fbserver	

32-bit Windows Platforms

Platform	Module	Firebird 1.0	Firebird 1.5	Special notes
Windows	Guardian	ibguard.exe	fbguard.exe	
Windows	Superserver binary	ibserver.exe	fbserver.exe	Not multi-processor capable.
Windows	Classic binary	N/A	fb_inet_server.exe	Windows local connect not available. TCP/IP, NetBEUI OK. Multi-processor capable.
Windows	Client library	gds32.dll	fbclient.dll	Fb 1.5 versions of server utilities, and all new applications, need only fbclient.dll. See notes below regarding gds32.dll compatibility for old applications.
Windows	Configuration file	ibconfig	firebird.conf	
Windows	Local IPC port	InterBase\IPC	Firebird\IPC	With default server settings you cannot do local connect from applications using an old client library (gds32.dll). If necessary, you can set up the server to use the old name of the IPC map, via firebird.conf.
Windows	Default Registry key	HKLM\SOFTWARE\Borland\InterBase	HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances	The path is stored in the "DefaultInstance" parameter. i.e. no more "CurrentVersion" key, and "RootDirectory" is replaced with "DefaultInstance".
The new service names on Windows are "Firebird Guardian - DefaultInstance" and "Firebird Server - DefaultInstance".				

Firebird 1.5.3 Point Release Additions

The following group of minor enhancements was added to the Firebird v.1.5.3 point release.-

(1.5.3) Two ISQL Improvements

C. Valderrama, D. Ivanov

1. Command line switch -b to bail out on error when used in non-interactive mode.
2. Return an error code to the operating system from command-line isql

(1.5.3) Make Old Column Naming Convention available

Paul Reeves

Added *OldColumnNaming* parameter to firebird.conf to allow users to revert to pre-V1.5 column naming in Select expressions.

(1.5.3) Security diagnostics added

Alex Peshkoff

Attempts to send signals via a missing gds_relay may be an exploit attempt. They are now logged.

(1.5.3) Closed an Endemic Security Hole

Alex Peshkoff

Previously, a user could log into a server on a Unix/Linux host remotely, using a Linux UID and password accepted on that host. It was recognised as a security hole and fixed in Firebird 2 development. It is an endemic security bug in previous versions and InterBase. The security fix has been back-ported to Firebird 1.5.3: a UID received from the client side is now not trusted.

Firebird 1.5.2 Point Release Additions

The following group of minor enhancements was added to the Firebird v.1.5.2 point release.-

(1.5.2) Performance improvement for permissions checking

N. Samofatov, D. Urban

Resource lists to check permissions are now computed on the fly as needed. For complex schemas, this significantly reduces memory and CPU time consumption.

(1.5.2) POSIX build and packaging changes

Nickolay Samofatov

- Work around bugs in GCC 3.3.2 and 3.3.3
- Support GCC 3.4 build
- Limit exports of Firebird libraries using version script
- Link client library and UDF libraries with POSIX threads. This cures problems with single-threaded hosts like PHP linking with libfbclient.so from CS packages

(1.5.2) POSIX improvements, FR # 1027636

D. Mullins, E. S. LaBianca

To prevent the startup status from being overwritten by the next status message, the `/etc/init.d/firebird` script needed to have a line consisting only of "echo" after `RETVAL=$?`.

Erik LaBianca extended the Firebird build system to generate source bundles in a generic fashion and without autoconf dependency. He uses this facility for his Fedora Core packages.

(1.5.2) Changes to the standard `ib_udf` library declaration script

Nickolay Samofatov

The default declarations of the string manipulation routines in `ib_udf.sql` were altered to accept strings with lengths up to 255 characters.

Firebird 1.5.1 Point Release Additions

Important

These are point release notes that augment the release notes for v.1.5.

(1.5.1) Introducing NPTL Builds for Higher Linuxen

Alex Peshkoff

Firebird Superserver has a link-time backward compatibility issue with the NPTL (Native POSIX Thread Library) that may cause it to be unstable on Linux distributions that enable the NPTL in the GNU C library, e.g. Red Hat 9, Mandrake 10, Fedora Core. The new NPTL builds of Superserver should solve these problems.

(1.5.1) Services API is now fully supported on Classic

Nickolay Samofatov

Features of GSEC and GSTAT are now supported via the Services API in CS builds. It means that the entire Services API now works in both Superserver and Classic architectures.

Known issue: GSEC doesn't deliver error status vectors to the client side if forked from the CS process. Any error will prevent the security database from being changed; but the exception is decoded by the server and the appropriate message is delivered via the API communication buffer instead of

the status vector. If this situation is not handled properly by the user program, it may cause application-specific errors.

For example, IBExpert displays the message "unexpected output buffer value".

(1.5.1) GSTAT can now connect to localhost

Dmitry Yemanov

GSTAT supported only the local connection string syntax: you could not specify "localhost:<path>" to retrieve the statistics for a local database. Since Win32 Classic does not yet support the local (IPC) protocol, it wasn't possible to use GSTAT with it. Now GSTAT is fixed to enable it.

You may use either "localhost:" (for TCP/IP) or "\\.\\" (for Named Pipes) to work around the Win32 Classic limitation.

Note

GSTAT still can not be run over remote databases. Because it has to open a database file directly to read the header page, it requires local access to the database file. This might change in a future version.

(1.5.1) Character set NONE data now accepted "as is"

J. Beesley, N. Samofatov

Changes were made in the engine to make the character set NONE more friendly about reading and writing data from and to fields (columns, variables) of another character set.

In Firebird 1.5.0, from a client connected with character set NONE, you could read data in two incompatible character sets—such as SJIS (Japanese) and WIN1251 (Russian)—even though you could not read one of those character sets while connected from a client with the other character set. Data would be received "as is" and be stored without raising an exception.

However, from this character set NONE client connection, an attempt to update any Russian or Japanese data columns using either parameterized queries or literal strings without introducer syntax would fail with transliteration errors; and subsequent queries on the stored "NONE" data would similarly fail.

In Firebird 1.5.1, both problems have been circumvented. Data received from the client in character set NONE are still stored "as is" but what is stored is an exact, binary copy of the received string. In the reverse case, when stored data are read into this client from columns with specific character sets, there will be no transliteration error. When the connection character set is NONE, no attempt is made in either case to resolve the string to well-formed characters, so neither the write nor the read will throw a transliteration error.

This opens the possibility for working with data from multiple character sets in a single database, as long as the connection character set is NONE. The client has full responsibility for submitting strings in the appropriate character set and converting strings returned by the engine, as needed.

Abstraction layers that have to manage this can read the low byte of the *sqlsubtype* field in the XSQLVAR structure, which contains the character set identifier.

While character set NONE literals are accepted and implicitly stored in the character set of their context, the use of introducer syntax to coerce the character sets of literals is highly recommended when

the application is handling literals in a mixture of character sets. This should avoid the string's being misinterpreted when the application shifts the context for literal usage to a different character set.

Note

Coercion of the character set, using the introducer syntax or casting, is still required when handling heterogeneous character sets from a client context that is anything but NONE.

Introducer syntax

```
_ISO8859_1 'ààààà'
```

Casting

```
CAST (<string> as varchar(n) character set ISO8859_1)
```

(1.5.1) Optional core dump on exceptions

Nickolay Samofatov

A debugging enhancement was added, to configure the server to abort a server process and produce a core dump when bugchecks or structured exceptions occur. The new parameter in `firebird.conf` is `BugcheckAbort`. It is off (=0) by default.

If turned on, this feature will produce a correct core dump on BUGCHECK or when an external function (UDF, BLOB filter, intl2 function) causes havoc. When `BugcheckAbort` is not enabled, structured exception handlers or a synchronous signal handler may mask the original cause of problem.

(1.5.1) New collation added for Lithuanian language

Jonas Jasas Jr

Collation sequence `LT_LT` was added for the `ISO8859_13` charset.

(1.5.1) Small Win32 installation utility enhancement

Olivier Mascia

The Win32 service installer now adds a description string to the services configuration info.

Chapter 3

Compatibility with Older Versions

This chapter describes compatibility issues for migrating existing databases to Firebird 1.5.

On-disk Structure (ODS)

The On-Disk Structure of Firebird 1.5 is designated ODS 10.1. This minor ODS upgrade was required because of three changes:

- three new indices for system tables
- minor changes in the BLR of two system triggers
- enhanced encoding of RDB\$TRIGGER_TYPE

Firebird 1.0.n

Certain enhancements requiring changes to the ODS have been deferred to the version 2 release. Meanwhile, you should be able to transport your Firebird 1.0.x databases directly. Take tested backups of your Firebird 1.0.x databases before porting them to 1.5.

InterBase® databases

If you are planning to “play” with Firebird using an existing InterBase database, with the intention of reverting to InterBase later, please take all precautions to back up your current version using the appropriate InterBase version of gbak. For beginning to work with your database in Firebird 1.5, use the Firebird 1.5 version of gbak to restore your backup.

Note

The *Operations Guide (OpGuide.pdf)* from the InterBase® 6.0 beta documentation set contains the command syntax for the gbak backup and restore program.

IB 7.x and probably IB 6.5 databases may work incorrectly after migration to FB 1.5 via backup/restore, if new IB-specific features were used in those databases.

File-names and Locations

In this release, a substantial number of software files have new names, as part of a gradual replacement of names inherited from InterBase® 6. Please read the section [Renamed Files and Modules](#) for descriptions and recommendations.

Concurrently-running Servers

Changes done to some system object names enable Fb 1.5 to be installed and used on a computer which already has InterBase or Firebird 1.0.x installed. On Windows, FB 1.5 also uses another Registry key--and use of the Registry entry is optional. If you set up the servers to use different network ports, it is possible run a few server instances concurrently, or run Fb 1.5 while IB or Fb 1.0.x is running.

Reverting to Firebird 1.0.x

Because of a large number of bug-fixes, the behaviour of databases might change if you downgrade a v.1.5 database to v.1.0.x. Especially, if you create primary, unique and foreign keys as named constraints, the default index names will be incompatible with v.1.0.x. Watch out for a future README detailing any such issues as might appear.

Linux Compatibilities

Because of a history of problems with the GNU C++ compiler, Firebird 1.5 Linux versions need higher versions of the glibc runtimes than previously. This means, unfortunately, that we are in a period where the capability of a particular distro to install and run the 1.5 binaries is somewhat hard to predict.

The following matrix may help. However, we welcome further information. Please share your experiences with these and other distros in the [firebird-devel](#) forum.

Distro	Level	Classic	Superserver
Red Hat	7.x	No	No
	8.0	Yes	Yes
Mandrake	9.0	Yes	Yes
	8.x	No	No
	9.0, 9.1, 9.2	Yes	Yes
SuSE	7.3	Yes - install packages libgcc-3.2-44.i586.rpm & libstdc++-3.2-44.i586.rpm before installing Firebird 1.5.	Not known
	8.0, 8.1	Yes	8.0 Yes(8.1 Not known)

Known Firebird 1.5-compatible Linux Distributions

SQL Language Enhancements

A large number of additions and enhancements were made to the various subsets of Firebird's SQL language.

Changes affecting All SQL

(1.5) Enhancement to single-line comment marker

Dmitry Yemanov

Single-line comments can be in any position in the line, not just the first. For use in scripts, DSQL, stored procedures and triggers.

So, in 1.5, the "--" marker can be used for a comment at the end of a line statement in a script, stored procedure, trigger or DSQL statement. It can thus be used to "comment out" unwanted parts of statements. All characters from the "--" marker until the next carriage return or line feed will be ignored.

```
...  
WHERE COL1 = 9 OR COL2 = 99 -- OR COL3 = 999
```

Previous implementation (v.1.0, Claudio Valderrama)

```
-- This is a comment
```

This new marker can be used for "commenting out" a single line of code in a script, DDL/DML statement, stored procedure or trigger. The logic is to ignore characters is as follows:

1. Skip '--' if it is found as the first character pair following an end-of-line marker (LF on Linux/Unix, CRLF on Windows)
2. Continue skipping characters until the next end-of-line marker

This logic is NOT intended for mixing with the block comment logic (/* a comment */). In other words, don't use the '--' style of commenting within a block comment and don't use the block-style of commenting within a '--' line.

Interactive isql sessions

Keep this in mind when working in an interactive isql session. isql will accept pieces of a statement in separate continuation segments, displaying the 'CON>' prompt until it receives the terminator symbol (normally ';'). If you type a '--' pair at the start of a continuation line, the ignoring logic will finish at

the end-of-line marker that is printed to the screen or your OUTPUT file when you press Enter.

There is potential for errors if you subsequently add a continuation, expecting it to be ignored.

The problem with isql arises because it has its own special commands that should be parsed only by isql. If they are not recognized due to tricky placement of "--", then they are passed to the engine. Obviously, the engine doesn't understand isql's SET and SHOW commands and rejects them.

(1.0) *CURRENT_USER and CURRENT_ROLE*

These two new context variables have been added to reference the USER and (if implemented) the ROLE of the current connection context.

Examples

```
CREATE GENERATOR GEN_USER_LOG;
CREATE DOMAIN INT_64 AS NUMERIC(18,0);
COMMIT;
CREATE TABLE USER_LOG(
    LOG_ID INT_64 PRIMARY KEY NOT NULL,
    OP_TIMESTAMP TIMESTAMP,
    LOG_TABLE VARCHAR(31),
    LOG_TABLE_ID INT_64,
    LOG_OP CHAR(1),
    LOG_USER VARCHAR(8),
    LOG_ROLE VARCHAR(31));

COMMIT;

CREATE TRIGGER ATABLE_AI FOR ATABLE
ACTIVE AFTER INSERT POSITION 0 AS
BEGIN
    INSERT INTO USER_LOG VALUES(
        GEN_ID(GEN_USER_LOG, 1),
        CURRENT_TIMESTAMP,
        'ATABLE',
        NEW.ID,
        'I',
        CURRENT_USER,
        CURRENT_ROLE);
END
```

CURRENT_USER is a DSQL synonym for USER that appears in the SQL standard. They are identical. There is no advantage of CURRENT_USER over USER.

1. If you insist on using an InterBase v.4.x or 5.1 database with Firebird, ROLE is not supported, so current_role will be NONE (as mandated by the SQL standard in absence of an explicit role) even if the user passed a role name.
2. If you use IB 5.5, IB 6 or Firebird, the ROLE passed is verified. If the role does not exist, it is reset to NONE without returning an error.

This means that in FB you can never get an invalid ROLE returned by CURRENT_ROLE, because it will be reset to NONE. This is in contrast with IB, where the bogus value is carried internally, although it is not visible to SQL.

Data Definition Language (DDL)

DDL is the language subset that is used for creating, altering and dropping metadata. The following changes have been implemented:

New Data Types

(1.5) BIGINT

BIGINT is the new SQL99-compliant name for the 64-bit signed, exact numeric type, with a scale of zero. Available in Dialect 3 only.

Examples

i)

```
DECLARE VARIABLE VAR1 BIGINT;
```

ii)

```
CREATE TABLE TABLE1 (FIELD1 BIGINT);
```

(1.5) Enhancements to named constraints

Dmitry Yemanov

Indexes that enforce named constraints may now be named with user-defined identifiers.

Previously, although it was possible to create named PRIMARY, FOREIGN KEY and UNIQUE constraints, the identifier of the automatically-generated enforcing index was calculated by the system, e.g., RDB\$FOREIGN13, and could not be altered. This remains the default behaviour when named constraints are not used.

However, language extensions have been added to enable

1. a system-generated index to receive automatically the same identifier as the named constraint it enforces
2. an index which enforces a named or unnamed constraint to be explicitly assigned a custom identifier and to be optionally constructed in DESCENDING order.

Note

It is not currently possible to use a pre-existing index.

Syntax Pattern

...

```
[ADD] CONSTRAINT [<constraint-identifier>]
<constraint-type> <constraint-definition>
[USING [ASC[ENDING] | DESC[ENDING]] INDEX <index_name>]
```

Caution

Make sure that foreign key and primary key indexes use the same sort order (DESC | ASC).

Examples

i) Named constraint and explicitly-named index:

```
CREATE TABLE ATEST (
    ID BIGINT NOT NULL,
    DATA VARCHAR(10));
COMMIT;
```

The following statement will create a primary key constraint named PK_ATEST and an enforcing, descending index named IDX_PK_ATEST:

```
ALTER TABLE ATEST
ADD CONSTRAINT PK_ATEST PRIMARY KEY(ID)
USING DESC INDEX IDX_PK_ATEST;
COMMIT;
```

ii) Alternative to i) above:

```
CREATE TABLE ATEST (
    ID BIGINT NOT NULL,
    DATA VARCHAR(10),
    CONSTRAINT PK_ATEST PRIMARY KEY(ID)
    USING DESC INDEX IDX_PK_ATEST;
```

iii) This statement creates the table ATEST with the primary key PK_ATEST. The enforcing index is also named PK_ATEST.:

```
CREATE TABLE ATEST (
    ID BIGINT NOT NULL,
    DATA VARCHAR(10),
    CONSTRAINT PK_ATEST PRIMARY KEY(ID));
```

(1.5) Multi-action triggers

Dmitry Yemanov

Triggers are enhanced to enable them to handle multiple row-level operations conditionally.

Syntax Pattern

```
CREATE TRIGGER name FOR table
[ACTIVE | INACTIVE]
```



```
{BEFORE | AFTER} <multiple_action>
[POSITION number]
AS trigger_body

<multiple_action> ::= <single_action> [OR <single_action> [OR <single_action>]]
<single_action> ::= {INSERT | UPDATE | DELETE}
```

Examples

i)

```
CREATE TRIGGER TRIGGER1 FOR TABLE1
ACTIVE BEFORE INSERT OR UPDATE AS
...;
```

ii)

```
CREATE TRIGGER TRIGGER2 FOR TABLE2
ACTIVE AFTER INSERT OR UPDATE OR DELETE AS
...;
```

ODS Change

Encoding of field RDB\$TRIGGER_TYPE (relation RDB\$TRIGGERS) has been extended to allow complex trigger actions. For details, refer to the document `readme.universal_triggers.txt` in the `/doc/sql.extensions` branch of the Firebird CVS tree.

Notes

1. One-action triggers are fully compatible at ODS level with FB 1.0.
2. RDB\$TRIGGER_TYPE encoding is order-dependant, i.e., BEFORE INSERT OR UPDATE and BEFORE UPDATE OR INSERT will be coded differently, although they have the same semantics and will be executed exactly the same way.
3. Both OLD and NEW context variables are available in multiple-action triggers. If the trigger invocation forbids one of them (e.g. OLD context for INSERT operation), then all fields of that context will evaluate to NULL. If they are assigned to an improper context, a runtime exception will be thrown.
4. The new Boolean context variables INSERTING/UPDATING/DELETING can be used to check the operation type at runtime. (See below.)

(1.5) RECREATE VIEW

Exactly the same as CREATE VIEW if the view does not already exist. If it does exist, RECREATE VIEW will try to drop it and create a completely new object. RECREATE VIEW will fail if the object is in use.

Uses the same syntax as CREATE VIEW.

(1.5) CREATE OR ALTER {TRIGGER | PROCEDURE }

Statement that will either create a new trigger or procedure (if it does not already exist) or alter it (if it already exists) and recompile it. The CREATE OR ALTER syntax preserves existing dependencies and permissions.

Syntax is as for CREATE TRIGGER | CREATE PROCEDURE, respectively, except for the additional keywords "OR ALTER".

(1.5) (1.0) Alter Trigger no longer increments the change count on tables

When the count of metadata changes on any single table reaches the maximum of 255, the database becomes unavailable. Backup and restore are required in order to reset the change count and make the database once again available. The intention of this feature is to enforce a database cleanup when table structures have undergone a lot of changes, not to inhibit useful capabilities in the engine.

Previously, each time a trigger was set ACTIVE|INACTIVE by an ALTER TRIGGER statement, the change count for the associated table would be incremented. This affected the usefulness of disabling and re-enabling trigger code for regular operations, since it would cause the change count to rise quickly. Now, it is not treated as a metadata change for table versioning purposes.

(1.5) NULLs in unique constraints and indices

Dmitry Yemanov

It is now possible to apply a UNIQUE constraint or a unique index to a column that does not have your database to Firebird 1.0.x or any InterBase version.

Syntax Details

```
<unique constraint or index definition> ::=  
  <unique specification> ( <unique column list UCL> )  
<unique specification> ::=  
  {[constraint-name]UNIQUE | UNIQUE INDEX index-name} |  
  [constraint-name] PRIMARY KEY}
```

where <unique column list> can contain one or more columns without the NOT NULL attribute, if <unique specification> is UNIQUE or UNIQUE INDEX index-name.

Caution

All columns in PRIMARY KEY still must be declared NOT NULL.

The constraint allows existence of only those rows for which search condition (i) or (ii) evaluates as True, according to the following logic:

1. If the <unique specification> specifies PRIMARY KEY, then the search condition shall be:

```
UNIQUE ( SELECT UCL FROM TN ) AND ( UCL ) IS NOT NULL
```

2. Otherwise, the <search condition> shall be:

```
UNIQUE ( SELECT UCL FROM TN )
```

In this case, the condition `UNIQUE` can not be True if `(SELECT UCL FROM TN)` could output two rows where all of the corresponding non-null segment pairs match.

The constraint allows existence of only those rows for which the aforementioned <search condition> evaluates to True. In a unique index or under a `UNIQUE` constraint, two sets of column values will be considered distinct and thus allowed if:

1. both sets contain only nulls, or
2. there is at least one pair of corresponding values of which one is non-null, and the other either null or a different non-null value.

Examples

UNIQUE constraint:

```
CREATE TABLE t (
  a INTEGER,
  b INTEGER,
  CONSTRAINT pk UNIQUE (a, b));
```

or UNIQUE index:

```
CREATE TABLE t (a INTEGER, b INTEGER);
COMMIT;

CREATE UNIQUE INDEX uqx ON t(a, b);
COMMIT;

INSERT INTO t
VALUES (NULL, NULL); /* ok, nulls allowed */

INSERT INTO t
VALUES (1, 2); /* as are non-nulls */

INSERT INTO t
VALUES (1, NULL); /* and combinations */

INSERT INTO t
VALUES (NULL, NULL); /* ok, all pairs of nulls are distinct */
```

but not:

```
INSERT INTO t
```

```
VALUES (1, NULL);  
/* fails because all corresponding non-null segments match */
```

It means that the PRIMARY KEY constraint doesn't allow NULLs whilst the UNIQUE constraint and unique indexes allow an arbitrary number of NULLs. For multi-column result sets of (SELECT UCL FROM TN), the common rules for NULLs are applied, i.e. (1, NULL) is distinct from (NULL, 1) and one (NULL, NULL) is distinct from any other (NULL, NULL).

(1.0) DROP GENERATOR

Enables unused generators to be removed from the database. Storage will be freed for re-use upon the next RESTORE. Available in SQL and DSQL.

Syntax Pattern

```
DROP GENERATOR <generator name>;
```

From Firebird v.1.0.x

The following were implemented in Firebird 1.0. They are described again here for the convenience of the reader.

(1.0) RECREATE PROCEDURE

This new DDL command lets you create a new stored procedure with the same name as an existing procedure, replacing the old procedure, without needing to drop the old procedure first. The syntax is identical to CREATE PROCEDURE.

Available in SQL and DSQL.

(1.0) RECREATE TABLE

This new DDL command lets you create a new structure for an existing table without needing to drop the old table first. The syntax is identical to CREATE TABLE.

Note

Observe that RECREATE TABLE does not preserve the data in the old table.

Available in SQL and DSQL.

Data Manipulation Language (DML)

Data manipulation language, or *DML* is the language of query statements, the commands we use to *manipulate data*, that is, to SELECT FROM, INSERT, UPDATE and DELETE from tables and to EXECUTE PROCEDURES.

(1.5) Expressions and variables as procedure arguments

Dmitry Yemanov

Calls to EXECUTE PROCEDURE ProcName(<Argument-list>) and SELECT <Output-list> FROM ProcName(<Argument-list>) can now accept local variables (in PSQL) and expressions (in DSQL and PSQL) as arguments.

(1.5) New constructs for CASE expressions

Arno Brinkman

a) CASE

Allow the result of a column to be determined by the outcome of a group of exclusive conditions.

Syntax Pattern

```
<case expression> ::=
    <case abbreviation> | <case specification>

<case abbreviation> ::=
    NULLIF <left paren> <value expression> <comma> <value expression> <right paren>
    | COALESCE <left paren> <value expression> { <comma> <value expression> }... <right paren>

<case specification> ::=
    <simple case> | <searched case>

<simple case> ::=
    CASE <value expression> <simple when clause>...
    [ <else clause> ]
    END

<searched case> ::=
    CASE <searched when clause>...
    [ <else clause> ]
    END

<simple when clause> ::= WHEN <when operand> THEN <result>
<searched when clause> ::= WHEN <search condition> THEN <result>
<when operand> ::= <value expression>
<else clause> ::= ELSE <result>
<result> ::= <result expression> | NULL
<result expression> ::= <value expression>
```

Examples

i) simple

```
SELECT
    o.ID,
    o.Description,
    CASE o.Status
        WHEN 1 THEN 'confirmed'
        WHEN 2 THEN 'in production'
        WHEN 3 THEN 'ready'
        WHEN 4 THEN 'shipped'
        ELSE 'unknown status ' || o.Status || ''
    END
```

```

END
FROM Orders o;

```

ii) *searched*

```

SELECT
  o.ID,
  o.Description,
  CASE
    WHEN (o.Status IS NULL) THEN 'new'
    WHEN (o.Status = 1) THEN 'confirmed'
    WHEN (o.Status = 3) THEN 'in production'
    WHEN (o.Status = 4) THEN 'ready'
    WHEN (o.Status = 5) THEN 'shipped'
    ELSE 'unknown status ' || o.Status || ' '
  END
FROM Orders o;

```

b) COALESCE

Allows a column value to be calculated by a number of expressions, from which the first expression to return a non-NULL value is returned as the output value.

Syntax Pattern

```

<case abbreviation> ::=
  | COALESCE <left paren> <value expression>
  { <comma> <value expression> }... <right paren>

```

Syntax Rules

1. COALESCE (V1, V2) is equivalent to the following <case specification>:

```

CASE WHEN V1 IS NOT NULL THEN V1 ELSE V2 END

```

2. COALESCE (V1, V2,..., Vn), for n >= 3, is equivalent to the following <case specification>:

```

CASE WHEN V1 IS NOT NULL THEN V1
      ELSE COALESCE (V2,...,Vn) END

```

Examples

```

SELECT
  PROJ_NAME AS Projectname,
  COALESCE(e.FULL_NAME,'[< not assigned >]') AS EmployeeName
FROM
  PROJECT p
  LEFT JOIN EMPLOYEE e
  ON (e.EMP_NO = p.TEAM_LEADER);

```

```
SELECT
    COALESCE(Phone, MobilePhone, 'Unknown') AS "Phonenumber"
FROM
    Relations;
```

c) **NULLIF**

Returns NULL for a sub-expression if it has a specific value, otherwise returns the value of the sub-expression.

Syntax Pattern

```
<case abbreviation> ::=
    NULLIF <left paren> <value expression> <comma> <value expression> <right paren>
```

Syntax Rules

NULLIF (V1, V2) is equivalent to the following <case specification>:

```
CASE WHEN V1 = V2 THEN NULL ELSE V1 END
```

Example

```
UPDATE PRODUCTS
SET STOCK = NULLIF(STOCK, 0)
```

(1.5) SQL99-compliant Savepoints

Nickolay Samofatov

User savepoints (alternative name nested transactions) provide a convenient method to handle business logic errors without needing to roll back the transaction. Available only in DSQL.

Use the SAVEPOINT statement to identify a point in a transaction to which you can later roll back.

Syntax Patterns

```
SAVEPOINT <identifier>;
```

<identifier> specifies the name of a savepoint to be created. After a savepoint has been created, you can either continue processing, commit your work, roll back the entire transaction, or roll back to the savepoint.

Savepoint names must be distinct within a given transaction. If you create a second savepoint with the same identifier as an earlier savepoint, the earlier savepoint is erased.

```
ROLLBACK [WORK] TO [SAVEPOINT] <identifier>;
```

This statement performs the following operations:

- Rolls back changes performed in the transaction after the savepoint
- Erases all savepoints created after that savepoint. The named savepoint is retained, so you can roll back to the same savepoint multiple times. Prior savepoints are also retained.
- Releases all implicit and explicit record locks acquired since the savepoint. Other transactions that have requested access to rows locked after the savepoint must continue to wait until the transaction is committed or rolled back. Other transactions that have not already requested the rows can request and access the rows immediately.

Note

This behaviour may change in future product versions.

Important

The Savepoint undo log may consume significant amounts of server memory, especially if you update the same records in the same transaction multiple times. Use the `RELEASE SAVEPOINT` statement to release system resources consumed by savepoint maintenance.

```
RELEASE SAVEPOINT <identifier> [ONLY];
```

`RELEASE SAVEPOINT` statement erases the savepoint <identifier> from the transaction context. Unless you specify the `ONLY` keyword, all savepoints established since the savepoint <identifier> are erased too.

Example using Savepoints

```
create table test (id integer);
commit;
insert into test values (1);
commit;
insert into test values (2);
savepoint y;
delete from test;
select * from test; -- returns no rows
rollback to y;
select * from test; -- returns two rows
rollback;
select * from test; -- returns one row
```

Internal savepoints

By default, the engine uses an automatic transaction-level system savepoint to perform transaction rollback. When you issue a `ROLLBACK` statement, all changes performed in this transaction are backed out via a transaction-level savepoint and the transaction is then committed. This logic reduces the amount of garbage collection caused by rolled back transactions.

When the volume of changes performed under a transaction-level savepoint is getting large (10^4 - 10^6 records affected) the engine releases the transaction-level savepoint and uses the TIP mechanism to roll back the transaction if needed.

Tip

If you expect the volume of changes in your transaction to be large, you can use the TPB flag `isc_tpb_no_auto_undo` to avoid the transaction-level savepoint being created.

Savepoints and PSQL

Implementing user savepoints in PSQL layer would break the atomicity rule for statements, including procedure call statements. Firebird provides exception handling in PSQL to undo changes performed in stored procedures and triggers. Each SQL/PSQL statement is executed under a system of automatic, internal savepoints, whereby either the entire statement will complete successfully or ALL its changes are rolled back and an exception is raised.

Each PSQL exception handling block is also bounded by automatic system savepoints.

(1.5) Explicit locking

Nickolay Samofatov

The addition of the optional WITH LOCK clause provides a limited explicit pessimistic locking capability for cautious use in conditions where the affected row set is

1. extremely small (ideally, a singleton) AND
2. precisely controlled by the application code.

Caution**This is for experts only!**

The need for a pessimistic lock in Firebird is very rare indeed and should be well understood before use of this extension is considered.

It is essential to understand the effects of transaction isolation and other transaction attributes before attempting to implement explicit locking in your application.

Syntax Pattern

```
SELECT ... FROM <sometable>
  [WHERE ...]
  [FOR UPDATE [OF ...]]
  [WITH LOCK]
...;
```

If the WITH LOCK clause succeeds, it will secure a lock on the selected rows and prevent any other transaction from obtaining write access to any of those rows, or their dependants, until your transaction ends.

If the FOR UPDATE clause is included, the lock will be applied to each row, one by one, as it is fetched into the server-side row cache. It becomes possible, then, that a lock which appeared to suc-

ceed when requested will nevertheless *fail subsequently*, when an attempt is made to fetch a row which becomes locked by another transaction.

The SELECT... WITH LOCK construct is available in DSQL and PSQL.

Important

The SELECT... WITH LOCK construct can succeed only in a top-level, single-table SELECT statement. It is **not available**

- in a subquery specification
- for joined sets
- with the DISTINCT operator, a GROUP BY clause or any other aggregating operation
- with a view
- with the output of a selectable stored procedure
- with an external table

Understanding the WITH LOCK clause

As the engine considers, in turn, each record falling under an explicit lock statement, it returns either the record version that is the most currently committed, regardless of database state when the statement was submitted, or an exception.

Wait behaviour and conflict reporting depend on the transaction parameters specified in the TPB block:

Table 4.1. How TPB settings affect explicit locking

TPB mode	Behaviour
isc_tpb_consistency	Explicit locks are overridden by implicit or explicit table-level locks and are ignored
isc_tpb_concurrency + isc_tpb_nowait	If a record is modified by any transaction that was committed since the transaction attempting to get explicit lock started, or an active transaction has performed a modification of this record, an update conflict exception is raised immediately
isc_tpb_concurrency + isc_tpb_wait	If the record is modified by any transaction that has committed since the transaction attempting to get explicit lock started, an update conflict exception is raised immediately.

TPB mode	Behaviour
	If an active transaction is holding ownership on this record (via explicit locking or by a normal optimistic write-lock) the transaction attempting the explicit lock waits for the outcome of the blocking transaction and, when it finishes, attempts to get the lock on the record again. This means that, if the blocking transaction committed a modified version of this record, an update conflict exception will be raised.
isc_tpb_read_committed + isc_tpb_nowait	If there is an active transaction holding ownership on this record (via explicit locking or normal update), an update conflict exception is raised immediately.
isc_tpb_read_committed + isc_tpb_wait	If there is an active transaction holding ownership on this record (via explicit locking or by a normal optimistic write-lock), the transaction attempting the explicit lock waits for the outcome of blocking transaction and when it finishes, attempts to get the lock on the record again. Update conflict exceptions can never be raised by an explicit lock statement in this TPB mode.

How the engine deals with **WITH LOCK**

When an UPDATE statement tries to access a record that is locked by another transaction, it either raises an update conflict exception or waits for the locking transaction to finish, depending on TPB mode. Engine behaviour here is the same as if this record had already been modified by the locking transaction.

No special gdscores are returned from conflicts involving pessimistic locks.

The engine guarantees that all records returned by an explicit lock statement are actually locked and DO meet the search conditions specified in WHERE clause, as long as the search conditions do not depend on any other tables, via joins, subqueries, etc. It also guarantees that rows not meeting the search conditions will not be locked by the statement. It can NOT guarantee that there are no rows which, though meeting the search conditions, are not locked.

Note

This situation can arise if other, parallel transactions commit their changes during the course of the locking statement's execution.

The engine locks rows at fetch time. This has important consequences if you lock several rows at once. Many access methods for Firebird databases default to fetching output in packets of a few hundred rows ("buffered fetches"). Most data access components cannot bring you the rows contained in the last-fetched packet, where an error occurred.

The optional **OF <column-names> sub-clause**

The FOR UPDATE clause provides a technique to prevent usage of buffered fetches, optionally with

the OF <column-names> to enable positioned updates.

Tip

Alternatively, it may be possible in your access components to set the size of the fetch buffer to 1. This would enable you to process the currently-locked row before the next is fetched and locked, or to handle errors without rolling back your transaction.

Caveats using WITH LOCK

- Rolling back of an implicit or explicit savepoint releases record locks that were taken under that savepoint, but it doesn't notify waiting transactions. Applications should not depend on this behaviour as it may get changed in the future.
- While explicit locks can be used to prevent and/or handle unusual update conflict errors, the volume of deadlock errors will grow unless you design your locking strategy carefully and control it rigorously.
- Most applications do not need explicit locks at all. The main purposes of explicit locks are (1) to prevent expensive handling of update conflict errors in heavily loaded applications and (2) to maintain integrity of objects mapped to a relational database in a clustered environment. If your use of explicit locking doesn't fall in one of these two categories, then it's the wrong way to do the task in Firebird.
- Explicit locking is an advanced feature, do not misuse it ! While solutions for these kinds of problems may be very important for web sites handling thousands of concurrent writers, or for ERP/CRM systems operating in large corporations, most application programs do not need to work in such conditions.

Examples using Explicit Locking

i) (simple)

```
SELECT * FROM DOCUMENT WHERE ID=? WITH LOCK
```

ii) (multiple rows, one-by-one processing with DSQL cursor)

```
SELECT * FROM DOCUMENT WHERE PARENT_ID=?  
FOR UPDATE WITH LOCK
```

(1.5) Improved Aggregate Handling

Arno Brinkman

Originally, grouped sets could be grouped only on named columns. In Firebird 1.0, it became possible

to group by a UDF expression. In 1.5, several further extensions to the handling of aggregate functions and the GROUP BY clause now allow groupings to be made by the degree of columns in the output specification (their 1-based "ordinal left-to-right position", as in the ORDER BY clause) or by a variety of expressions.

Caution

Not all expressions are currently allowed inside the GROUP BY list. For example, concatenation is not allowed.

Group By syntax

```
SELECT ... FROM .... [GROUP BY group_by_list]

group_by_list : group_by_item [, group_by_list];

group_by_item : column_name
               | degree (ordinal)
               | udf
               | group_by_function;

group_by_function : numeric_value_function
                  | string_value_function
                  | case_expression
                  ;

numeric_value_function : EXTRACT '(' timestamp_part FROM value ')';

string_value_function : SUBSTRING '(' value FROM pos_short_integer ')'
                       | SUBSTRING '(' value FROM pos_short_integer FOR nonneg_short_integer ')'
                       | KW_UPPER '(' value ')'
```

Important

The group_by_item cannot be a reference to any aggregate-function (including any that are buried inside an expression) from the same context.

HAVING

The having clause only allows aggregate functions or valid expressions that are part of the GROUP BY clause. Previously it was allowed to use columns that were not part of the GROUP BY clause and to use non-valid expressions.

ORDER BY

When the context is an aggregate statement, the ORDER BY clause only allows valid expressions that are aggregate functions or expression parts of the GROUP BY clause.

Previously it was allowed to use non-valid expressions.

Aggregate functions inside subqueries

It is now possible to use an aggregate function or expression contained in the GROUP BY clause inside a subquery.

Examples

```
SELECT
  r.RDB$RELATION_NAME,
  MAX(r.RDB$FIELD_POSITION),
  (SELECT
    r2.RDB$FIELD_NAME
  FROM
    RDB$RELATION_FIELDS r2
  WHERE
    r2.RDB$RELATION_NAME = r.RDB$RELATION_NAME and
    r2.RDB$FIELD_POSITION = MAX(r.RDB$FIELD_POSITION))
FROM
  RDB$RELATION_FIELDS r
GROUP BY
  1
/* ***** */
SELECT
  rf.RDB$RELATION_NAME AS "Relationname",
  (SELECT
    r.RDB$RELATION_ID
  FROM
    RDB$RELATIONS r
  WHERE
    r.RDB$RELATION_NAME = rf.RDB$RELATION_NAME)
  AS "ID",
  COUNT(*) AS "Fields"
FROM
  RDB$RELATION_FIELDS rf
GROUP BY
  rf.RDB$RELATION_NAME
```

Mixing aggregate functions from different contexts

Aggregate functions from different contexts can be used inside an expression.

Examples

```
SELECT
  r.RDB$RELATION_NAME,
  MAX(i.RDB$STATISTICS) AS "Max1",
  (SELECT
    COUNT(*) || ' - ' || MAX(i.RDB$STATISTICS)
  FROM RDB$RELATION_FIELDS rf
  WHERE
    rf.RDB$RELATION_NAME = r.RDB$RELATION_NAME) AS "Max2"
FROM
  RDB$RELATIONS r
  JOIN RDB$INDICES i on (i.RDB$RELATION_NAME = r.RDB$RELATION_NAME)
GROUP BY
  r.RDB$RELATION_NAME
HAVING
  MIN(i.RDB$STATISTICS) <> MAX(i.RDB$STATISTICS)
```

Note! This query gives results in FB1.0, but they are WRONG!

Subqueries are supported inside an aggregate function

Using a singleton SELECT expression inside an aggregate function is supported.

Example

```
SELECT
  r.RDB$RELATION_NAME,
  SUM((SELECT
    COUNT(*)
  FROM
    RDB$RELATION_FIELDS rf
  WHERE
    rf.RDB$RELATION_NAME = r.RDB$RELATION_NAME))
FROM
  RDB$RELATIONS r
  JOIN RDB$INDICES i
    on (i.RDB$RELATION_NAME = r.RDB$RELATION_NAME)
GROUP BY
  r.RDB$RELATION_NAME
```

Nested aggregate functions

Using an aggregate function inside another aggregate function is possible if the inner aggregate function is from a lower context (see example above).

Grouping by degree (ordinal number)

Using the degree number of the output column in the GROUP BY clause 'copies' the expression from the select list (as does the ORDER BY clause). This means that, when a degree number refers to a subquery, the subquery is executed at least twice.

(1.5) ORDER BY clause can specify expressions and nulls placement

Nickolay Samofatov

Order by expression

The ORDER BY clause lets you specify any valid expressions to sort query results. If the expression consists of a single number, it is interpreted as column (degree) number, as previously.

Nulls placement

The ordering of nulls in the result set can be controlled using a *nulls placement* clause.

Results can be sorted so that nulls are placed either above (NULLS FIRST) or below (NULLS LAST) the sorted non-nulls.

Behaviour when nulls placement is unspecified is NULLS LAST.

Syntax Pattern

```
SELECT ... FROM .... [ORDER BY order_list]....;

order_list : order_item [, order_list];
order_item : <expression> [order_direction] [nulls_placement]
order_direction : ASC | DESC;
nulls_placement : NULLS FIRST | NULLS LAST;
```

Restrictions

- If NULLS FIRST is specified, no index will be used for sorting.
- The results of a sort based on values returned from a UDF or a stored procedure will be unpredictable if the values returned cannot be used to determine a logical sorting sequence.
- The number of procedure invocations from specifying a sort based on a UDF or stored procedure will be unpredictable, regardless of whether the ordering is specified by the expression itself or by an ordinal number representing an expression in the column-list specification.
- An ordering clause for sorting the output of a union query may use only ordinal (degree) numbers to refer to the ordering columns.

Examples*i)*

```
SELECT * FROM MSG
ORDER BY PROCESS_TIME DESC NULLS FIRST
```

ii)

```
SELECT FIRST 10 * FROM DOCUMENT
ORDER BY STRLEN(DESCRIPTION) DESC
```

iii)

```
SELECT DOC_NUMBER, DOC_DATE FROM PAYORDER
UNION ALL
SELECT DOC_NUMBER, DOC_DATA FROM BUDGORDER
ORDER BY 2 DESC NULLS LAST, 1 ASC NULLS FIRST
```

(1.5) SELECT FIRST 0.. is Now Valid

In Firebird 1.5, zero can be accepted as an argument for FIRST in the `SELECT FIRST m .. SKIP n` construction. An empty result set will be returned.

(1.0) SELECT FIRST m .. SKIP n

(from the v.1.0 release notes)

Syntax Pattern


```
SELECT [FIRST (<integer expr m>)] [SKIP (<integer expr n>)]
```

Retrieves the first *m* rows of the selected output set. The optional **SKIP** clause will cause the first *n* rows to be discarded and return an output set of *m* rows starting at *n* + 1. In the simplest form, *m* and *n* are integers but any Firebird expression that evaluates to an integer is valid.

Available in SQL and DSQL except where otherwise indicated.

Arguments *m* and *n*

Parentheses are required for expression arguments and are optional otherwise.

The arguments can also bind variables, e.g.

```
SKIP ? * FROM ATABLE
```

returns the remaining dataset after discarding the *n* rows at the top, where *n* is passed in the "?" variable.

```
SELECT FIRST ? COLUMNA, COLUMNB FROM ATABLE
```

returns the first *m* rows and discards the rest.

An identifier that evaluates to an integer may also be used in GDML, although not in SQL or DSQL.

FIRST and SKIP Elements

The **FIRST** clause is also optional, i.e. you can include **SKIP** in a statement without **FIRST** to get an output set that simply excludes the rows appointed to **SKIP**.

Example

```
SELECT SKIP (5+3*5) * FROM MYTABLE;

SELECT FIRST (4-2) SKIP ? * FROM MYTABLE;

SELECT FIRST 5 DISTINCT FIELD FROM MYTABLE;
```

Two Gotchas with SELECT FIRST

1. This:

```
delete from TAB1
  where PK1 in (select first 10 PK1 from TAB1);
```

will delete all of the rows in the table. Ouch! the sub-select is evaluating each 10 candidate rows for deletion, deleting them, slipping forward 10 more...ad infinitum, until there are no rows left. Beware!

2. Queries like:

```
...
WHERE F1 IN ( SELECT FIRST 5 F2 FROM TABLE2
              ORDER BY 1 DESC )
```

won't work as expected, because the optimization performed by the engine transforms the correlated WHERE...IN (SELECT...) predicate to a correlated EXISTS predicate. It's obvious that in this case FIRST N doesn't make any sense:

```
...
WHERE EXISTS (
  SELECT FIRST 5 TABLE2.F2 FROM TABLE2
  WHERE TABLE2.F2 = TABLE1.F1
  ORDER BY 1 DESC )
```

Other Firebird 1.0.x Features

The remaining items in this section are DML enhancements that were introduced in Firebird 1.0.x, described again for the reader's convenience.

(1.0) GROUP BY UDF

It is now possible to aggregate a SELECT by grouping on the output of a UDF. e.g.

```
select
  strlen(rtrim(rdb$relation_name)),
  count(*) from rdb$relations
group by strlen(rtrim(rdb$relation_name))
order by 2
```

A side-effect of the changes enabling grouping by UDFs is that, whereas previously you could not call built-in Firebird functions in GROUP BY, now, by creating a dummy UDF wrapper, you can do:

```
select count(*)
from rdb$relations r
group by bin_or((select count(rdb$field_name)
  from rdb$relation_fields f
  where f.rdb$relation_name = r.rdb$relation_name),1)
```

(1.0) SUBSTRING(<string expr> FROM <pos> [FOR <length>])

Internal function, available in SQL and DSQL, implementing the ANSI SQL SUBSTRING() function. It will return a stream consisting of the byte at <pos> and all subsequent bytes up to the end of the string. If the optional FOR <length> is specified, it will return the lesser of <length> bytes or the number of bytes up to the end of the input stream.

The first argument can be any expression, constant or identifier that evaluates to a string.

<pos> must evaluate to an integer. <pos> in the string starts at 1, like other SQL positional elements.

Neither <pos> nor <length> can be query parameters: they must evaluate to constants.

Because <pos> and <length> are byte positions, the identifier of the input string can be a binary blob, or a sub_type 1 text blob with an underlying one-byte-per-character charset. The function currently does not handle text blobs with Chinese (2 byte/char maximum) or Unicode (3 byte/char maximum)

character sets.

For a string argument (as opposed to a blob), the function will tackle ANY charset.

Example

```
UPDATE ATABLE
SET COLUMNB = SUBSTRING(COLUMNB FROM 4 FOR 99)
WHERE ...
```

Please refer also to the later section on External Functions (UDFs) for details of changes and additions to external substring functions in the standard UDF library.

Stored Procedure and Trigger Language (PSQL)

The following enhancements have been made to PSQL, the set of language extensions available for writing stored procedures and triggers.

(1.5) EXECUTE STATEMENT

Alex Peshkov

EXECUTE STATEMENT "string" is a PSQL extension which takes a string that is a valid dynamic SQL statement and executes it as if it had been submitted to DSQL.

Available in triggers and stored procedures.

Syntax Patterns

The syntax may have three forms.-

Syntax 1

Executes "string" as an SQL operation that does not return any data rows, viz. INSERT, UPDATE, DELETE, EXECUTE PROCEDURE or any DDL statement except CREATE/DROP DATABASE.

```
EXECUTE STATEMENT <string>;
```

Example

```
CREATE PROCEDURE DynamicSampleOne (Pname VARCHAR(100))
AS
DECLARE VARIABLE Sql VARCHAR(1024);
DECLARE VARIABLE Par INT;
BEGIN
    SELECT MIN(SomeField) FROM SomeTable INTO :Par;
    Sql = 'EXECUTE PROCEDURE ' || Pname || '(';
    Sql = Sql || CAST(Par AS VARCHAR(20)) || ')';
    EXECUTE STATEMENT Sql;
END
```

Syntax 2

Executes "string" as an SQL operation, returning single data row. Only singleton SELECT operators may be executed with this form of EXECUTE STATEMENT.

```
EXECUTE STATEMENT <string> INTO :var1, [&, :varn] ;
```

Example

```
CREATE PROCEDURE DynamicSampleTwo (TableName VARCHAR(100))
AS
DECLARE VARIABLE Par INT;
BEGIN
    EXECUTE STATEMENT
        'SELECT MAX(CheckField) FROM ' || TableName INTO :Par;
    IF (Par > 100) THEN
        EXCEPTION Ex_Overflow 'Overflow in ' || TableName;
END
```

Syntax 3

Executes "string" as SQL operation, returning multiple data rows. Any SELECT operator may be executed with this form of EXECUTE STATEMENT.

```
FOR EXECUTE STATEMENT <string> INTO :var1, &, :varn
DO
    <compound-statement>;
```

Example

```
CREATE PROCEDURE DynamicSampleThree (
    TextField VARCHAR(100),
    TableName VARCHAR(100))
RETURNS (Line VARCHAR(32000))
AS
DECLARE VARIABLE OneLine VARCHAR(100);
BEGIN
    Line = '';
    FOR EXECUTE STATEMENT
        'SELECT ' || TextField || ' FROM ' || TableName
        INTO :OneLine
    DO
        IF (OneLine IS NOT NULL) THEN
            Line = Line || OneLine || ' ';
    SUSPEND;
END
```

Caveats with EXECUTE STATEMENT

The 'EXECUTE STATEMENT' DSQL string cannot contain any parameters in any syntax variation. All variable substitution into the static part of the SQL statement should be performed before the execution of EXECUTE STATEMENT.

This feature is intended only for very cautious use and should be used with all factors taken into account. It should be a rule of thumb to use EXECUTE STATEMENT only when other methods are im-

possible, or perform even worse than EXECUTE STATEMENT.

Caution

EXECUTE STATEMENT is potentially unsafe in several ways:

1. There is no way to validate the syntax of the enclosed statement.
2. There are no dependency checks to discover whether tables or columns have been dropped.
3. Operations will be slow because the embedded statement has to be prepared every time it is executed.
4. Return values are strictly checked for data type in order to avoid unpredictable type-casting exceptions. For example, the string '1234' would convert to an integer, 1234, but 'abc' would give a conversion error.
5. If the stored procedure has special privileges on some objects, the dynamic statement submitted in the EXECUTE STATEMENT string does not inherit them. Privileges are restricted to those granted to the user who is executing the procedure.

(1.5) New Context Variables

Dmitry Yemanov

A number of new context variables for PSQL have been implemented.

CURRENT_CONNECTION and CURRENT_TRANSACTION

These context variables return the system identifier of the *active connection* or the *current transaction* context, respectively. Return type is INTEGER. Available in DSQL and PSQL.

Important

Because these values are stored on the database header page, they will be reset after a database restore.

Syntax Patterns

```
CURRENT_CONNECTION  
CURRENT_TRANSACTION
```

Examples

```
SELECT CURRENT_CONNECTION FROM RDB$DATABASE;  
  
NEW.TXN_ID = CURRENT_TRANSACTION;
```

```
EXECUTE PROCEDURE P_LOGIN(CURRENT_CONNECTION) ;
```

ROW_COUNT

Returns an integer, the number of rows affected by the last DML statement. Available in PSQL, in the context of the procedure or trigger module.

Currently returns zero from a SELECT statement.

Syntax Pattern

```
ROW_COUNT
```

Examples

```
UPDATE TABLE1 SET FIELD1 = 0 WHERE ID = :ID;
IF (ROW_COUNT = 0) THEN
    INSERT INTO TABLE1 (ID, FIELD1) VALUES (:ID, 0);
```

Note

ROW_COUNT cannot be used for checking the rows affected by an EXECUTE STATEMENT command.

SQLCODE and GDSCODE

Each context variable returns an integer which is the numeric error code for the active exception. Available in PSQL, within the scope of the particular exception handling block. Both will evaluate to zero outside the block.

The GDSCODE variable returns a numeric representation of the GDS (ISC) error code, e.g. '335544349L' will return 335544349.

A 'WHEN SQLCODE' or 'WHEN ANY' exception block will catch a non-zero value for the SQLCODE variable and return zero for GDSCODE. Only a 'WHEN GDSCODE' block can catch a non-zero GDSCODE variable (and will return zero in SQLCODE).

If a user-defined exception is thrown, both SQLCODE and GDSCODE variables contain zero, regardless of the exception handling block type.

Syntax Pattern

```
SQLCODE
GDSCODE
```

Example

```
BEGIN
```

```
...
WHEN SQLCODE -802 DO
    EXCEPTION E_EXCEPTION_1;
WHEN SQLCODE -803 DO
    EXCEPTION E_EXCEPTION_2;
WHEN ANY DO
    EXECUTE PROCEDURE P_ANY_EXCEPTION(SQLCODE);
END
```

See also the EXCEPTION HANDLING ENHANCEMENTS, below, and the document README.exception_handling in the firebird2/doc/sql.extensions branch of the Firebird CVS tree.

INSERTING, UPDATING and DELETING

Three pseudo-Boolean expressions that can be tested to determine the type of DML operation being executed. Available in PSQL, only in triggers. Intended for use with multi-action triggers (see the DML section, above).

Syntax Pattern

```
INSERTING
UPDATING
DELETING
```

Example

```
IF (INSERTING OR UPDATING) THEN
BEGIN
    IF(NEW.SERIAL_NUM IS NULL) THEN
        NEW.SERIAL_NUM = GEN_ID(G_GENERATOR_1, 1);
```

(1.5) Enhancements to Exception Handling in PSQL

Dmitry Yemanov

The common syntax for an EXCEPTION statement in PSQL is:

```
EXCEPTION [name [value]];
```

The enhancements in 1.5 allow you to

1. define a run-time message for a named exception
2. re-initiate (re-raise) a caught exception within the scope of the exception block
3. Obtain a numeric error code for a caught exception

1) Run-time exception messaging

Syntax Pattern

```
EXCEPTION <exception_name> <message_value>;
```

Examples

a)

```
EXCEPTION E_EXCEPTION_1 'Error!';
```

b)

```
EXCEPTION  
  E_EXCEPTION_2 'Wrong type for record with ID=' || new.ID;
```

2) Re-raising an exception

Note

This has no effect outside an exception block.

Syntax Pattern

```
EXCEPTION;
```

Examples

a)

```
BEGIN  
  ...  
  WHEN SQLCODE -802 DO  
    EXCEPTION E_ARITH_EXCEPT;  
  WHEN SQLCODE -802 DO  
    EXCEPTION E_KEY_VIOLATION;  
  WHEN ANY THEN  
    EXCEPTION;  
END
```

b)

```
WHEN ANY DO  
BEGIN  
  INSERT INTO ERROR_LOG (...) VALUES (SQLCODE, ...);  
  EXCEPTION;  
END
```


3) Run-time error codes

See SQLCODE / GDSCODE (above).

(1.5) LEAVE / BREAK statement

Terminates the flow in a loop, causing flow of control to move to the statement following the END statement that completes that loop. Available for WHILE, FOR SELECT and FOR EXECUTE language constructs only, otherwise a parser error will be thrown. Available in triggers as well as stored procedures.

Important

The SQL-99 standard keyword LEAVE deprecates the existing use of BREAK.

Syntax Pattern

```
LEAVE ;
```

Examples

i)

```
BEGIN
  <statements>;
  IF (<conditions>) THEN
    LEAVE;
  <statements>;
END
```

ii)

```
WHILE (<condition>) DO
  BEGIN
    <statements>;
    WHEN ... DO
      LEAVE;
  END
```

The condition that branches to a LEAVE statement must be inside a block that is controlled by a looping construct (i.e., WHILE or FOR SELECT...INTO...DO).

Important

It is emphasised that LEAVE will not terminate other types of BEGIN...END block.

Note

LEAVE | BREAK and EXIT statements can now be used in triggers.

(1.5) Valid `PLAN` statements can now be included in triggers

Ignacio J. Ortega

Until now, a trigger containing a `PLAN` statement would be rejected by the compiler. Now, a valid plan can be included and will be used.

(1.5) Empty `BEGIN..END` blocks

Dmitry Yemanov

Empty `BEGIN..END` blocks in PSQL modules are now legal. For example, you can now write "stub" modules like

```
CREATE TRIGGER BI_atable FOR atable
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
END ^
```

(1.5) Declare and define local variable in single statement

Claudio Valderrama

Simplifies syntax and allows local variables to be declared and defined (or initialized) in one statement.

Syntax Pattern

```
DECLARE [VARIABLE] name <variable_type> [{ '=' | DEFAULT } value];
```

Example

```
DECLARE my_var INTEGER = 123;
```

New Reserved Words

The following new Firebird keywords should be added to the list of reserved words published for InterBase 6.0.1.

BIGINT (1.5)	CASE (1.5)	CURRENT_CONNECTION (1.5)
CURRENT_ROLE	CURRENT_TRANSACTION (1.5)	CURRENT_USER
RECREATE	ROW_COUNT (1.5)	RELEASE
SAVEPOINT		

The following keywords are reserved for future planned use:

ABS	BOOLEAN	BOTH
CHAR_LENGTH	CHARACTER_LENGTH	FALSE
LEADING	OCTET_LENGTH	TRIM
TRAILING	TRUE	UNKNOWN

The following keywords were reserved words in Firebird 1.0 and are no longer reserved in Firebird 1.5:

BREAK	DESCRIPTOR	FIRST
IIF	SKIP	SUBSTRING

The following non-reserved words are recognised in 1.5 as keywords when used in their respective structural contexts:

COALESCE	DELETING	INSERTING
LAST	LEAVE	LOCK
NULLIF	NULLS	STATEMENT
UPDATING	USING	

The following new InterBase 6.5 and 7 keywords (not reserved in Firebird) should also be treated as if they were reserved, for compatibility:

BOOLEAN	FALSE	GLOBAL
PERCENT	PRESERVE	ROWS
TEMPORARY	TIES	TRUE

Miscellaneous Enhancements

This chapter contains a number of enhancements to utilities, character sets and external functions (UDFs).

Character Sets

New character sets, collations and changes in this release are listed below. V. 1.0 additions are included for the reader's convenience.

V. 1.5

Various Contributors

WIN1251_UA collation

Added WIN1251_UA collation (for both Russian and Ukrainian languages) for WIN1251 character set.

WIN1251 Uppercasing

Corrected default uppcasing for WIN1251

Hungarian Collation

Added ISO_HUN (for Hungarian language) for ISO8859_2 character set

V. 1.5 (Binary Collations only)

Blas Rodrigues Somoza

DOS737 PC

Greek

DOS775 PC

Baltic

DOS858

Variant of Cp850 with Euro character (€)

DOS862

PC Hebrew

DOS864

PC Arabic

DOS866

MS-DOS Russian

DOS869

IBM Modern Greek

WIN1255

Windows Hebrew

WIN1256

Windows Arabic

WIN1257

Windows Baltic

ISO8859_3

Latin 3 (Esperanto, Maltese, Pinyin, Sami, Croatian and others)

ISO8859_4

Latin 4 (Baltic, Greenlandic, Lappish)

ISO8859_5

Cyrillic

ISO8859_6

Arabic

ISO8859_7

Greek

ISO8859_8

Hebrew

ISO8859_9

Turkish

ISO8859_13

Baltic

V. 1.0.x

Various Contributors

Hungarian

Added case insensitive Hungarian collation set (*Sandor Szollosi*)

ISO8859-2

Firebird now supports character set ISO8859-2 (for Czech language).

New ISQL Features

New in ISQL:

"readline" Capability in the ISQL Shell

Mark O'Donohue

Command history support (like Unix readline) has been added to the isql shell. Now you can use the Up and Down arrow keys to step back or forward through the commands submitted in the isql session.

External Functions (UDFs)

User-defined functions.-

In the ib_udf Library

rpad()

Juan Guerrero

```
rpad (instring, length, padcharacter)
```

Right-pads the supplied string *instring* by appending padcharacters until the result string has the given length.

The input string can be any length less than 32766 bytes. Length must not exceed 32765 bytes.

Declaration

```
DECLARE EXTERNAL FUNCTION rpad
  CSTRING(80), INTEGER, CSTRING(1)
  RETURNS CSTRING(80) FREE_IT
  ENTRY_POINT 'IB_UDF_rpad' MODULE_NAME 'ib_udf';
```

lpad()

Juan Guerrero

```
lpad (instring, length, padcharacter)
```

Left-pads the supplied string *instring* by prepending padcharacters until the result string has the given length.

The input string can be any length less than 32766 bytes. Length must not exceed 32765 bytes.

Declaration

```
DECLARE EXTERNAL FUNCTION lpad
  CSTRING(80), INTEGER, CSTRING(1)
  RETURNS CSTRING(80) FREE_IT
  ENTRY_POINT 'IB_UDF_lpad' MODULE_NAME 'ib_udf';
```

log()

Paul Vinkenoog

`log (x, y)`

This function had an old bug, whereby the arguments x and y were erroneously reversed. It should return the logarithm base x of y but in fact it returned the log base y of x. It has been corrected.

Warning

If it was used in your applications previously, PLEASE CHECK YOUR APPLICATION CODE! it either returned the wrong results; or someone, at some point, did a workaround and reversed the arguments deliberately in order to get the right calculation.

In the fbudf library

Changes in Firebird 1.5 affect functions in the *fbudf* external function library, as follows.-

1. The **NVL* and **NULLIF* functions remain for backward compatibility, but are deprecated by the introduction of the new internal functions *CASE*, *COALESCE* and *NULLIF*.
2. If you are porting a database that was created in Firebird 1.0.x and you declared the fbudf functions *truncate* and *round*, those declarations will no longer work with Firebird 1.5 because the *entry_point* names were changed. You will need to drop the functions and redeclare them, using the declarations from the *fbudf.sql* script in the 1.5 /UDF directory.

Caution

Note that fbudf cannot handle string fields bigger than (32Kb - 1) bytes in length. This limit may have ill effects where strings are concatenated before being passed to UDFs taking string arguments. If the sum of field is beyond the limit, the behavior is undefined. The function may return a non-sense result or the fbudf code may perform an illegal operation.

Chapter 6

New Configuration Files

In Firebird 1.5 there are two configuration files: *firebird.conf* and *aliases.conf*. The *firebird.conf* file replaces *ibconfig* (Windows) and *isc_config* (POSIX) that were in the Firebird 1.0.x distributions. *Aliases.conf* is new: it ties in with some database access parameters in *firebird.conf* to enable connection to databases without passing filesystem paths across the wire.

The Firebird Root Directory

The root directory of your Firebird installation is used in many ways, both during installation and as an attribute that server routines, configuration parameters and clients depend on. Because several ways exist to tell the server where to find a value for this attribute, developers and system administrators should be aware of the precedence trail that the server follows at startup, to determine it correctly.

Precedence Trail for Locating the Firebird Root

Win32 Superserver and Classic builds (both server and client):

1. FIREBIRD environment variable
2. *RootDirectory* parameter in *firebird.conf*
3. Registry:

HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances\DefaultInstance and looks for the field *DefaultInstance*.

4. The directory one level above the one where the server binary is located

Win32 Embedded:

1. FIREBIRD environment variable
2. *RootDirectory* parameter in *firebird.conf*
3. The directory where *fbembed.dll* (renamed *fbclient.dll*) is located

Linux Classic:

1. FIREBIRD environment variable
2. *RootDirectory* parameter in *firebird.conf*
3. Default installation path (*/opt/firebird*)

Linux Superserver:

1. FIREBIRD environment variable
2. *RootDirectory* parameter in *firebird.conf*
3. The directory one level above the one where the server binary is located (retrieved via symlink *"/proc/self/exe"*, if supported)
4. Default installation path (*/opt/firebird*)

Server Configuration File--*firebird.conf*

Default values are applicable to most parameters. Parameter names and values are case-sensitive on Linux but not on Windows. To set any parameter to a non-default setting, delete the comment (#) marker and edit the value. You can edit the configuration file while the server is running. To activate configuration changes, it is necessary to stop and restart the service.

Entries are in the form:

```
parameter_name value
```

- *parameter_name* is a string that contains no whitespace and names a property of the server being configured.
- *value* is a number, Boolean (1=True, 0=False) or string that specifies the value for the parameter

Parameters

Filesystem-related

RootDirectory

String, the absolute path to a directory root on the local filesystem. It should remain commented unless you want to force the startup procedure to override the path to the root directory of the Firebird server installation, that it would otherwise detect for itself.

DatabaseAccess

Supports the database-aliasing feature. In previous versions, the server could attach to any database in its local filesystem and was accessed by applications passing the file's absolute filesystem path. This parameter provides options to restrict the server's access to aliased databases only, or to only databases located in specific filesystem trees.

DatabaseAccess may be None, Restrict or Full.

Full (the default) permits database files to be accessed anywhere on the local filesystem.

None permits the server to attach only databases that are listed in aliases.conf.

Restrict allows you to configure the locations of attachable database files to a specified list of filesystem tree-roots. Supply a list of one or more tree-roots, separated by semi-colons, to define one or more permissible locations.

For example,

```
Unix: /db/databases;/userdir/data
Windows: D:\data
```

Relative paths are treated as relative to the path that the running server recognizes as the root directory. For example, on Windows, if the root directory is C:\Program Files\Firebird, then the following value will restrict the server to accessing database files only if they are located under C:\Program Files\Firebird\userdata:

```
DatabaseAccess = Restrict userdata
```

Note

Database shadowing - the current DatabaseAccess handling has a bug that means you must use the Restrict option if you are shadowing any database on the server.

ExternalFileAccess

Was external_file_directory in isc_config/ibconfig but syntax has changed.

Provides three levels of security regarding EXTERNAL FILES (fixed format text files that are to be accessed as database tables). The value is a string, which may be None, Full or Restrict.

None (the default value) disables any use of external files on your server.

Restrict provides the ability to restrict the location of external files for database access to specific path-trees. Supply a list of one or more tree-roots, separated by semi-colons (;), within and beneath which external files may be stored.

For example,

```
Unix: /db/extern;/mnt/extern
Windows: C:\ExternalTables
```

Relative paths are treated as relative to the path that the running server recognizes as the root directory of the Firebird installation. For example, on Windows, if the root that

the running server recognizes as the root directory of the Firebird installation is C:\Program Files\Firebird, then the following value will restrict the server to accessing external files only if they are located in C:\Program Files\Firebird\userdata\ExternalTables:

```
ExternalFileAccess = Restrict userdata\ExternalTables
```

Full permits external files to be accessed anywhere on the system.

Caution

See the CAUTION below the next entry, UdfAccess.

UdfAccess

Was as `external_function_directory` in `isc_config/ibconfig` but syntax has changed. It replaces not just the name of the earlier parameter, but also the form in which the values are presented. The purpose of the changes was to enable optional levels of protection for external user-defined library modules, a recognized target for malicious intruder attacks. UdfAccess may be None, Restrict or Full.

Restrict (the default setting) retains the functionality provided by the `external_function_directory` parameter in Firebird 1.0, to restrict the location of callable external libraries to specific filesystem locations. Supply a list of one or more tree-roots, separated by semi-colons (;), within and beneath which UDF, BLOB filter and character set definitions may be stored.

For example,

```
Unix: /db/extern:/mnt/extern
Windows: C:\ExternalModules
```

Relative paths are treated as relative to the path that the running server recognizes as the root directory of the Firebird installation. For example, on Windows, if the root of the Firebird installation is C:\Program Files\Firebird, then the following value will restrict the server to accessing external libraries only if they are located in C:\Program Files\Firebird\userdata\ExternalModules:

```
UDFAccess = Restrict userdata\ExternalModules
```

None disallows all use of user-defined external libraries.

Full permits external libraries to be accessed anywhere on the system.

Caution

Avoid setting up custom directory trees for UdfAccess and ExternalFileAccess such that they share a parent tree-root. The default settings are safe. If you are setting up your own and you don't make separated directory trees for them, the server can be easily hacked to execute unauthorised code. An example of what to avoid:

```
UdfAccess = UDF; /bad_dir
ExternalFileAccess = /external; /bad_dir/files
```

UdfAccess & ExternalFileAccess here have a common sub-tree, /bad_dir/files, where someone could place his external file /bad_dir/files/hackudf.so and execute his own code on the compromised system.

Resource-related

CpuAffinityMask

(was `cpu_affinity` in `isc_config/ibconfig`). With Firebird SuperServer on Windows, there is a problem with the operating system continually swapping the entire SuperServer process back and forth between processors on SMP machines. This ruins performance. This parameter can be used on SMP systems on Windows to set Firebird SuperServer's processor affinity to a single CPU.

Caution

Firebird Superservers, up to and including Release 1.5, may not support the Hyperthreading feature of some later-model motherboards on Windows. To avoid balancing problems, you may need to disable hyperthreading at system BIOS level.

CpuAffinityMask takes one integer, the CPU mask.

Example

```
CpuAffinityMask = 1
```

only runs on the first CPU (CPU 0).

```
CpuAffinityMask = 2
```

only runs on the second CPU (CPU 1).

```
CpuAffinityMask = 3
```

runs on both first and second CPU.

Calculating the affinity mask value

You can use this flag to set Firebird's affinity to any single processor or (on Classic server) any combination of the CPUs installed in the system.

Consider the CPUs as an array numbered from 0 to n-1, where n is the number of processors in-

stalled and i is the array number of a CPU. M is another array, containing the MaskValue of each selected CPU. The value A is the sum of the values in M .

Use the following formula to arrive at M and calculate the MaskValue A :

$$\begin{aligned} M_i &= 2^i \\ A &= M_1 + M_2 + M_3 \dots \end{aligned}$$

For example, to select the first and fourth processors (processor 0 and processor 3) calculate as follows:

$$A = 2^0 + 2^3 = 1 + 8 = 9$$

DeadlockTimeout

(was `deadlock_timeout` in `isc_config/ibconfig`). Number of seconds (integer) that the lock manager will wait after a conflict has been encountered, before purging locks from dead processes and doing a further deadlock scan cycle. Normally, the engine detects deadlocks instantly. The deadlock timeout kicks in only if something goes wrong.

The default of 10 seconds is about right for most conditions. Setting it lower does not necessarily improve the speed with which problem deadlocks return a conflict exception. If it is too low, the effect may be unnecessary extra scans which degrade system performance.

DefaultDbCachePages

(was `database_cache_pages` in `isc_config/ibconfig`). Server-wide default (integer) number of database pages to allocate in memory, per database. The configured value can be overridden at database level.

The default value for SuperServer is 2048 pages. For Classic, it is 75.

SuperServer and Classic use the cache differently. SS pools its cache for use by all connections; Classic allocates a static cache to each connection.

EventMemSize

Integer, representing number of bytes of memory reserved for the event manager. Default is 65536 (64 Kb).

LockAcquireSpins

(was `lock_acquire_spins`). Relevant only on SMP machines running Classic server. In Classic server, only one client process may access the lock table at any time. A mutex governs access to the lock table. Client processes may request the mutex conditionally or unconditionally. If it is conditional, the request fails and must be retried. If it is unconditional, the request will wait until it is satisfied. LockAcquireSpins establishes the number of attempts that will be made if the mutex request is conditional.

Integer. The default is 0 (unconditional). There is no recommended minimum or maximum.

LockHashSlots

(was `lock_hash_slots` in `isc_config/ibconfig`). Use this parameter for tuning the lock hash list. Under heavy load, throughput might be improved by raising the number of hash slots to disperse the list in shorter hash chains.

Integer-prime number values are recommended. The default is 101.

LockGrantOrder

(was `lock_grant_order` in `isc_config/ibconfig`). When a connection wants to lock an object, it gets a lock request block which specifies the object and the lock level requested. Each locked object has a lock block. Request blocks are connected to those lock blocks either as requests that have been granted, or as pending requests.

The `LockGrantOrder` parameter is a Boolean. The default (1=True) indicates that locks are to be granted on a first-come-first-served basis. The False setting (0), emulating InterBase v3.3 behavior, grants the lock as soon as it becomes available. It can result in lock requests being "starved".

LockMemSize

This integer parameter represents the number of bytes of shared memory allocated for the lock manager. For a Classic server, the `LockMemSize` gives the initial allocation, which will grow dynamically until memory is exhausted ("Lock manager is out of room"). If there are a lot of connections or big page caches, increase this parameter to avoid these errors.

In SuperServer, the memory allocated for the lock manager does not grow.

The default size on Linux and Solaris is 98304 bytes (96 Kb). On Windows, it is 262144 (256 Kb).

LockSemCount

Integer parameter, specifying the number of semaphores available for interprocess communication (IPC). The default is 32. Set this parameter in non-threading environments to raise or lower the number of available semaphores.

SortMemBlockSize

This parameter allows you to configure, in bytes, the size of each memory block used by the in-memory sorting module. The installation default is 1 Mb; you can reconfigure it to any size up to the currently configured maximum value set by the `SortMemUpperLimit` parameter (see below).

SortMemUpperLimit

The maximum amount of memory, in bytes, to be allocated by the in-memory sorting module. The installation default is 67108864 bytes (64 Mb) for SuperServer and 8388608 (8 Mb) for the Classic server.

Caution

For Classic, bear in mind that increasing either the block size or the maximum limit affects each client connection/server instance and will ramp up the server's memory consumption accordingly.

Communications-related

ConnectionTimeout

(was `connection_timeout` in `isc_config/ibconfig`). Number of seconds to wait before abandoning an attempt to connect. Default 180.

DummyPacketInterval

(was `dummy_packet_interval` in `isc_config/ibconfig`). This is the number of seconds (integer) the server should wait on a silent client connection before sending dummy packets to request acknowledgment.

Warning

DO NOT USE THIS OPTION on a Win32 server running TCP/IP clients. It causes a persistent increase in usage of kernel non-paged memory which may hang or crash Windows on the client side as explained at <http://support.microsoft.com/default.aspx?kbid=296265>.

Win32-with-TCP/IP apart, it's the only way to detect and disconnect inactive clients when either NamedPipes (NetBEUI), XNET or IPC protocols are used. There are no known issues on POSIX systems.

Normally, Firebird uses the *SO_KEEPA LIVE* socket option to keep track of active connections. If you do not like the default two-hour keepalive timeout, adjust your server OS settings appropriately:

- On UNIX-like OS's, modify the contents of `/proc/sys/net/ipv4/tcp_keepalive_*`.
- On Windows, follow instructions in [this article](#).

Default should be 0 - not 60 which was the old default in Firebird 1.0 and most of the 1.5 release candidates. A setting of 60 should be treated as the default on systems where you need to make use of this dummy packet polling.

RemoteServiceName

Default = `gds_db`. See *RemoteServicePort*, next.

RemoteServicePort

These two parameters provide the ability to override either the TCP/IP service name or the TCP/IP port number used to listen for client database connection requests, if one of them differs from the installed defaults (`gds_db/tcp 3050`).

Change one of the entries, not both. The *RemoteServiceName* is checked first for a matching entry in the services file. If there is a match, the port number configured for *RemoteServicePort* is used. If there is not a match, then the installation default, port 3050, is used.

Note

If a port number is provided in the TCP/IP connection string, it will always take precedence over *RemoteServicePort*.

RemoteAuxPort

The inherited InterBase behavior, of passing event notification messages back to the network layer through randomly selected TCP/IP ports, has been a persistent source of network errors and conflicts with firewalls, sometimes to the extent of causing the server to crash under some conditions. This parameter allows you to configure a single TCP Port for all event notification traffic.

The installation default (0) retains the traditional random port behaviour. To dedicate one specific port for event notifications, use an integer which is an available port number.

RemoteBindAddress

By default, clients may connect from any network interface through which the host server accepts traffic. This parameter allows you to bind the Firebird service to incoming requests through one NIC and to reject connection requests from any other network interfaces. This should help to overcome problems in some subnetworks where the server is handling traffic through multiple NICs.

String, in a valid dotted IP format. Default value (not bound) is no value.

TcpRemoteBufferSize

The engine reads ahead of the client and can send several rows of data in a single packet. The larger the packet size, the more rows are sent per transfer. Use this parameter-with caution and complete comprehension of its effects on network performance!-if you need to enlarge or reduce the TCP/IP packet size for send and receive buffers. It affects both the client and server.

Value is an integer (size of packet in bytes) in the range 1448 to 32768. The installation default is 8192.

POSIX-specific

LockSignal

Integer parameter, UNIX signal to use for interprocess communication. Default: 16

RemoteFileOpenAbility

Warning

USE ONLY WITH EXTREME CAUTION

Boolean parameter which, if set True, allows the engine to open database files which reside on a networked filesystem (NFS) mounted partition. Because the filesystem is beyond the control of the local system, this is a very risky feature that should not be enabled for the purpose of opening any read/write database whose survival matters to you.

The default is 0 (False, disabled) and you should leave it that way unless you are very clear about its effects.

TcpNoNagle

(was `tcp_no_nagle` in `isc_config/ibconfig`). On Linux, by default, the socket library will minimize physical writes by buffering writes before actually sending the data, using an internal algorithm (implemented as the `TCP_NODELAY` option of the socket connection) known as Nagle's Algorithm. It was designed to avoid problems with small packets, called *tinygrams*, on slow networks.

By default, `TCP_NODELAY` is enabled (value 0) when Firebird Superserver is installed on Linux. On slow networks, disabling it can actually improve speed.

Tip

Watch out for the double negative--set the parameter True to disable `TCP_NODELAY` and False to enable it.

In releases up to and including v.1.5, this feature is active only for Superserver.

Windows-specific

CreateInternalWindow

The "Windows local" protocol uses a hidden window for inter-process communication between the local client and the server. This IPC window is created at server startup when `CreateInternalWindow` is true (1, the default). Set it to 0 (off) to run the server without a window and thus to disable local protocol.

With local protocol disabled, it is possible to run multiple instances of the server simultaneously.

DeadThreadsCollection

A setting for the thread scheduler on Windows, this integer parameter establishes the number of priority switching cycles (see *PrioritySwitchDelay*, below) that the scheduler is to execute before a thread is destroyed (or closed).

Immediate destruction (or closure) of worker threads would require a semaphore and blocking call, generating significant overhead. Instead, a thread scheduler maintains threads in a pool. When a thread has completed its task, it is marked as idle. The idle thread is destroyed (or closed) after *n* iterations of the scheduler loop, where *n* is the value of the *DeadThreadsCollection* parameter.

For a server handling a very large number of connections--in the high hundreds or more--the parameter value will need to be raised from its default setting of 50.

GuardianOption

Boolean parameter used on Windows servers to determine whether the Guardian should restart the server every time it terminates abnormally. The installation default is to do so (1=True). To disable the restart behavior, set this parameter off (0=False).

IpcMapSize

(was *server_client_mapping* in *ibconfig*). Size in bytes of one client's portion of the memory-mapped file used for interprocess communication (IPC) in the connection model used for "Windows local" connection. It has no equivalent on other platforms.

Integer, from 1024 to 8192. The default is 4096.

Increasing the map size may improve performance when retrieving very wide or large data row sets, such as those returning graphics BLOBs.

Note

NOTE This can no longer be modified in the Guardian's system tray icon dialog.

IpcName

Default value: *FirebirdIPI*

The name of the shared memory area used as a transport channel in local protocol.

Important

The Release 1.5 default value--*FirebirdIPI*--is not compatible with older releases of Firebird nor with *InterBase®*. Use the value *InterBaseIPI* to restore compatibility, if necessary.

MaxUnflushedWrites

This parameter was introduced in Version 1.5 to handle a bug in the Windows server operating systems, whereby asynchronous writes were never flushed to disk except when the Firebird server underwent a controlled shutdown. (Asynchronous writes are not supported in Windows 9x or ME.) Hence, on 24/7 systems, asynchronous writes were never flushed at all.

This parameter determines how frequently the withheld pages are flushed to disk when Forced Writes are disabled (asynchronous writing is enabled). Its value is an integer which sets the number of pages to be withheld before a flush is flagged to be done next time a transaction commits.

Default is 100 in Windows installations and -1 (disabled) in installations for all other platforms.

If the end of the *MaxUnflushedWriteTime* cycle (see below) is reached before the count of with-

held pages reaches the `MaxUnflushedWrites` count, the flush is flagged immediately and the count of withheld pages is reset to zero.

MaxUnflushedWriteTime

This parameter determines the maximum length of time that pages withheld for asynchronous writing are flushed to disk when Forced Writes are disabled (asynchronous writing is enabled). Its value is an integer which sets the interval, in seconds, between the last flush to disk and the setting of a flag to perform a flush next time a transaction commits. Default is 5 seconds in Windows installations and -1 (disabled) in installations for all other platforms.

PrioritySwitchDelay

A setting for the thread scheduler on Windows, this integer establishes the time, in milliseconds, which is to elapse before the priority of an inactive thread is reduced to LOW or the priority of an active thread is advanced to HIGH. One iteration of this switching sequence represents one thread scheduler cycle.

The default value is 100 ms, chosen on the basis of experiments on Intel PIII/P4 processors. For processors with lower clock speeds, a longer delay will be required.

PriorityBoost

Integer, sets the number of extra cycles given to a thread when its priority is switched to HIGH. The installation default is 5.

ProcessPriorityLevel

(was `server_priority_class` in `ibconfig`). Priority level/class for the server process. This parameter replaces the `server_priority_class` parameter of pre-1.5 releases-see below-with a new implementation.

The values are integer, as follows:

- 0 - normal priority
- positive value - high priority (same as `-B[oostPriority]` switch on `instsvc.exe` configure and start options)
- negative value - low priority

Note

All changes to this value should be carefully tested to ensure that they actually cause the engine to be more responsive to requests.

RemotePipeName

Applicable only for NetBEUI connections.

String parameter, the name of the pipe used as a transport channel in NetBEUI protocol. The named pipe is equivalent to a port number for TCP/IP. The default value--*interbas*-- is compatible with older releases of Firebird and with InterBase®.

Related to Sort space

When the size of the internal sort buffer is too small to accommodate the rows involved in a sort operation, Firebird needs to create temporary sort files on the server's filesystem. By default, it will look for the path specified in the environment variable `FIREBIRD_TMP`. If that variable is not present, it will try to use the root of the `/tmp` filesystem on Linux/UNIX, or `C:\temp` on Windows NT/2000/XP.

None of these locations can be configured for size.

Firebird provides a parameter for configuring the disk space that will be used for storing these temporary files. It is prudent to use it, to ensure that sufficient sort space will be available under all conditions.

All CONNECT or CREATE DATABASE requests share the same list of temporary file directories and each creates its own temporary files. Sort files are released when the sort is finished or the request is released.

Note

In Release 1.5, the name of the parameter changed from tmp_directory to TempDirectories and the syntax of the parameter value also changed.

TempDirectories

(replaces tmp_directory entries in isc_config/ibconfig). Supply a list of one or more directories, separated by semi-colons (;), under which sort files may be stored. Each item may include an optional size argument, in bytes, to limit its storage. If the argument is omitted, or is invalid, Firebird will use the space in that directory until it is exhausted, before moving on to the next listed directory.

For example,

```
Unix: /db/sortfiles1 1000000000;/firebird/sortfiles2
Windows: E:\sortfiles 500000000
```

Relative paths are treated as relative to the path that the running server recognizes as the root directory of the Firebird installation. For example, on Windows, if the root directory is C:\Program Files\Firebird, then the following value will tell the server to store temporary files in C:\Program Files\Firebird\userdata\sortfiles, up to a limit of 500 Mb:

```
TempDirectories = userdata\sortfiles 500000000
```

Note

No quoted paths (as was required in Firebird 1.0).

Compatibility

CompleteBooleanEvaluation

Establishes the Boolean evaluation method (complete or shortcut). The default (0=False) is to "short-cut" a Boolean evaluation expression involving the AND or OR predicates by returning as soon as a result of True or False is obtained that cannot be affected by the results of any further evaluation.

Under very rare (usually avoidable) conditions, it might happen that an operation inside an OR or an AND condition that remains unevaluated due to the shortcut behavior has the potential to affect the outcome of the original result. If you have the misfortune to inherit an application that has such characteristics in its SQL logic, you might wish to use this parameter to force complete evaluation until you have the opportunity to perform surgery on it.

Parameter type is Boolean.

Important

Don't overlook the fact that this flag affects all Boolean evaluations performed in any databases on the server.

OldParameterOrdering

Version 1.5 addressed and fixed an old InterBase bug that caused output parameters to be returned to the client with an idiosyncratic ordering in the XSQLDA structure. The bug was of such longevity that many existing applications, drivers and interface components have built-in work-arounds to correct the problem on the client side.

Releases 1.5 and later reflect the corrected condition in the API and are installed with `OldParameterOrdering=0` (False). Set this Boolean parameter True if you need to revert to the old condition for compatibility with existing code.

Important

Don't overlook the fact that this setting affects all databases on the server and will potentially produce exceptions or wrong results if set on in an environment where applications are not compensating for the expected bug.

OldColumnNaming

New parameter added at v.1.5.3

This parameter allow users to revert to pre-V1.5 column naming behaviour in SELECT expressions. If this parameter changed from its default 0 setting, the engine will not attempt to supply run-time identifiers, e.g. `CONCATENATION` for derived fields where the developer has neglected to provide identifiers.

Important

This setting affects all databases on the server and will potentially produce exceptions or unpredictable results where mixed applications are implemented.

Database File Aliasing

Firebird release 1.5 introduced database file aliasing to improve the portability of applications and to tighten up control of both internal and external database file access.

Aliases.conf

Configure database file aliases in the text file *aliases.conf*, located in the root directory of your Firebird server installation. The installed *aliases.conf* looks similar to this:

```
#
# List of known database aliases
# -----
#
# Examples:
```

```
#  
# dummy = c:\data\dummy.fdb  
#
```

As in all of Firebird's configuration files, the '#' symbols are comment markers. To configure an alias, simply delete the '#' and change the dummy line to the appropriate database path:

```
# fbdb1 is on a Windows server:  
fbdb1 = c:\Firebird\sample\Employee.fdb  
# fbdb2 is on a Linux server  
fbdb2 = /opt/databases/killergames.fdb  
#
```

You can edit aliases.conf whilst the server is running. There is no need to stop and restart the server in order for new aliases.conf entries to be recognised.

Connecting using an aliased path

The modified connection string in your client application looks like this:

```
Server_name:aliasname
```

With the example above, the following connection string will ask the Firebird server running on a Linux box named "myserver" to find and connect the client to the database at the path identified in aliases.conf as "fbdb2":

```
myserver:fbdb2
```

Note

Because the gstat tool does not use a database connection to read the database file, a full path is still required for using gstat. (This may change).

Chapter 7

Firebird 1.5 Project Teams

Table 7.1. Firebird Development Teams

Developer	Country	Major Tasks
Dmitry Yemanov	Russian Federation	Release coordinator; DSQL and PSQL enhancements; implementor of Embedded Server, numerous metadata enhancements, database aliasing, multi-action triggers, BigInt data type, new context variables, Windows Classic server; numerous bug-fixes
Nickolay Samofatov	Russian Federation	SQL feature designer/implementor (Savepoints, pessimistic locking); metadata enhancements; major engine re-implementations; bug-finder and fixer; architectural troubleshooter; enabled Services API on Linux Classic; performance enhancements; Linux Classic builds
Arno Brinkman	The Netherlands	Optimizer enhancements; many new DSQL features
Claudio Valderrama	Chile	Code scrutineer; bug-finder and fixer; PSQL enhancements; UDF fixer, designer and implementor
Alex Peshkoff	Russian Federation	New PSQL and DSQL features; security features coordinator and author; code fixer; Linux Superserver builds
Mike Nordell	Sweden	Translated Firebird codebase to C++; performance enhancements; feature porting; bug-finder and fixer
Blas Rodriguez Somoza	Spain	Developer of new character sets; major code cleaner and tree surgeon; MinGW builds
Roman Rokytssky	Germany	Jaybird implementor and co-coordinator
David Jencks	U.S.A.	JayBird designer and co-coordinator; designer of Firebird documentation tools
Carlos Guzman Alvarez	Spain	Developer and coordinator of .NET provider for Firebird
John Bellardo	U.S.A.	Implemented plug-in interface for character sets; co-

Developer	Country	Major Tasks
		ordinator, Darwin builds; initial implementor of new memory model
Erik Kunze	Germany	Bug-finder and fixer; code-cleaner; SINIX-Z builds
Dmitry Sibiriyakov	Russian Federation	Code cleanup; MinGW builds
Pavel Cisar	Czech Republic	Linux builds (Release 1.0); QA tools designer/co-ordinator
Ann Harrison	U.S.A.	Bug-fixer; technical advisor; increased maximum indexes allowed
Mark O'Donohue	Australia	readline feature in isql; bug-fixing; boot builds (Release 1.0); code-fixing (Release 1.0)
Paul Reeves	France	QA; Win32 installers; standard Win32 control panel applet
Ignacio J. Ortega	Spain	Added PLAN feature to triggers; code cleanup
Konstantin Kuznetsov	Russian Federation	Solaris Intel builds
Olivier Mascia	Belgium	Re-implementor of Win32 installation services
Peter Jacobi	Germany	Improvements, updating of character sets
Tilo Muetze	Germany	Firebird documentation project coordinator
Paul Vinkenoog	The Netherlands	Coordinator, Firebird documentation project; UDF fixes
Artur Anjos	Portugal	Enhanced Win32 control panel applets; Firebird Configuration Manager developer; internationalization of same
Achim Kalwa	Germany	Enhanced Win32 control panel applets
Sean Leyne	Canada	Bugtracker organizer; code cleaner
Ryan Baldwin	U.K.	Jaybird Type 2 driver developer
Sandor Szollosi	Hungary	Implementor of character set collations

Developer	Country	Major Tasks
Dmitry Kuzmenko	Russian Federation	Bugfixed GSTAT
Artem Petkevych	Ukraine	Bug-fixed ARRAY type
Vlad Horsun	Ukraine	Speed-boosted sweep; fixed 2-phase commit bug
Tomas Skoda	Slovakia	Bug-fixer
Evgeny Kilin	Russian Federation	Bug-fixer
Oleg Loa	Russian Federation	Bug-fixer
Erik S. La Bianca	U.S.A.	Bug-fixer
Tony Caduto	U.S.A.	Unofficial Win32 installers
Juan Guerrero	Spain	New UDFs
Chris Knight	Australia	FreeBSD builds
Neil McCalden	U.K.	Solaris builds
Grzegorz Prokopsi	Hungary	Debian builds
Paul Beach	U.K.	HP-UX builds
Geoffrey Speicher	U.S.A.	FreeBSD builds
Helen Borrie	Australia	Release notes author; field-tester and Thought Police

"The Field Test Heroes"

Pavel Kuznetsov, Eugene Kilin, Dmitry Kovalenko, Vladimir Kozloff, Barry Kukuk, Yakov Maryanov, Christian Pradelli, Daniel Rail, Volker Rehn, David Ridgway, David Rushby, Pavel Shibanov, Ruslan Strelba

Chapter 8

INSTALLATION NOTES

Windows 32-bit Installs

READ THIS FIRST!

With the introduction of two new server models on Win32 the choices for installing Firebird have proliferated.

- Make sure you are logged in as Administrator (doesn't apply on Win9x or ME)
- All models--Superserver, Classic and Embedded Server as well as server tools-only and client-only--can be installed using the Windows installer application. For a full-release install, it is highly recommended to use the installer if there is one available.
- Use gbak to back up your old isc4.gdb security database. You can restore it later as security.fdb
- If you have special settings in ibconfig there may be some values which you want to transfer to equivalent parameters in firebird.conf. Study the notes about firebird.conf to work out what can be copied directly and what parameters require new syntax.
- If certain configuration files exist in the installation directory they will be preserved if you run the installer and OVERWRITTEN if you decompress a zip kit into the default location. The files are

security.fdb
firebird.log
firebird.conf
aliases.conf

- Each model can be installed from a zipfile. This method will be faster than the installer if you have plenty of experience installing Firebird 1.5 from zipfiles. It will be highly exasperating if you are a Firebird newbie.

- It is assumed that.-
 1. you understand how your network works
 2. you understand why a client/server system needs both a server and clients
 3. you have read the rest of these release notes-or at least realise that you need to read them if something seems to have gone wrong
 4. you know to go to the firebird-support list if you get stuck. Join at <http://www.yahoogroups.com/groups/firebird-support>

If you already have an earlier version of Firebird or InterBase® on your server and you think you might want to go back to it, set up your fall-back position before you begin.

- Use the existing version of GBAK to back up your database files in transportable format
 - Go to your System directory and make a backup copy of gds32.dll. You might want to name the backup "gds32.dll.ib5" or "gds32.dll.fb103" or something similarly informative; or hide it in another directory
 - It might be a good idea to make a backup of the Microsoft C++ runtime, msvcp60.dll, too. The installer shouldn't overwrite your version of this file, but strange things have been known to happen.
 - STOP ANY FIREBIRD OR INTERBASE SERVER THAT IS RUNNING
- The installer will try to detect if an existing version of Firebird or InterBase is installed and/or running. In a non-installer install, you are on your own!
- The default root location of Firebird 1.5 will be C:\Program Files\Firebird\Firebird_1_5. If your old version is already in a directory of that name and you want to use the 1.5 defaults, rename the existing directory
 - For installing Firebird as a service: if you want to make use of the new secure login feature, create a "firebird service user" on the system-any name and password you like-as an ordinary user with appropriate privileges.

You should read the document named README.instsvc.txt first. If you have a zip kit, you will find it in the /doc directory of the zipfile's root. If you don't have a zip kit available, the file won't be available until after the installation. You can read the same document at this URL: <http://cvs.sourceforge.net/viewcvs.py/firebird/firebird2/doc/README.instsvc>

Naming databases on Windows

Note that now the recommended extension for database files on Windows ME and XP is ".fdb" to

avoid possible conflicts with "System Restore" feature of Windows. Failure to address this issue on these platforms will give rise to the known problem of delay on first connection to a database whose primary file and/or secondary files are named using the conventional ".gdb" extension.

The issue is described in more detail in [Other Win32 Issues](#) at the end of the Windows installation notes.

READ THIS NEXT!

One of the design goals of Firebird 1.5 is to prepare the way for multiple installs of the server. This will allow users to run different versions side by side. Firebird 1.5 does support this, although it is not well documented and very much requires intervention from a skilled user. Future versions of Firebird will make this process far less complicated. In the meantime Firebird 1.5 needs to prepare the ground. This forces us to confront the issue of library installation.

At the same time, Microsoft have taken their own steps to manage installation of different library versions. Taken together these two separate issues mean a new approach to library installation for Firebird 1.5 and beyond.

Installation of Microsoft system libraries

The problems associated with installing different versions of Microsoft system libraries are so notorious that it has acquired the name 'DLL Hell'. From the release of Windows 2000 onwards Microsoft have made it almost impossible to upgrade system dll's. To resolve this Microsoft now recommends that each application installs local copies of any system libraries that are required.

Firebird 1.5 follows this practice and places the required libraries in the \bin directory along with the server.

Installation of fbclient.dll

Firebird 1.5 and beyond no longer use gds32.dll as the client library. It is now called fbclient.dll. Given the problems that Microsoft have had with DLL hell it wouldn't make much sense if we continued to store the Firebird client library in the system directory. And as we want to allow installation of multiple engines simultaneously we would be creating our own DLL hell if we continued the practice of using the system directory for the client library.

So, from Firebird 1.5 on, the client library resides in the \bin directory along with all the other binaries.

New Registry Key

A new registry key has been added and all Firebird 1.5 compliant applications should use this key if they need to read a Registry key to locate the correct version of Firebird that they wish to use. The new key is:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Firebird Project\Firebird Server\Instances
```

Firebird will guarantee that one entry under this key always exists. It will be known as

"DefaultInstance"

and will store the path to the root directory of (yes, you've guessed it) the default installation. Those that don't care about particular installations can always use the default instance to locate the fbclient.dll.

Future versions of Firebird will see other entries under Instances. Applications will be able to enumerate the registry entries to determine which library instance they wish to load.

Supporting legacy applications and drivers

Traditionally, applications that use InterBase or Firebird have expected to load the gds32.dll client library from the system directory. Firebird 1.5 ships a tool named 'instclient.exe' that can install a clone of fbclient.dll to the Windows System directory. This clone gets patched on the fly so that its file version information begins with "6.3", to provide compatibility for old applications that check the GDS32.DLL file version and can not make sense of a number string such as "1.5".

During the installation process the installer checks to see if an installation of InterBase or Firebird exists. If nothing is installed it will write gds32.dll into the system directory. If it detects that any possible version of Firebird or InterBase may already be installed it will not install the gds32.dll in the system directory. However, the 'instclient.exe' tool can be used to do it later.

It is intended that future versions of Firebird will not attempt to install gds32.dll into the system directory and ultimately it will be completely removed from the distribution.

InstClient.exe Tool

This 'instclient.exe' tool can also install the FBCLIENT.DLL itself in the Windows system directory, if required. This will take care of tools or applications that need to load it from there.

The instclient.exe utility should be located in the 'bin' directory of your Firebird installation and must be run from there in a command shell.

Usage of instclient.exe:

```
instclient i[nstall] [ -f[orce] ] library
           q[query] library
           r[emove] library
```

where library is: fbclient | gds32

'-z' can be used with any other option, prints version.

Version information and shared library counts are handled automatically. You may provide the -f[orce] option to override version checks.

Caution

If you -f[orce] the installation, it could break another Firebird or InterBase® version already installed. You might have to reboot the machine in order to finalize the copy.

For more details, see the document README.Win32LibraryInstallation.txt which is located in either

the root of your installation path or in ..\doc.

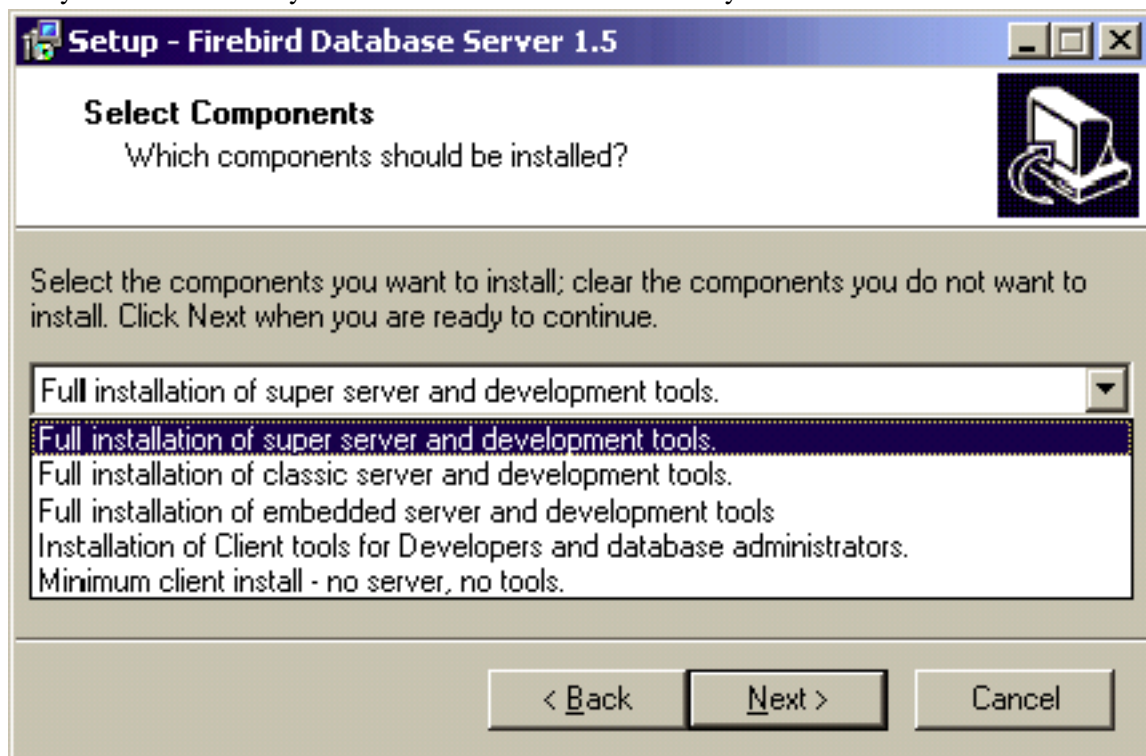
Cleaning up release candidate installs

It should be noted that the installer removes fbclient.dll from the <system> directory if the file is found there. The installer also removes the deprecated HKLM\Software\FirebirdSQL registry key.

Using the Win32 Firebird Installer

This is the easy one.

Just run the executable and respond to the dialogs. After you have answered about four dialogs, you should see one with a drop-down list box which--when dropped down--looks similar to this. This really is the last chance you'll have to choose the installation you want:



Choose the installation you want and hit "Next" to carry on responding to dialogs.

Service or application?

If you select to install Superserver or Classic, and your OS version supports services, you will be asked to choose whether to run Firebird as a service or as an application. Unless you have a compelling need to run the server as an application, choose service.

Manual or automatic?

With automatic option, Firebird will start up whenever you boot the host machine. With the manual option you can start the server on demand.

Guardian option

Guardian is a utility than can run "over the top" of Superserver and restart it if it crashes for any reason. For development, you might choose not to use it. For deployment, it can avoid the situation where the server stops serving and nobody can find the DBA to restart it.

Installation (Root) directory

If you decide not to use the default root location, browse to a location you have pre-created; or just type in a full path. The path you type in doesn't have to exist: the installer will prompt you and create it if it doesn't exist.

Eventually, the dialogs will stop and the server will either silently start the server or prompt you for permission to reboot. Reboot will be needed if the installer overwrote your msvc60.dll, or an old gds32.dll was already loaded when the installer started up.

Uninstallation

The Firebird uninstall routine (run from Add/Remove Programs in the Control Panel) preserves and renames the following key files:

- preserves security.gdb or renames it to security.fbnnnn
- preserves firebird.log
- preserves firebird.conf or renames it to firebird.confnnnn
- preserves aliases.conf or renames it to aliases.confnnnn

"nnnn" is the build number of the old installation.

No attempt is made to uninstall files that were not part of the original installation.

Shared files such as fbclient.dll and gds32.dll will be deleted if the share count indicates that no other application is using them.

The Registry keys that were created will be removed.

Installing Superserver from a zip kit

The installation of FB 1.5 is similiar in principle to previous versions. If you don't have a special setup program (it's distributed separately) the steps are the following:

- unzip the archive into the separate directory (since a few file names were changed, it doesn't make sense to unzip v1.5 files into the directory with IB/FB1)
- change the current directory to \$FIREBIRD\bin (here and below \$FIREBIRD is the directory where v1.5 files are located)
- run instreg.exe:

```
instreg.exe install
```

It causes the installation path of the directory above to be written into the registry (HKLM\Software\Firebird Project\Firebird Server\Instances\DefaultInstance)

- if you want to register a service, also run instsvc.exe:

```
instsvc.exe install
```

- optionally, you may need to copy both fbclient.dll and gds32.dll to the OS system directory

Installing Classic Server from a zip kit

To install the CS engine, the only difference is the additional switch for instsvc.exe:

```
instsvc.exe install -classic
```

Important

Notice that this means that you may have only one architecture of the engine--either fbserver.exe (Superserver) or fb_inet_server.exe (the parent process for Classic)--installed as a service.

The Control Panel applet is not installed with Classic--deliberately. Don't try to install and use it. The concept of terminating a service does not apply to the Classic model.

Simplified setup

If you don't need a registered service, then you may avoid running both instreg.exe and instsvc.exe. In this case you should just unzip the archive into a separate directory and run the server:

```
fbserver.exe -a
```

It should treat its parent directory as the root directory in this case.

Uninstallation

To remove FB 1.5 without a Windows Uninstaller you should:

- stop the server
- run "instreg.exe remove"
- run "instsvc.exe remove"
- delete installation directory

- delete fbclient.dll and gds32.dll from the OS system directory

Installing Embedded Server from a Zip Kit

The embedded server is a client with a fully functional server linked as a dynamic library (fbembedded.dll). It has exactly the same features as the usual Superserver and exports the standard Firebird API entry points.

Registry

The Registry entries for Firebird (where the server normally looks for the location of the root directory) are ignored. The root directory of the embedded server is the directory above where its binary file (library) is located.

Database access

Only "true local" access is allowed. The embedded server has no support for remote protocols, so even access via localhost won't work.

Authentication and security

The security database (security.fdb) is not used in the embedded server and hence is not required. Any user is able to attach to any database. Since both the server and the client run in the same (local) address space, security becomes a question of physical access.

SQL privileges are checked, as in other server models.

Compatibility

You may run any number of applications with the embedded server without any conflicts. Having a full Firebird or InterBase® server running is not a problem either.

But you should be aware that you cannot access the same database from multiple embedded servers simultaneously, because they have SuperServer architecture and hence exclusively lock attached databases.

File structure for the Embedded Server

Just copy fbembedded.dll into the directory where your application resides. Then rename it to either fbclient.dll or gds32.dll, depending on your database connectivity software. Make copies having both names if you will need to use the server tools (isql, gbak, etc.)

You should also copy firebird.msg, firebird.conf (if necessary) and ib_util.dll to the same directory.

If external libraries, are required for your application, e.g. INTL support (fbintl.dll) or UDF libraries, they should be located apart from the application directory. To be able to use them, place them into a directory tree which emulates the Firebird server one, i.e., in subdirectories named /intl and /udf directly beneath the directory where the Firebird root files are.

Important

Don't overlook the need to have the Microsoft® Visual C and Visual C++ runtimes (msvcrt.dll and msvcp60.dll, respectively) present in the system directory.

Example

D:\my_app\app.exe


```
D:\my_app\gds32.dll (renamed fbembed.dll)
D:\my_app\fbclient.dll (renamed fbembed.dll)
D:\my_app\firebird.conf
D:\my_app\aliases.conf
D:\my_app\isql.exe
D:\my_app\ib_util.dll
D:\my_app\gbak.exe
D:\my_app\firebird.msg
D:\my_app\intl\fbintl.dll
D:\my_app\udf\fbudf.dll
```

Then, start your application. It will use the embedded server as a client library and will be able to access local databases.

Note

Provided you set up the directory structure according to these rules, it will not be necessary to configure the RootDirectory explicitly in firebird.conf. However, if you decide to deploy the embedded server and your application in some different directory structure, be sure to read the document README_embedded.txt of the Embedded Server distribution first, for instructions about additional configuration.

Other Win32 Issues

Winsock2

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. To find out how to go about upgrading, [visit this link](#).

Windows ME and XP

Windows ME and XP (Home and Professional editions) there is a feature called *System Restore*, that causes auto-updating (backup caching?) of all files on the system having a ".gdb" suffix. The effect is to slow down InterBase/Firebird database access to a virtual standstill as the files are backed up every time an I/O operation occurs. (On XP there is no System Restore on the .NET Servers).

A file in the Windows directory of ME, c:\windows\system\filelist.xml, contains "protected file types". ".gdb" is named there. Charlie Caro originally recommended deleting the GDB extension from the "includes" section of this file. However, since then, it has been demonstrated that WinME might be rebuilding this list. In XP, it is not possible to edit filelist.xml at all.

On ME, the permanent workarounds suggested are one of:

- use FDB (Firebird DB) as the extension for your primary database files--RECOMMENDED
- move databases to C:\My Documents, which is ignored by System Restore
- switch off System Restore entirely (consult Windows doc for instructions).

On Windows XP Home and Professional editions you can move your databases to a separate partition and set System Restore to exclude that volume.

Windows XP uses smart copy, so the overhead seen in Windows ME may be less of an issue on XP, for smaller files at least. For larger files (e.g. Firebird database files, natch!) there doesn't seem to be a better answer as long as you have ".gdb" files located in the general filesystem.

XP Shutdown

Windows XP shutdown behaviour is a known "dark area". There is a significantly long pause while the server service stops. During that time the display indicates that Firebird is running as an application.

The problem only seems to affect Windows XP and it only appears if the Guardian is not being used to stop the server service. That is the workaround until a fix can be found.

POSIX Platforms

(Originally by Mark O'Donohue, revised for 1.5)

The Firebird server comes in two forms, Classic, which runs as a service, and SuperServer, which runs as a background daemon. Classic is the more traditional UNIX service, while Superserver uses threads, rather than processes. For the user just starting out with Firebird, either will do, although the Classic server is likely to prove a better platform for initially experimenting with Firebird.

READ THIS FIRST

- You will need to be root user to install Firebird.
- Installation on Linuxen requires a glibc package installed that is equal to or greater than glibc-2.2.5 and a libstdc++.so equal to or greater than libstdc++-5.0.
- For a rough evaluation of compatible Linuxen, refer to the section [Linux Compatibilities](#), but don't take it as the "last word". Linux binary distros out-of-the-box can vary depending on where and when they were built.
- Ensure that the 'ed' and 'vim' editor packages are installed on your system. If the required editor is not present, the package will install but the installation scripts will fail.

Note

This dependency may in future be replaced by 'sed'.

Installing on Linux

The following instructions describe the Classic installation. For installation of Superserver the "CS" in the package name is repaced by "SS". For example, the package `FirebirdCS-1.5.0-nnnn.i686.rpm` is replaced by `FirebirdSS-1.5.0-nnnn.i686.rpm`.

Log in as root, or open a root shell. In the example filenames below, replacy *nnnn* with the build number of the kit you actually have.

RPM Installer

For the RPM installer, type:

```
$rpm -ivh FirebirdCS-1.5.0-nnnn.i686.rpm
```

Installing the Tarball

To install the tarball, place the ".tar.gz" file and type:

```
$tar -xzf FirebirdCS-1.5.0-nnnn.tar.gz
$cd FirebirdCS-1.5.0-nnnn.i686
$./install.sh
```

What the Linux install scripts will do

The Linux install scripts will

1. Attempt to stop any currently running server
2. Add the user 'firebird' and the group 'firebird' if they do not already exist.
3. Install the software into the directory /opt/firebird and create links for libraries in /usr/lib and header files in /usr/include
4. Automatically add gds_db for port 3050 to /etc/services if the entry does not already exist
5. Automatically add localhost.localdomain and HOSTNAME to /etc/host.equiv
6.
 - a. SuperServer only installs a /etc/rc.d/init.d/firebird server start script.
 - b. Classic server installs a /etc/xinetd.d/firebird start script or, for older inetd systems, adds an entry to the /etc/inetd file
7. Specific to SuSE, a new rcfirebird link is created in /usr/bin for the init.d script and an /etc/rc.config Firebird entry is created.
8. Starts the server/service. Firebird should start automatically in runlevel 2, 3 or 5

9. Generates and sets a new random SYSDBA password and stores it in the file /opt/firebird/SYSDBA.password.
10. Adds an entry to aliases.conf for the sample database, employee.fdb.

Testing your Linux installation

Step 1 - Accessing a database

In a shell:

```
$cd /opt/firebird/bin
$./isql -user sysdba -password <password>1

SQL>connect localhost:employee.fdb /* this is an aliased path */

SQL>select * from sales;
SQL>select rdb$relation_name from rdb$relations;
SQL>help;

SQL>quit;
```

Note

¹A password has been generated for you on installation. It can be obtained from the /opt/firebird/SYSDBA.password file, located in the Firebird root directory.

Step 2 - Creating a database

From version 1.5 onward, the Firebird server runs by default as the user 'firebird'. While this has always been the recommended configuration, the previous default was for the server to run as 'root' user. When running as root user, the server had quite wide-ranging ability to read, create and delete database files anywhere on the POSIX filesystem.

For security reasons, the service should have a more limited ability to read/delete and create files.

While the new configuration is better from a security perspective, it requires some special considerations to be taken into account for creating new databases:

1. the user 'firebird' has to have write permission to the directory in which you want to create the database.
2. the recommended value of the DatabaseAccess attribute in the /opt/firebird/firebird.conf file should be set to None, to permit access only through entries in the aliases.conf file.
3. use entries in aliases.conf to abstract users from the physical locations of databases. More notes on aliases are to be found in the topic [Database File Aliasing](#).

Procedures for creating a new database can vary with different configurations but the following configuration and steps are recommended:

1. If a directory that is owned by the user 'firebird' does not exist, then change to root user and create the directory:

```
$su - root
$mkdir -p /var/firebird
$chown firebird:firebird /var/firebird
```

2. Create a new physical database and set up an alias entry to point to it. As root or firebird user, run the following script:

```
$cd /opt/firebird/bin
$./createAliasDB.sh test.fdb /var/firebird/test.fdb
```

(Usage is: createAliasDB.sh <dbname> <pathtodb>)

3. As an alternative (for step 2) the steps in the createAliasDB.sh script can be performed manually by:

```
$vi /opt/firebird/aliases.conf
```

and add the line at the end of the file:

```
test.fdb /var/firebird/test.fdb
```

4. Then create the database:

```
$/opt/firebird/bin/isql -u sysdba -p <password>
SQL>create database 'localhost:test.fdb';
SQL>quit;
```

5. If the *DatabaseAccess* value in /opt/firebird/firebird.conf is set to Full or a restricted path value (for example: DatabaseAccess=/var/firebird) another alternative to step 2 is to create the physical database file directly, using the absolute path with the filename:

```
$/opt/firebird/bin/isql -u sysdba -p <password>
SQL>create database '/var/firebird/test.fdb';
SQL>quit;
```

If you use this configuration, the database file can also be directly accessed without an entry in

the aliases file:

```
$/opt/firebird/bin/isql -u sysdba -p <password>
SQL>connect '/var/firebird/test.fdb';
SQL>quit;
```

Utility Scripts

In addition to the standard install files the following scripts are provided in the bin directory of this release.-

changeDBAPassword.sh

Change the Firebird SYSDBA user password. For Superserver, this script will change the init script /etc/rc.d/init.d/firebird to use the new password as well.

createAliasDB.sh

Usage: createAliasDB.sh <dbname> <dbpath>

This script creates a new physical database and adds an entry in the aliases.conf file.

fb_config

A script that can be used in makefiles to generate the required include paths and lib include directives for the installed version of Firebird. *fb_config -help* will give a complete list of options.

changeGdsLibraryCompatibleLink.sh

Classic only-Change the client library link for libgds.so between the multithreaded libfbclient.so and the single threaded libfbembed.so library that allows an embedded direct open of the db file. For compatibility with previous installs, libgds.so by default points to libfbembed.so.

Linux Server Tips

"Embedded" or direct access to database files

The Classic install offers an "embedded" mode of access that allows programs to open database files directly. To operate in this mode, a database-enabled user requires privileged access to some of the Firebird configuration and status files.

Now that it is the 'firebird' user (not root) that is the default user to run the software, you need to know how to get a user into the firebird group to enable direct access to databases. It is documented in the readme notes, but the following steps should get you where you need to be.

To add a user (e.g. skywalker) to the firebird group, the root user needs to do:

```
$ usermod -G firebird skywalker
```

Next time 'skywalker' logs on, he can start working with firebird databases.

To list the groups that a user belongs to, type the following at the command line:

```
$ groups
```

NTPL problems on higher Linuxen

The new NPTL (Native POSIX Thread Library) in Red Hat 9 (so far) will cause problems for SuperServer and locally compiled programs, including the utilities. Gbak, particularly, will throw a *Broken Pipe* error. To fix:-

1. In /etc/init.d/firebird

```
LD_ASSUME_KERNEL=2.2.5
export LD_ASSUME_KERNEL
```

That takes care of the server instance.

2. You need to have the LD_ASSUME_KERNEL environment variable set up within the local environment as well, so add the following to /etc/profile, to ensure every user picks it up for the command line utilities.

after

```
HISTSIZE=1000
```

add

```
LD_ASSUME_KERNEL=2.2.5
```

On the following line, export it (this is all in one line):

```
export PATH USER LOGNAME MAIL HOSTNAME
HISTSIZE INPUT_RC LD_ASSUME_KERNEL
```

Uninstalling on Linux

If you need to uninstall, do it as root user. The following examples use Classic server but the same holds true for SuperServer by replacing the CS with SS.

Uninstalling an RPM package

For rpm packages:

```
$rpm -e FirebirdCS-1.5.0
```

Uninstalling a tarball installation

for the .tar.gz install:

```
$/opt/firebird/bin/uninstall.sh
```

Solaris

Install Firebird Classic & SuperServer on Solaris 2.7 Sparc, not currently available. Please refer to v.1 releasenotes as a reference to 1.5 installations.

MacOS X

Install Firebird Classic on MacOS X / Darwin, not currently available. Please refer to v.1 releasenotes as a reference to 1.5 installations.

FreeBSD

Not currently available. Please refer to v.1 releasenotes as a reference to 1.5 installations.

Debian

Not currently available. Please refer to the relevant pages at the Debian site for your Debian version and Firebird 1.5 build.

Chapter 9

Configuring the Port Service on Client and Server

By default a Firebird server listens on port 3050 for connection requests from clients. Its registered port service name is *gds_db*. The good news is that, if you can go with these defaults, you have to do nothing on either server or clients to configure the port service.

You can use a different port, a different port service name, or both. You might need to do this if port 3050 is required for another service, for example, a concurrently running *gds_db* configured for a different version of Firebird or for an InterBase® server.

There are several ways to override the defaults. Both the server and the clients must be configured to override the port service name or number, or both, in at least one of these ways:

- in the client connection string
- in the command used to start the server executable
- by activating the `RemoteServicePort` or `RemoteServiceName` parameters in `firebird.conf` (V.1.5 onward)
- in the daemon configuration (for Classic on POSIX)
- by an entry in the Services file

Before examining each of these techniques, it will be helpful to look at the logic used by the server to set the listening port and by the client to set the port that it should poll for requests.

How the server sets the listening port

The server executable has an optional command-line switch (`-p`) by which it can signify either the port number it will be listening on or the name of the port service that will be listening. At this point, if the switch is present, either port number 3050 or the port service name (*gds_db*) is replaced by the argument supplied with the `-p` switch.

Next--or first, if there is no `-p` switch--a v.1.5 server checks `firebird.conf` to look at the parameters `RemoteServiceName` and `RemoteServicePort`:

- If both are commented with `"#"` then the defaults are assumed and no further change occurs. Any -

p argument stands and the "missing" argument remains as the default.

- If RemoteServiceName has been uncommented, but not RemoteServicePort, then the port service name is substituted only if it has not been overridden already by the -p switch.
- If RemoteServicePort has been uncommented, but not RemoteServiceName, then the port number is substituted only if it has not been overridden already by the -p switch.
- If both RemoteServicePort and RemoteServiceName are uncommented, then the RemoteServiceName takes precedence if it has not already been overridden by a -p argument. If there is already a port service name override, the RemoteServiceName value is ignored and the RemoteServicePort value overrides 3050.
- At this point, if an override of either the port number or the service name has been signaled, both v.1.0 and v.1.5 servers proceed to check the Services file for an entry with the correct combination of service name and port number. If a match is found, all is well. If not, and the port service name is not gds_db, the server will throw an exception and fail to start. If gds_db is the port service name and it cannot be resolved to any other port, it will map to port 3050 automatically.

If the default service name is to be overridden, then you will need to make an entry in the Services file--see the topic below, [Configuring the Services File](#).

Using the -p switch override

Note

Please note that this switch is available in Firebird 1.0.x, too, but was previously undocumented.

Starting the server with the optional -p switch enables you to override either the default port number (3050) or the default port service name (gds_db) that the server uses to listen for connection requests. The switch can override one, but not both.

From Firebird 1.5 onward, you can use the -p switch in combination with a configuration in firebird.conf to enable an override to both the port number and the port service name.

Syntax for TCP/IP

The command-line syntax for starting a server that applications are going to access through TCP/IP is as follows:

```
server-command <other switches> -p port-number | service-name
```

For example, to start the Superserver as an application and override the service name *gds_db* with *fb_db*:

```
fbserver -a -p fb_db
```

Or, to override port 3050 to 3051:

```
fbserver -a -p 3051
```

Syntax for WNet redirection

On a WNet network (Named Pipes, NetBEUI), replace the -p switch argument syntax above with the following "backslash-backslash-dot-at" syntax:

```
fbserver -a -p \\.@fb_db
```

Or:

```
fbserver -a -p \\.@3051
```

Classic on POSIX: the inetd or xinetd daemon

With Firebird Classic server on Linux or UNIX, the inetd or xinetd daemon is configured to listen on the default port and broadcast the default service name. The installation script will write the appropriate entry in the configuration file `/etc/inetd.conf` or `/etc/xinetd.conf`.

You can edit the current configuration if necessary. Following is an example of what you should see in `xinetd.conf` or `inetd.conf` after installing Firebird Classic on Linux:

```
# default: on
# description: FirebirdSQL server
#
service gds_db
{
    flags                = REUSE KEEPALIVE
    socket_type          = stream
    wait                 = no
    user                 = root
# user                  = @FBRUser@
    log_on_success        += USERID
    log_on_failure        += USERID
    server                = /opt/firebird/bin/fb_inet_server
disable                 = no
}
```

If you configured the port service to be different to the defaults, you will need to alter `xinetd.conf` or `inetd.conf` accordingly. Restart `xinetd` (or `inetd`) with `kill -HUP` to make sure the daemon will use the new configuration.

Caution

Connection requests will fail if both xinetd (or inetd) and fbserver (or ibserver) attempt to listen on the same port. If your host machine has this double configuration, it will be necessary to set things up so that each server version has its own service port.

Using a configuration file parameter

You can configure either `RemoteServiceName` or `RemoteServicePort` in `firebird.conf` to override either the default port number (3050) or the default port service name (`gds_db`) that the server uses to listen for connection requests.

The engine will use one `RemoteService*` parameter, but not both. If you configure both, it will ignore `RemoteServicePort` in all cases, except where the server start command was invoked with the `-p` switch supplying an override to the port service name. Thus, you can use the `-p` switch and a `RemoteService*` parameter, in combination, to override both port number and service name.

If the default service name is to be overridden, then you need to make an entry in the Services file.

Tip

GOTCHA! If you uncomment `RemoteServiceName` or `RemoteServicePort`, but leave the default values intact, they will be treated as overrides. It will then be necessary to make an explicit entry in the services file for the default port service settings.

Setting up a client to find the service port

If you set up your server with the installation defaults (service `gds_db` listening on port 3050) then no configuration is required. If the server is listening on a different port or is using a different port service name, the client application and/or its host machine need some enabling configuration to help the Firebird client library to find the listening port.

The connection string used by a client can include information for polling the server's listening port in various ways. Firebird 1.5 clients can optionally use a local copy of `firebird.conf`. Changes may also be needed in the client's services file.

Using the connection string

If only the port number or the service name has been reconfigured, then include the alternative port number or service name in the connection string. This works for all versions of Firebird.

Syntax for TCP/IP connections

To connect to a database named `server` named `alice` that is broadcasting on port 3050 with the service name `fb_db`, the connection string would be:

For POSIX:

```
alice/fb_db:/data/teaparty.fdb
```

Or, if the service name is gds_db and the port number is 3051:

```
alice/3051:/data/teaparty.fdb
```

For Windows:

```
alice/fb_db:D:\data\teaparty.fdb
```

Or, if the service name is gds_db and the port number is 3051:

```
alice/3051:D:\data\teaparty.fdb
```

Note

Notice that the separator between the server name and the port is a slash, not a colon. The colon before the physical path string is still required.

Syntax for WNet connections

On a WNet network, use UNC-style notation:

```
\\alice@3051\d:\teaparty.fdb
```

or

```
\\alice@fb_db\d:\teaparty.fdb
```

If the server's configured port number or service name is an override, then you need to make an entry in the Services file.

Using port syntax with database aliases

To connect through a non-default port with a database alias, affix the port number or service name to the server name, not to the alias. For example, suppose the database alias is stored in aliases.conf as

```
rabbit = /data/teaparty.fdb
```

Your application's connection string for server 'alice' would be:

```
alice/fb_db:rabbit
```

or

```
alice/3051:rabbit
```

Using a copy of firebird.conf

From Firebird 1.5 onward, you can optionally include a client-side copy of `firebird.conf` in the `firebird` root directory and configure `RemoteServiceName` or `RemoteServicePort` to help the client to locate the server port.

- You can configure one of these two parameters to extend the override provided for the other one through the connection string (above); or to override only the `RemoteServiceName` or the `RemoteServicePort` without using the connection string to do it.
- If you need to avoid passing the port service name or the port number in the connection string and the server is using non-defaults for both, you can configure both `RemoteServiceName` and `RemoteServicePort`. You would use this technique if your client application needed to retain the capability to connect to InterBase or Firebird 1.0 servers.

Location of Firebird artifacts on clients

When you rely on `firebird.conf` on client machines, it is important that the client library knows where to find it. You will need to set up a Firebird root directory and inform the system of its location. Use the `FIREBIRD` environment variable to do this. Windows clients can alternatively use the Firebird Registry key. With a correct client setup, you can also make use of a local version of the message file.

Configuring the Services File

You do not need to configure a service port entry for your Firebird server or clients if the server uses the installation defaults, `gds_db` on port 3050. If `gds_db` is the port service name and it cannot be resolved to any other port, it will map to port 3050 automatically.

If you are configuring the service port for a different port or service name, both the server and the client must be explicitly updated to reflect the reconfiguration. On both Linux and Windows, this information is stored in the services file.

Locating the services file

Windows XP/S2003

`C:\windows\system32\drivers\etc\services`

Windows NT/2000

`C:\winnt\system32\drivers\etc\services`

Windows 95/98/ME

`C:\windows\services`

Linux/UNIX

`/etc/services`

A services entry looks like this:

```
gds_db 3050/tcp # Firebird Server 1.5
```

Open the file using a text editor and either add a line or edit the existing `gds_db` entry, as follows:

- for a Firebird 1.0.x server or client, alter either the service name or the port number to reflect how your server will start up.
- for a Firebird 1.5 or higher server, edit or add a line, as required. If you have a Firebird 1.0 or Inter-Base server installed on the same host, retain the entries they require and add a new entry reflecting how the Firebird 1.5 server will start up.

Chapter 10

Further Information

Firebird Development

More information can be found about the Firebird database engine from [the Firebird Project website](#).

If you are interested in being involved in Firebird development, or would like to raise a possible bug for discussion, please feel welcome to join our Firebird-Devel list. To subscribe, simply send an empty email message to:
`<firebird-devel-request@lists.sourceforge.net?subject=subscribe>`.

Important

Please do not use the firebird-devel list for posting support questions, *except* if your question happens to be about *building Firebird from sources*.

The core developers (amongst others!) also hang out in the Firebird-Architect list. This is also a non-support list, where architectural features and changes are discussed, often with great vehemence. See the next section for list subscription information.

Lists and Newsgroups

For technical support, please join the firebird-support list by going to the [Lists and Newsgroups](#) index page at the Firebird website and clicking on the link to join the **Firebird-support list**. The Firebird-support list handles technical problems with Firebird and also InterBase®.

At the Lists and Newsgroups index page you will find links to all of the more specialised lists as well: Firebird-Java for Jaybird support, Firebird-NET-Provider, Firebird-ODBC-devel, Firebird-PHP, Firebird-Python, etc., as well as lists for the various Delphi interface components.

The open source community operates several other discussion lists on various aspects of Firebird development. For details, please refer to the Lists and Newsgroups index page.

Newsgroup Mirrors

The Firebird developers' list and the general community lists, along with some other lists of interest to Firebird and InterBase developers, are mirrored as newsgroups at [the news.atkin.com news server](#). For historical reasons, some of them have weird names, e.g. "egroups.ib-support" is the mirror for the firebird-support list.

Tip

If you want to post messages from a news mirror to the list, you must be subscribed to *that list*.

Paid Support

Paid support worldwide for Firebird can be arranged through IBPhoenix. Contact addresses and numbers are at the [IBPhoenix site](#).

Database recovery services for Firebird or InterBase databases can be handled by IBPhoenix. For analyzing and possibly repairing corrupted databases yourself, try [IBSurgeon.com](#). The IBSurgeon team also provides recovery services.

Sponsorship

Requests/offers for sponsored enhancements to Firebird can be taken directly to the [Firebird Foundation Inc.](#), a non-profit organisation of Firebird community members that manages sponsorship and donor funding for the Firebird Project. Send an email to <foundation@firebirdsql.org>.

You may prefer to contact the Firebird project admins initially - send an email to <firebirds@users.sourceforge.net>.

Tools and Drivers

Following is a selection of third-party products, both open source and commercial, that are available and used widely by Firebird developers.

Database Admin Tools

Several excellent choices of GUI admin desktops for Firebird are listed in the Contributed Downloads page at the [IBPhoenix site](#). Some are open source, some are freeware, others are established commercial products.

Note

Borland's IBConsole program is not recommended as a database administration client for Firebird 1.5.

Drivers and Components

JAVA

The Jaybird JDBC driver project develops actively as part of the Firebird project. It has a released Type 4 JDBC/JCA driver and a Type 2 driver in beta. Sources and binaries can be downloaded from the [Firebird release pages at Sourceforge](#).

For advice and to participate in development and testing, join up with the [Firebird-java forum](#) on Yahoogroups.

.NET and Mono

Firebird has an ongoing .NET driver project. Sources and binaries can be downloaded from the [Firebird release pages at Sourceforge](#).

For advice and to participate in development and testing, join up with the [Firebird .NET provider forum](#) at Sourceforge.

Delphi and C++Builder

Users have two powerful alternatives for full, direct connectivity with the Firebird 1.5 API, both well supported with developer and peer support:

- Jason Wharton's IB Objects at <http://www.ibobjects.com>
- FIBPLus at <http://www.devrace.com>

C++

the freeware C++ access library 'IBPP' (<http://www.ibpp.org>), MPL license, hosted on sourceforge.net, fully portable between Win32 and Linux and probably other POSIX platforms. It is useful when you want low-level C-API kind of access, but with a light C++ abstraction level not bound to any particular development environment.

ODBC

A list of ODBC drivers can be found listed in the Contributed Downloads page at [the IBPhoenix site](#). Lab for ongoing development of the open source Firebird/IBPhoenix ODBC/JDBC driver is in this forum: <http://lists.sourceforge.net/lists/listinfo/firebird-odbc-devel>

PHP

A group is continually working on bringing the old InterBase PHP extension up to standard for Firebird. To ask about this project, join up with the Firebird-PHP forum at <http://www.yahoogroups.com/community/firebird-php>

PYTHON

KInterbasDB is a Python extension package that implements Python Database API 2.0-compliant support for Firebird. Full native API support for the Firebird client; in stable release and under active development. BSD open source licence. Downloads and news at <http://kinterbasdb.sourceforge.net/>

Documentation

The documentation for InterBase v 6.0 applies also to the current FireBird release. A beta version of InterBase(tm) 6 manuals is available in Adobe Acrobat format from various sources on the web. They used to be available from the Borland archives but the links have been removed. The easiest way to find a download site is to Google search for "opguide.pdf". That is the name of just one of the six manuals but, where you find one, you will generally find them all.

Important

RELEASE NOTES (such as this document) provide the *incremental documentation* for the Firebird releases and sub-releases. It is important to keep them on hand as the supplement to the old InterBase® beta manuals.

A structured [Documentation Index](#) is maintained on the Firebird community site. This is work-in-progress and all additions are welcome - send a message to <firebird-docs@lists.sourceforge.net>

Several installation guidelines and other HowTos may be found in the documentation area which can be linked to from the Resources menu at the top of the main Firebird website pages. Keep watching these links as documentation is continually under development.

The authoritative bookshop title for Firebird, in English, is "The Firebird Book: A Reference for Database Developers" by Helen Borrie (publ. Apress, August 2004, ISBN 1590592794). It is easily obtainable from bookstores, from IBPhoenix and from a large number of web-based book outlets including Amazon.

The main repository for documentation and white papers on user and technical issues is the [IB-Phoenix site](#). IB Phoenix also publishes a comprehensive subscription CD on a regular basis (single issues also available) which contains manuals published by them - Using Firebird and The Firebird Reference Guide.

Some additional documentation may be discovered by visiting the [Borland techpubs area](#).

Chapter 11

Bugfixes and Additions since Release 1.0

The figure in brackets, e.g. (1.5) indicates the first release or sub-release in which the described improvement or bug-fix was distributed.

Improvements

instsvc/instreg executables

By O. Mascia

(1.5) Enhanced Win32 install tools instsvc.exe and instreg.exe

Increased # of indexes per table

By A. Harrison, ported to 1.5 by N.Samofatov

(1.5) Maximum number of indices per table increased from 64 to (DB_PAGE_SIZE/16)-2

Better message detail

By D. Yemanov, A. Brinkman, A. Peshkoff

(1.5) Added new (more specific) error messages for some of v1.5 changes.

Security fix on Windows

By A. Peshkoff

(1.5) Added -login switch to instsvc allowing FB service to be installed as non-localsystem account.

Varchar trimming over wire

By D. Yemanov

(1.5) Re-introduced trimming of VARCHAR fields in the remote protocols.

Configuration improvement

By A. Peshkoff

(1.5) Made path management in firebird.conf conform to the OS requirements.

POSIX Tweak

By N. Samofatov

(1.5) Allow easy adjustment of LockSemCount on POSIX platforms, without need to use gds_drop or reboot machine to make new setting take effect

Keyword reduction

By N. Samofatov

(1.5) Make FIRST/SKIP keywords non-reserved.

PSQL Extensions for Loop Breakout

By D. Yemanov

(1.5) BREAK/LEAVE and EXIT statements are now available for usage in triggers.

Mix Aggregate Contexts

By A. Brinkman

(1.5) Enabled aggregate functions from different parent context to be used inside another aggregate function.

Example:

```
SELECT MAX((SELECT COUNT(*) FROM RDB$RELATIONS))  
FROM RDB$RELATIONS
```

INTL improvement: UPPER()

By N. Samofatov, D. Yemanov

(1.5) Make UPPER function work for WIN1251 charset without explicit collations.

Configuration Manager change

By A. Peshkoff

(1.5) Now the server will exit from encountering a missing or wrong firebird.conf by sending an error report to the system log.

Parser changes

By D. Yemanov

(1.5) Changed parser.-

1. ROWS_AFFECTED is renamed to ROW_COUNT
2. CONNECTION_ID/TRANSACTION_ID are renamed to CURRENT_CONNECTION/CURRENT_TRANSACTION
3. Some of the newly introduced tokens are made non-reserved

Improved lock manager

By N. Samofatov

(1.5) Deadlocks are now detected and reported as soon all blocking processes received notifications, i.e. instantly in all normal cases.

Advanced security capabilities

By A. Peshkoff

(1.5) Implemented configurable access for databases, external tables and UDF libraries.

SQL Enhancement

By D. Yemanov, N. Samofatov

(1.5) Allow NULLs in unique constraints and indices (SQL-99 spec).

Performance improvement

By N. Samofatov

(1.5) VIO undo log now uses B+ tree to store savepoint record data. It improves performance when doing multiple updates of record in a single transaction just a little (usually 2-3 orders of magnitude for 100000 records).

Improved EXECUTE STATEMENT

By A. Peshkoff

(1.5) Now it's possible to return values from the dynamic SQL using the INTO <variables> syntax.

Syntax:

```
EXECUTE STATEMENT <value> INTO <var_list>; (singleton form)
```

or

```
FOR EXECUTE STATEMENT <value>  
  INTO <var_list>  
  DO <stmt_list>;
```

Improved optimizer

By A. Brinkman

(1.5) Subselects in SET clause of UPDATE now can use indices.

(1.5) When an equal-node and other nodes (geq, leq, between...) are available for an index retrieval, then prefer the equal node.

(1.5) Better optimizations of "complex" JOIN queries (LEFT JOIN, views, SPs, etc).

Enhanced isc_database_info capability

By N. Samofatov

(1.5) List of currently active transactions is now available via isc_database_info call.

Performance improvement

By Mike Nordell

(1.5) Implemented shortcut boolean evaluation. Behaviour is controlled by "CompleteBooleanEvaluation" option of firebird.conf. Default is 0 (shortcut evaluation).

Performance improvement for IA32 CPU

By Mike Nordell

(1.5) Speed-up for index operations on IA32 CPU architecture.

Generic cleanup

By Blas Rodriguez Somoza, Erik Kunze

(1.5) Removed a lot of unused code.

Improved control when FW=OFF

By Blas Rodriguez Somoza

(1.5) Changed behaviour of the forced writes mode: now, if FW=off (disabled), you can control how often dirty pages are flushed on disk (allows better reliability when FW is disabled on Win32 platforms).

File Changes

By D. Yemanov

(1.5) The security database has been renamed to security.fdb and this version of Firebird has a new configuration file (firebird.conf).

New UDFs

By Juan Guerrero

(1.5) New user-defined functions LPAD and RPAD added to IB_UDF library.

Security connection cache

By D. Yemanov

(1.5) Connection to the security database is now cached, thus allowing to decrease time of subsequent database attachments.

Performance Improvements

By D. Yemanov

(1.5) 1. Reduce memory usage by the server.

(1.5) 2. Direct external I/O when the memory is not available for the sorting.

(1.5) 3. Increased number of streams and predicates supported by the optimizer.

New Character Sets

By Blas Rodriguez Somoza

(1.5) New character sets: DOS737, DOS775, DOS858, DOS862, DOS864, DOS866, DOS869, WIN1255, WIN1256, WIN1257, ISO8859_3, ISO8859_4, ISO8859_5, ISO8859_6, ISO8859_7, ISO8859_8, ISO8859_9, ISO8859_13. No collations are available yet for these charsets.

CREATE VIEW changes

By D. Yemanov

(1.5) Disallowed PLAN subclause.

Changed aggregate tracking behavior

By A. Brinkman

(1.5) Introduced backwards compatibility within aggregates. Deepest field inside aggregate determines where an aggregate-context should belong.

New API functions

By D. Yemanov

(1.5) IB7-compliant functions to return version of the client library--isc_get_client_version(), isc_get_client_major_version(), isc_get_client_minor_version()

Sort/merge improvement

By D. Yemanov

(1.5) Merging (SORT MERGE plans) is now done via in-memory sorting module.

Performance improvement

By N. Samofatov

(1.5) New memory manager's internals have been changed to give us better performance.

Win32 build changes

By D. Yemanov

(1.5) Changes.-

1. Changed names of USER32 objects to allow the server to run simultaneously with IB/FB1.
2. Map name for local (IPC) protocol is changed, so v1.5 client library is no longer compatible with the previous versions via IPC.
3. All transport protocol names (INET port and service, WNET pipe, IPC map) are now configurable via firebird.conf.

Triggers improvement

By D. Yemanov

(1.5) Added runtime action checks (INSERTING/UPDATING/DELETING predicates).

Example

```
if (INSERTING) then
    new.OPER_TYPE = 'I';
else
    new.OPER_TYPE = 'U';
```

SELECT Improvement

By N. Samofatov

(1.5) Allowed arbitrary expressions in the ORDER BY clause.

Generic code cleanup

By A. Peshkoff, N. Samofatov

(1.5) Cleaned up structures within Y-valve.

Implemented '--' Comments

By D. Yemanov

(1.5) Single-line comments (--) are now allowed in any position of the SQL statement.

FIRST/SKIP and ORDER BY changes

By D. Yemanov

(1.5) Changes --

1. Implemented ORDER BY clause in subqueries.
2. Disallowed FIRST/SKIP for views.
3. Allowed zero as valid argument for FIRST.

Improved optimizer

By A. Brinkman

(1.5) Improvement: let subqueries also use indices when their parent is a stored procedure.

Limitation lifted

By D. Yemanov

(1.5) Removed request size limitation.

Nulls first/last

By N. Samofatov

(1.5) Nulls first/last options for indexes.

Collations in UNIONs

By N. Samofatov

(1.5) Collation handling in "order by" clause of unions.

Generic code cleanup

By Erik Kunze, Ignacio J. Ortega, D. Sibiryakov

(1.5) Renamings, new safe macros, support for mingw.

Explicit record locking

By N. Samofatov

(1.5) Explicit record locking implementation finalized. Should be stable and consistent now.

Improved optimizer

By A. Brinkman

(1.5) Improvement: better handling of AND nodes inside an OR node.

Improved optimizer

By D. Yemanov

(1.5) If a few indices with much different selectivity could be used for index retrieval, only better of them are used while others are ignored.

Classic Server for Win32

By D. Yemanov

(1.5) CS architecture is now supported on Win32, but it still cannot be considered stable, so any feedback is welcome.

Code Cleanup

By N. Samofatov

(1.5) Stored procedures are no longer recompiled before deletion.

New WIN1251-UA Collation

By D. Yemanov

(1.5) New collation for WIN1251 charset: WIN1251-UA for both Ukrainian and Russian languages.

Client library change

By D. Yemanov

(1.5) API routines are no longer exported by ordinals.

New configuration manager

By D. Yemanov

(1.5) Enable the same plain file based configuration for all supported platforms.

Improved optimizer

By A. Brinkman

(1.5) Added better support for using indices with "OR". Pick the best available compound index from all "AND" nodes.

Added support for detecting use of index with sub-selects in aggregate select.

Explicit Savepoints

By N. Samofatov

(1.5) Added support for explicit savepoint management in DSQL.

Protocol cleanup

By Sean Leyne

(1.5) IPX/SPX network protocol is no longer supported.

Obsolete platforms cleanup

By Sean Leyne

(1.5) Some platform are no longer supported by the current source code. DELTA, IMP, DG_X86, M88K, UNIXWARE, Ultrix, NeXT, ALPHA_NT, DGUX, MPE/XL, DecOSF, SGI, HP700, Netware, MSDOS, SUN3_3.

Improved thread scheduler for Win32 SS

By A. Peshkoff

(1.5) : now the server should be more responsive under heavy load.

Explicit Locking

By N. Samofatov

(1.5) Added support for explicit locking. Wait behavior in isc_tpb_wait transaction modes is not stable yet.

Syntax

```
SELECT <...>
  [FOR UPDATE [OF col [, col ...]]
  [WITH LOCK]]
```

Generic cleanup

By Erik Kunze

(1.5) ISC_STATUS_LENGTH and MAXPATHLEN macros.

Empty BEGIN...END blocks

By D. Yemanov

(1.5) PSQL: enabled support for empty BEGIN...END blocks.

FR # 437859

By D. Yemanov

(1.5) Implemented execute procedure and string concat, allowing any expression to be used as a SP parameter.

FR # 562417

By D. Yemanov

(1.5) Aggregate concatenated empty char.

FR # 451927

By D. Yemanov

(1.5) New ROWS_AFFECTED system variable in PSQL: return number of rows affected by the last INSERT/UPDATE/DELETE statement. For any other statement than INSERT/UPDATE/DELETE, result is always zero.

FR # 446240

By D. Yemanov

(1.5) Dynamic exception messages: allow to throw an exception with a message different to the one the exception was created with.

Syntax

```
EXCEPTION name [value];
```

FR # 547383

By D. Yemanov

(1.5) New SQLCODE and GDSCODE system variables providing access to the code of the caught error within the WHEN-block in PSQL. Outside WHEN-block, returns 0 (success).

Exception re-initiate semantics: allows an already caught exception in PSQL to be re-thrown from the WHEN-block.

Syntax

EXCEPTION;

No effect outside WHEN-block.

New NULL order handling

By N. Samofatov

(1.5) Allow user-defined ordering of NULLs.

New Registry key is used on Win32

(1.5) SOFTWARE\FirebirdSQL\Firebird

FR # 451925

By D. Yemanov

(1.5) User-defined constraint index names: allows name of an index enforcing a constraint to be either the same as the constraint name or any user-defined name.

New RECREATE VIEW statement

By D. Yemanov

(1.5) New RECREATE VIEW statement, shorthand for DROP VIEW / CREATE VIEW coupling of statements.

Syntax

```
RECREATE VIEW name <view_definition>;
```

By various authors

(1.5) Renamed distribution files to make sure we're Firebird. Now they're fbserver, fbclient, firebird.msg etc. The client library is fbclient now and it should be used in all new FB-based projects.

Minor ODS upgrade (ODS 10.1)

By D. Yemanov, N. Samofatov

(1.5) Added new system indices (RDB\$INDEX_41, RDB\$INDEX_42, RDB\$INDEX_43), now ODS version is 10.1.

FR # 451935

By D. Yemanov

(1.5) New CREATE OR ALTER statement for triggers and stored procedures, allows creating or altering a database object according to whether it exists or not.

Syntax

```
CREATE OR ALTER name <object_definition>;
```

Enhanced declaration of local variables in PSQL

By Claudio Valderrama

(1.5) Simplify syntax and allow declaring and defining variable at the same time.

Syntax

```
DECLARE [VARIABLE] name <variable_type> [{ '=' | DEFAULT } value];
```

Example

```
DECLARE my_var INTEGER = 123;
```

FR ## 555839, 546274

By A. Brinkman

(1.5) Enhanced grouping: allow to GROUP BY internal functions and subqueries. Also allow to GROUP BY ordinal (i.e. column position, a.k.a degree of column in output set).

FR # 451917

By A. Brinkman

(1.5) New COALESCE internal function allowing a column value to be calculated by a number of expressions, the first expression returning a non NULL value is returned as the column value.

FR # 451917

By A. Brinkman

(1.5) New NULLIF internal function returns NULL for a sub-expression if it has a specific value, otherwise returns the value of the sub-expression.

FR # 451917

By A. Brinkman

(1.5) New CASE internal function allows the result of a column to be determined by the results of a case expression.

Readline

By M. O'Donohue

(1.5) Command history retrieval (like Unix readline) added to ISQL.

FR # 446206

By D. Yemanov

(1.5) New BIGINT datatype allowing native SQL usage of 64-bit exact numerics (Dialect 3 only).

FR # 451922

By D. Yemanov

(1.5) Universal triggers allowing one trigger to be fired for a number of action types.

FR ## 446238, 446243

By D. Yemanov

(1.5) New CONNECTION_ID and TRANSACTION_ID system available in PSQL. Return appropriate internal identifier stored on the database header page.

FR # 446180

By D. Yemanov

(1.5) Server-side database aliases: attach to any database using an "alias" name instead of its physical pathname. The list of known database aliases is stored in aliases.conf file under the server installation root.

Example

Alias entry in the configuration file:

```
my_database = c:\dbs\my\database.gdb
```

Connection string in application:

```
localhost:my_database
```

Plugin Manager

By John Bellardo

(1.5) New plugin manager and INTL interface.

In-memory sorting

By D. Yemanov

(1.5) If SORT plan is used for a SQL statement, the sorting is done in memory. If there's not enough memory for this operation, reverts to old method using temporary file.

FR # 446256

By A. Peshkoff

(1.5) New EXECUTE STATEMENT PSQL extension statement allows execution of dynamic SQL statements in SPs/triggers.

Cleanup

By Sean Leyne, Erik Kunze

(1.5) Major code cleanup.

Memory Optimisation

By John Bellardo

(1.5) New memory manager.

Exception handling

By Mike Nordell, John Bellardo

(1.5) New exception handling logic.

autoconf build

By John Bellardo, M. O'Donohue, Erik Kunze

(1.5) New autoconf-based build configuration.

Codebase Conversion

By

(1.5) The code port from C to C++.

Release 1.5, 1.5.1 and 1.5.2 Bugs Fixed in v.1.5.3

Bug # 1242982

fixed by A. Brinkman

Compound index key mangling exhibited a bug. Now fixed.

Not registered

fixed by A. Brinkman

Unnecessary evaluation was being performed on the last argument of the COALESCE function.

Bug # 1016969

fixed by A. Brinkman

Using search parameters in a SUM() operation would return incorrect results

Bug # 1016969

fixed by A. Brinkman

UPDATE with a CASE expression involving parameters would throw an SQLCode -804 exception ("Data type unknown").

Not registered

fixed by A. Brinkman

COALESCE/CASE displayed a bug regarding BLOB sub-type.

Not registered

fixed by V. Horsun

The wrong error was being detected when a write failure occurred. (v.1.5 Regression.) Fix was back-ported from Firebird 2 HEAD.

Bug # 1106825

fixed by D. Yemanov

An access violation could occur in fcblient.dll v1.5.2 on disconnecting. (v.1.5.2 Regression.) Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by D. Yemanov

Generators were being initialized with garbage values on restoring from a metadata-only backup. (v.1.5.1 regression.) Fix backported from Firebird 2 HEAD.

Not registered

fixed by D. Yemanov

User savepoints were not being released when commit retaining was issued. Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by C. Valderrama

Back-ported some isql fixes from Firebird 2 HEAD:

1. Another fix for the -b (Bail On Error) option when SQL commands are issued and no database connection yet existed.
2. Applied Miroslav Penchev's patch for a bug discovered by Ivan Prenosil, where the -Q switch would always return 1 to the operating system.
3. Fixed a conflict between single-line and block comments.

Not registered

fixed by D. Yemanov

If a table contained a computed column of BLOB or ARRAY type, the first column of the table could be zeroed during a restore. Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by C. Valderrama, D. Yemanov

The server would lock up if a request to attach to security.fdb was unsuccessful. (v.1.5 regression.) Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by C. Valderrama

The fbudf function AddMonth() exhibited wrong behaviour when facing January. Fix back-ported from Firebird 2 HEAD.

Bug # 1124720

fixed by A. Peshkoff

The FOR EXECUTE STATEMENT ... DO SUSPEND construct in PSQL was exhibiting problems. Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by N. Samofatov

Bugchecks were being exhibited on AMD64 (and possibly other platforms) when a database was copied, rather than migrated using backup/restore. Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by A. Peshkoff

There were issues with descending indices used in referential constraints. Fix back-ported from Firebird 2 HEAD.

Bug # 1242982

fixed by A. Brinkman

An equality search on the first segment of a compound index, if it was an integer type, would result in redundant additional scans on specific values (2^n, e.g. 131072). Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by D. Yemanov

Comparisons between strings in character set NONE and another character set would cause an error. (V. 1.5.2 regression.)

Release 1.5 and 1.5.1 Bugs Fixed in v.1.5.2

Not registered

fixed by V. Horsun

When a SEGV error (or other asynchronous exception) is thrown from a badly written UDF, the server should log its name. This feature was broken in FB 1.5. Restored in v.1.5.2

Not registered

fixed by V. Horsun, N. Samofatov

In v.1.5.0, exit(3) was called on critical errors on Windows, precluding the use of the JIT debugger to analyse problems in UDF routines. A fix in v.1.5.1 caused the debugger to be called always, introducing a problem with automatic restarts of the server.

V.1.5.2 now calls the debugger only if you set the *BugcheckAbort* configuration file option to 1. The main goal of the v1.5.2 fix is to avoid Dr.Watson showing its close-or-debug message box.

Not registered

fixed by C. Valderrama

Support for multi-dimensional array fields was broken v1.5.1. Restored in v.1.5.2

Not registered

fixed by V. Horsun, N. Samofatov

Plans for selectable stored procedures containing multiple FOR loops were being reported in reverse order, compared with v.1.5.0. Restored in v.1.5.2

Not registered

fixed by A. Brinkman

In v.1.5.1, a bug in the optimiser caused unnecessary fetches in joined relations when "OR" was used on the base relation. Now fixed

Not registered

fixed by N. Samofatov

There was a problem with deadlock detection when pessimistic locking (WITH LOCK syntax) was used.

```
create table test (id integer);
insert into test values(1);
insert into test values(2);
Commit;
```

Transaction 1 (READ COMMITTED, WAIT):

```
select * from test
where id = 1 with lock;
```

Transaction 2 (READ COMMITTED, WAIT):

```
select * from test
where id = 2 with lock;
select * from test
where id = 1 with lock;
```

Transaction 1:

```
select * from test
where id = 2 with lock;
```

This set of conditions would result in a permanent deadlock. V.1.5.2 detects and reports such a deadlock as an error.

Not registered

fixed by A. Peshkoff

The server would crash when NULL was passed to EXECUTE STATEMENT ... INTO. For example,

```
...
VAR = NULL;
EXECUTE STATEMENT :VAR;
...
```

caused the server to die. Now fixed

Not registered

fixed by H. Borrie, D. Yemanov

Documentation for Windows Embedded Server, README.user.embedded in the /doc subdir of the Windows installation, section 2.2 "Database Access" was "corrected" wrongly in v.1.5.1. This section was unclear in v.1.5.0. In v.1.5.1 it was plain wrong. It is now correct.

Release 1.5 Bugs Fixed in v.1.5.1

Not registered

fixed by Alex Peshkoff

(1.5.1) Modification of /etc/init.d/firebird during Linux installation now works properly.

Bug # 750659

fixed by Nickolay Samofatov

(1.5.1) The wrong soname for libib_util.so caused warnings in ldconfig and also meant that it could not be loaded automatically by external function libraries using calls to ib_util_malloc(). That made such libraries unusable, unless a PRELOAD environment setting was added for fbserv-

er/fb_inet_server.

Not registered

fixed by Nickolay Samofatov

(1.5.1) Memory corruptions in the embedded DSQL API implementation were corrected.

Not registered

fixed by Arno Brinkman

(1.5.1) The following two optimizer bugs were fixed:

1. IS NULL is now handled correctly when a view is used in an outer join.
2. MERGE is used instead of JOIN when indices are not applicable for a join.

Not registered

fixed by Dmitry Sibiryakov

(1.5.1) 'Extract' bug in ISQL was fixed: previously, procedure parameters would be extracted incorrectly.

Not registered

fixed by Arno Brinkman

(1.5.1) Mapping of COUNT(*) in a HAVING clause when used with IN, ANY/SOME, ALL has been corrected.

Bug # 918653

fixed by Nickolay Samofatov

(1.5.1) NULLS FIRST ordering parameter now works in union queries ().

Not registered

fixed by Nickolay Samofatov

(1.5.1) The algorithm for dependency tracking that was introduced into gbak in v.1.5, made restoring a database with many interdependent procedures too slow. It has been replaced by a faster algorithm.

Not registered

fixed by Alex Peshkoff

(1.5.1) Fixed a memory leak in EXECUTE STATEMENT...INTO.

Not registered

fixed by Nickolay Samofatov

(1.5.1) A fresh implementation of the editline facility in isql was done to resolve a problem with isql crashing on some platforms.

Not registered

fixed by Paul Reeves

(1.5.1) A serious memory leak was fixed in the Windows control panel applet. If the applet window was left open for several hours it would eventually trigger a low memory condition on the

server.

Not registered

fixed by Paul Reeves

(1.5.1) Switching the server status via the control panel applet from certain configurations would lead to inconsistencies on Win XP and Win 2003. There were timing issues that were only apparent on these platforms. The problem was fixed.

Not registered

fixed by Alex Peshkoff

(1.5.1) On Windows, server startup was failing to look for the FIREBIRD environment variable when trying to locate the root directory. Fixed.

Not registered

fixed by Helen Borrie

(1.5.1) DOC for Embedded Server (README_embedded.txt) was updated slightly to reduce confusion about the client connection string format.

Old Bugs Fixed

The following list consists of bugfixes that have been done since the release of Firebird 1.0.

"SF" bug items refer to the Firebird bugtracker on SourceForce. "QC" bugs refer to the InterBase section of the bug tracker on Borland's Quality Central.

Bug # 1175157

fixed by V. Horsun

(1.5.3) An error in the thread scheduler would cause the server to lock up if an I/O error occurred at database attachment.

Not registered

fixed by A. Peshkoff

The `isc_user_*` (add/modify/delete) functions worked wrongly under Administrator account on Win32.

(1.5.3) At some point during InterBase development, the intention was to make the Win32 implementation work so that Superuser privileges on Unix were emulated for Administrator on Win32 (SuperUser/root on Unix/Linux can log into the security database without entering any username and password). It worked, up to a point.

However, if the Win32 Administrator user tried to call these functions through a SYSDBA login, the connection would hang with some strange path resolution errors.

The code for a root-style login has been disabled in the Win32 clients (`fbclient.dll` and `fbembed.dll`). Now any Windows user, including Administrator, must supply the SYSDBA user name and password.

Not registered

fixed by A. Peshkoff

(1.5.3) Denial-of-Service vulnerability: an extra-long database file name could crash the server. Endemic security bug. One of many overflow vulnerabilities fixed in the Firebird 2 code, this fix has been backported to v.1.5.3.

Not registered

fixed by A. Peshkoff

(1.5.3) The server would crash during some DDL operations. Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by C. Valderrama

(1.5.3) In some cases, a BLOB filter declaration would cause the server to crash. Fix back-ported from Firebird 2 HEAD.

Bug # 739480

fixed by A. Peshkov

(1.5.3) A locally exploitable stack overflow vulnerability was detected and fixed.

Not registered

fixed by D. Yemanov

(1.5.3) The server could crash while performing a metadata scan for a complex table. Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by D. Yemanov

(1.5.3) Database corruption could occur due to allowing certain pre-trigger actions, such as deleting a record in a BEFORE UPDATE trigger. Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by V. Horsun

(1.5.3) CPU load would rise to 100% when a write failure occurred. Fix back-ported from Firebird 2 HEAD.

Bug # 1076858

fixed by V. Horsun

(1.5.3) A source of possible corruption was exhibiting in the Classic server as a page type exception "page 0 is of wrong type (expected 6, found 1)". Identified, fixed and back-ported from Firebird 2 HEAD.

Not registered

fixed by V. Horsun

(1.5.3) When the gfix service code tried to reattach to a database that had become unavailable, the server would crash. An endless loop would occur due to the inability of the service to interact with the end user, causing the service buffer to overflow eventually and result in the crash. Fixed and back-ported from Firebird 2 HEAD.

Not registered

fixed by D. Yemanov

(1.5.3) Character set and/or collation specified for local variables in PSQL would get lost, potentially causing string conversion errors. Fix backported from Firebird 2 HEAD.

Not registered

fixed by D. Yemanov

(1.5.3) If any DDL operations were active during a database shutdown, the server would crash. Fix back-ported from Firebird 2 HEAD.

Bug # 1110717

fixed by D. Yemanov

(1.5.3) Subqueries in in VIEWS were returning character data with the wrong character set. Fix back-ported from Firebird 2 HEAD.

Not registered

fixed by C. Valderrama

(1.5.3) There was an issue with quoted identifiers in the ISQL command SHOW GENERATORS in Dialect 3.

Not registered

fixed by A. Harrison

(1.5.3) No more than 32767 identifiers were being generated by GPRE.

Not registered

fixed by A. Peshkoff

(1.5.3) Wrong permissions were installed for the QLI help database.

Bug # 1045970

fixed by D.Yemanov

(1.5.2) An old legacy bug that has continued to bug us is that, when a client had some events registered and its network connection had been terminated abnormally (hardware failure, reset button or task manager), then the server would start using 100% of the CPU time until the "parent" port (client connection which called `isc_que_events()` API routine) reported on its failure.

This bug affected all FB versions (more or less, depending on the *DummyPacketInterval* configuration option) and only TCP/IP connections.

Further work has been done to rectify the problem in v.1.5.2. It now appears to be solved.

Bug # 544132

fixed by C. Valderrama

(1.5.2) UDF with NULL input parameter, problem fixed.

Bug # 728839

fixed by C. Valderrama

(1.5.2) Left join would defeat a UDF by mangling a null descriptor. Fixed.

Not registered

fixed by D. Yemanov

(1.5.2) Error trying to delete from a naturally updatable view containing computed expressions. If you had a view like this:

```
create view v_test (f, s)
as
  select f1, f2 + f3
  from t
```

then the server returned the "attempted update of read-only column" error when you tried to perform a DELETE operation. This is fixed in FB 1.5.2 and above.

Not registered

fixed by A. Peshkoff

(1.5.2) Numeric data types represented by floating-point variables were being processed incorrectly in an EXECUTE STATEMENT with dialect 1 databases--numerics were being scaled incorrectly:

```
create table a (b numeric(18,3));
commit;
insert into a values(12345.678);
commit;
set term ^;
create procedure c
  returns(d numeric(18,3))
as
begin
  for execute statement 'select b from a'
  into :d
  do suspend;
end ^
commit^
set term ;^
select * from c;
```

returned 12.346 instead of 12345.678. Now fixed.

Not registered

fixed by V. Horsun

If DISTINCT was used in an aggregate function and the record set being processed (aggregated) was empty, then we had a small memory leak. This memory was not returned until disconnect.

This routine would eat 120MB on FB 1.5.1 and previous:

```
CREATE PROCEDURE MEM_LEAK
AS
  DECLARE I INT = 1000;
  DECLARE T INT;
  DECLARE C INT;
BEGIN
  WHILE (I > 0) DO
  BEGIN
    SELECT RDB$INDEX_TYPE,
    COUNT(DISTINCT RDB$RELATION_NAME)
    FROM RDB$INDICES
    WHERE 0=1
```

```
GROUP BY 1
INTO :T, :C;

I = I - 1;
END
END
```

It is fixed in v.1.5.2.

Not registered

fixed by D. Yemanov

(1.5.2) The server leaked resources when an exception was thrown from a selectable stored procedure. The procedural request wasn't freed properly and caused errors like "too many concurrent executions of the same request" after 750-1000 iterations.

```
CREATE PROCEDURE P (INP INTEGER)
RETURNS (OUTP INTEGER)
AS
BEGIN
    OUTP = INP / 0;
    SUSPEND;
END

UPDATE T SET ID = 1
WHERE (SELECT OUTP FROM P(1)) = 1
```

The leaking request blocks were returned on disconnect. Now fixed.

Not registered

fixed by P. Reeves

(1.5.2) The server log was polluted with SIGPIPE errors when running SuperServer on UNIX. The legacy InterBase code was logging sigpipe errors for SS running on *nix. Unfortunately sigpipe errors come in their thousands (when they come at all) with the result that the log filled up rather quickly. In extreme cases this led to filling up the entire partition.

Logging of SIGPIPE errors has thus been disabled.

Not registered

fixed by V. Horsun

(1.5.2) 100% CPU usage was exhibited by the cache_writer thread in some rare cases (reported by Adrianos dos Santos Fernandes).

To reproduce in v.1.5.1 or lower, open two command prompts.

Prompt 1:

```
isql
CREATE DATABASE 'test.fdb';
CREATE TABLE T (N INTEGER);
EXIT;

gbak -B test.fdb test.fbk
del test.fdb
gbak -C test.fbk test.fdb
```


Prompt 2:

```
gbak -B test.fdb test.fbk
```

The server would consume 99% of CPU until the isql prompt 2 was disconnected. The bug didn't occur when passing `-GARBAGE_COLLECT` in the last command.

Not registered

fixed by A. Karyakin, D. Yemanov

(1.5.2) A possible source of server crash was discovered in the `op_connect` handler. When a TCP/IP packet lacking user information was received on the server port, the server could crash. Because it was the first packet (`op_connect`) in the client-server protocol, it exposed the server to any kind of DoS attack. Anyone could kill the server with just one TCP packet. Now fixed.

Not registered

fixed by D. Yemanov

(1.5.2) The server could crash with complex queries where lots of streams were used in a sort/merge. A complex union with many aggregations and merge joins could crash the server because of a streams buffer overflow. Although the current limit is 255 streams per request, the temporary buffer could accommodate only 128 items. Now fixed

Not registered

fixed by C. Waters, D. Yemanov

(1.5.2) The server was blocking when events were used with Network Access Transaction (NAT) gateways. Auxiliary connections (for events) were established by the client library using the server-reported TCP/IP address. But the address returned by the server may be incorrect if it is behind a NAT box.

The fix was to use the address that was used to connect the main socket, not the address reported by the server.

Not registered

fixed by V. Horsun

(1.5.2) Sweeper would not release its lock when database shutdown was executed. A server crash could occur when a database shutdown was initiated while the sweep is being in progress. Now fixed.

Not registered

fixed by V. Horsun

(1.5.2) The least significant bits of a floating-point value would be lost when rounding the value to an integer or `int64` value.

Dialect 3 database:

```
SELECT CAST(CAST( 1.005E0 AS NUMERIC(15,2))
  AS VARCHAR(30)) FROM RDB$DATABASE
UNION ALL
SELECT CAST(CAST( 1.015E0 AS NUMERIC(15,2))
```

```
AS VARCHAR(30)) FROM RDB$DATABASE
UNION ALL
SELECT CAST(CAST( 1.025E0 AS NUMERIC(15,2))
AS VARCHAR(30)) FROM RDB$DATABASE
UNION ALL
SELECT CAST(CAST( 1.035E0 AS NUMERIC(15,2))
AS VARCHAR(30)) FROM RDB$DATABASE
UNION ALL
SELECT CAST(CAST( 1.045E0 AS NUMERIC(15,2))
AS VARCHAR(30)) FROM RDB$DATABASE
UNION ALL
SELECT CAST(CAST( 1.055E0 AS NUMERIC(15,2))
AS VARCHAR(30)) FROM RDB$DATABASE
UNION ALL
SELECT CAST(CAST( 1.065E0 AS NUMERIC(15,2))
AS VARCHAR(30)) FROM RDB$DATABASE
UNION ALL
SELECT CAST(CAST( 1.075E0 AS NUMERIC(15,2))
AS VARCHAR(30)) FROM RDB$DATABASE
UNION ALL
SELECT CAST(CAST( 1.085E0 AS NUMERIC(15,2))
AS VARCHAR(30)) FROM RDB$DATABASE
UNION ALL
SELECT CAST(CAST( 1.095E0 AS NUMERIC(15,2))
AS VARCHAR(30)) FROM RDB$DATABASE
```

FB 1.5.1 returns

```
F_1
----
1.00
1.01
1.03
1.04
1.05
1.05
1.06
1.08
1.09
1.10
```

The problem is fixed. FB 1.5.2 returns

```
F_1
----
1.01
1.02
1.03
1.04
1.05
1.06
1.07
1.08
1.09
1.10
```

Not registered

fixed by N. Samofatov

(1.5.2) A few memory access problems were detected when testing HEAD under Valgrind. Fixed

in HEAD and back-ported to v.1.5.2

64-bit build issues

Experimental 64-bit builds by N. Samofatov

(1.5.2) 64-bit SuperServer builds on platforms such as Linux/AMD64/NPTL, which use the high-order bits of a 64-bit thread ID, were exhibiting run-time errors.

CURRENT_TIMESTAMP was yielding unpredictable results on 64-bit platforms.

Warning

The v.1.5.2 64-bit builds were subsequently withdrawn, due to database corruptions caused by data alignment bugs and possibly others.

Events bug

fixed by Dmitry Yemanov

(1.5.1) Fixed a persistent issue with events, where a client using events would cause the server's CPU usage to go up to 100 percent when it terminated its connection. The problem existed in all Firebird versions, but became more acute in v. 1.5 because it has the default dummy packet interval at 0 and relies instead on TCP/IP keepalive packets.

Now, the client library tries to wake up the server's port when detaching, allowing the server to scan for broken connections and close them sooner.

Not registered

fixed by Vladimir Tsvigun

(1.5.1) Fixed a bug in the QLI utility, whereby assigning a variable would generate an access violation.

gbak bug (1)

fixed by Claudio Valderrama

(1.5.1) The -n[o_validity], that should make gbak restore a database without restoring validity constraints, was ignoring NOT NULL constraints and restoring them regardless. Now it works as it ought to.

gbak bug (2) SF # 750659

fixed by Claudio Valderrama

(1.5.1) When a fresh database needs to be created by metadata-backup/restore, the user-defined generators should be set to zero. However, during a metadata-only backup, gbak would preserve the values of all generators, including non-system ones. Now, they are correctly reset.

Not registered

fixed by Arno Brinkman

(1.5.1) Corrected a bug that prevented proper restoration of inter-dependent views.

Memory problems (1)

fixed by Dmitry Yemanov

(1.5.1) A memory leak in the Services API.

Memory problems (2)

fixed by Nickolay Samofatov

(1.5.1) A memory leak in `isc_database_info()`.

Memory problems (3) QC # 7496

fixed by Oleg Loa

(1.5.1) Memory leaks occurred in PSQL modules when aggregation errors were handled in a WHEN block.

Memory problems (4) SF # 919246

fixed by Nickolay Samofatov

(1.5.1) A stability problem with usage of previously freed memory in the client library was found and eliminated. It was present in InterBase and earlier Firebird versions but it re-surfaced in Firebird 1.5, due to the memory manager actively returning memory to the system. When using TCP/IP with Windows clients, application programs would hang because of an access violation in `fbclient.dll`, or `fbclient.dll` itself would hang. In some cases, further connections to the same database would be rejected.

Generator bugs (1)

fixed by Nickolay Samofatov

(1.5.1) Attempting to update a generator in a read-only database would cause database detach to be prolonged and such a situation would leave the database locked to other connections. Fixed the broken CCH finalization sequence and a page lock leak.

Generator bugs (2)

fixed by Oleg Loa, Vlad Horsun, Dmitry Yemanov

(1.5.1) Generator pages were not flushed properly, causing generator value changes to be lost after a server failure, even in Forced Writes mode

Old tracking logic bug fixed, SF ## 627057, 922602

fixed by ?

(1.5.1) A nested query that contained a variable in the WHERE clause would be erroneously flagged as invariant (), e.g.

```
select max(d2.id) from demo d2
  where d2.id < :id
```

Not registered

fixed by Nickolay Samofatov

(1.5.1) Request cloning logic was broken. Clones of procedures/triggers were not accounting for invariants and the requests opened by them at all. Thus, they were inheriting invariant values from previous executions and were not freeing resources (highly limited! - you can have only 1000 copies of a request open). This is why most recursive procedures didn't work at all and using procedures from multiple Superserver connections produced results that were inconsistent or timing-dependent.

Invariant dependency tracking was not working properly. The engine now keeps account of which variables an invariant depends on and clears the cached invariant value when values are being assigned to these variables.

Win32 lock manager bugs

fixed by Nickolay Samofatov

(1.5.1) A number of problems with the win32 lock manager, that manifested themselves as low performance when several operations were waiting on locks simultaneously, were solved.

SF bug # 784121

fixed by Claudio Valderrama

(1.5.1) Expression evaluation was not supported in LEFT JOIN.

Not registered

fixed by Dmitry Yemanov

(1.5.1) Server would crash when views and selectable procedures were intermixed.

Not registered

fixed by Nickolay Samofatov

(1.5.1) Fixed an obscure bug where the server would crash if LIKE ESCAPE <symbol> was asked to operate on NULL. LIKE syntax is:

```
<string> LIKE <string> [ESCAPE <one-char-string>]
```

A request such as

```
select * from rdb$database
  where 'a' like 'a' escape cast(null as char(1))
```

would crash the engine. It now complies with the SQL standard and returns an empty result set from such a query.

UDF bugs

fixed by Fred Polizo Jr

(1.5.1) Length of string types containing binary data (OCTETS) was being determined incorrectly in UDFs.

Unregistered bugs

fixed by P. Jacobi

(1.5) Fixed minor inconsistencies in charsets naming.

Unregistered bug

fixed by D. Yemanov

(1.5) GSTAT crashed in some switch combinations.

Unregistered bug

fixed by D. Yemanov

(1.5) Fixed broken savepoint handling in BREAK|LEAVE|EXIT.

Unregistered bug

fixed by A. Brinkman

(1.5) Fixed optimizer to prefer single indices to composite ones and to prefer full-match unique indices.

Bug # 721792

fixed by N. Samofatov

(1.5) Long running connection caused mem leak in OS kernel device

Bug # 775003

fixed by P. Vinkenoog, N. Samofatov

(1.5) UDF log(x, y) in fact returned log(y, x)

Bug # 774987

fixed by P. Vinkenoog, N. Samofatov

(1.5) UDFs ltrim("") and rtrim("") returned NULL; rtrim forgot first char

Unregistered bug

fixed by A. Peshkoff

(1.5) Fixed server crash caused by lost transaction context.

Unregistered bug

fixed by A. Peshkoff

(1.5) Fixed server crash caused by any combination of a subquery and BETWEEN.

Bug # 736318

fixed by D. Yemanov

(1.5) "<value> STARTING WITH <field>" would fail when using indices.

fixed by A. Peshkoff

(1.5) Non-existent deadlock is raised after execution of pre-(update/delete) triggers.

Unregistered bug

fixed by D. Yemanov

(1.5) Preparing some big queries could crash the server randomly.

Unregistered bug

fixed by Oleg Loa

(1.5) UDF arguments of types DATE/TIME were wrong in dialect 3.

Unregistered bug

fixed by Vlad Horsun, D. Yemanov

(1.5) Possible referential integrity violation.

Bug # 745090 & other beta installation issues

fixed by Erik S. LaBianca, N. Samofatov

(1.5) Permissions problem for firebird.conf. Also generate aliases.conf on install; use rpmbuild to create Linux packages

Unregistered bug

fixed by A. Peshkoff

(1.5) Wrong attachment reference after exception in PSQL.

Unregistered bug

fixed by Vlad Horsun, D. Yemanov

(1.5) Potential index corruptions during garbage collection.

Unregistered bug

fixed by N. Samofatov

(1.5) Solved problems with temporary files management:

1. Security hole on all POSIX platforms except FREEBSD/OPENBSD related to mktemp usage (possible DoS attacks or privileges elevation)
2. Only 27 unique filenames generated on win32 (which could cause unpredictable behavior in SS builds)

Unregistered bug/Event Manager change

fixed by D. Yemanov

(1.5) Disabled usage of specifically-configured auxiliary port in CS builds due to known issues.

Unregistered bug

fixed by D. Yemanov

(1.5) Potential crashes on disconnect when event notification was used.

Fixed Services API

fixed by N. Samofatov

(1.5) Enabled statistics Services API for POSIX CS builds.

Note

1. Appropriate changes in Win32 CS are not ready yet
2. Backup/restore service was fixed, tested and should work
3. Database validation was partially fixed and may work
4. Other services are probably non-functional in CS builds yet

Fixed Services API

fixed by D. Yemanov

(1.5) Partly enabled Services API for win32 CS builds.

Service Manager changes: Win32 Classic

Change done by D. Yemanov

(1.5) Features of GSTAT/GSEC are not available via Services API in win32 CS (until a later sub-release).

Unregistered bug

fixed by D. Yemanov

(1.5) Wrong record statistics were reported when an operation failed for some reason.

Unregistered bug

fixed by A. Peshkoff

(1.5) stdin/stdout could not be used to redirect console I/O in the Win32 build of GBAK.

Unregistered bug

fixed by N. Samofatov

(1.5) Lock table resizing in CS was broken. No more "lock manager out of room" or crashes (possible in all previous CS builds of Interbase/Firebird).

Bug # 625899

fixed by A. Peshkoff

(1.5) BUGCHECK(291): Possible database corruption when you modified/deleted the same record in pre-trigger for which this trigger was called.

Unregistered bug

fixed by Oleg Loa

(1.5) Buffer overrun in isc_database_info() call.

Unregistered bug

fixed by Jim Starkey, Paul Reeves

(1.5) Wrong type of event delivery (unnecessary usage of OOB packets).

Unregistered bug

fixed by A. Brinkman

(1.5) Server would crash in some Services API operations.

Unregistered bugs

fixed by Mike Nordell, A. Peshkoff, N. Samofatov, D. Yemanov

(1.5) Fixed resource/memory leaks.

Unregistered bug

fixed by D. Yemanov

(1.5) Buffer overrun with multidimensional arrays.

Bug # 213460, # 678718

fixed by D. Yemanov

(1.5) Various issues with events used on multihomed hosts.

Note

Now it's also possible to setup a definite port for event processing.

Unregistered bugs

fixed by Mike Nordell, A. Peshkoff

(1.5) Fixed some resource leaks.

Unregistered bug

fixed by N. Samofatov

(1.5) Potential for database corruption when backing out the savepoint after large number of DML operations (so transaction-level savepoint is dropped) and record was updated *not* under the savepoint and the deleted under the savepoint.

Unregistered bug

fixed by D. Yemanov

(1.5) Server would hang during disconnect after mass updates.

Unregistered bug

fixed by A. Brinkman

(1.5) "Context already in use" error was exhibited in the case of DISTINCT with subqueries.

Unregistered bug

fixed by A. Brinkman

(1.5) Long delays during connecting/disconnecting on WinXP.

Bug # 523589

fixed by A. Brinkman

(1.5) View was affecting the result of a query. Comment: Problem was that RSE's (inside a view) were not flagged as variant.

Unregistered bug

fixed by D. Yemanov

(1.5) Sometimes GFIX didn't allow to specify "-user" and "-password" switches ("incompatible swiches" error).

Bug # 508594

fixed by A. Brinkman

(1.5) LEFT JOIN with VIEWS: simple LEFT JOIN on a VIEW with only an ON clause didn't use an index even if it was possible.

Unregistered bug

fixed by D. Yemanov

(1.5) Trashed RDB\$FIELD_LENGTH for views that contain concatenation of long CHAR/VARCHAR fields.

Unregistered bug

fixed by D. Yemanov

(1.5) Cursors (WHERE CURRENT OF clause) could not be used in triggers.

Bug # 221921

fixed by A. Brinkman

(1.5) Case reported where ORDER BY had no effect.

Bug # 213859

fixed by A. Brinkman

(1.5) Bug case where subquery was used in connection with IN() clause.

Unregistered bug

fixed by A. Brinkman

(1.5) Engine crashed when UNIONS were used in a VIEW and that VIEW was used in the WHERE clause inside a subquery.

Unregistered bug

fixed by D. Yemanov

(1.5) "Request synchronization error" with BREAK statement.

Bug # 521952

fixed by A. Brinkman

(1.5) No current record for fetch operation.

Unregistered bug

fixed by D. Yemanov

(1.5) QLI didn't understand BIGINT datatype.

Unregistered bug

fixed by A. Brinkman

(1.5) Length of text variables inside procs/triggers wasn't copied to descriptor structure.

Unregistered bug

fixed by Erik Kunze

(1.5) Buffer overflow (MAXPATHLEN) and rewritten local function dirname.

Old but unregistered bug

fixed by N. Samofatov

(1.5) Make SQLDA parameter mapping consistent with order and number of parameters in source SQL string.

Note

You can enable older mapping behavior (for backward compatibility) using the "OldParameterOrdering" configuration manager parameter.

Unregistered bug

fixed by A. Peshkoff

(1.5) Exceptions inside for/while loops in triggers were not handled correctly.

Bug # 623992

fixed by Paul Reeves, Mark O'Donohue

(1.5) Double forward slash in connection string gave problems.

Unregistered bug

fixed by A. Peshkoff

(1.5) Deadlock occurred unexpectedly during some database operations.

Unregistered bug

fixed by N. Samofatov

(1.5) Quoted identifiers problem in plan expressions.

Bug # 558364

fixed by Ignacio J. Ortega

(1.5) Triggers fail to compile if PLAN used.

Unregistered bug

fixed by Vlad Horsun, Erik Kunze

(1.5) Distributed (2PC) transaction could not be properly rolled back due to network errors.

Bug # 496784

fixed by N. Samofatov

(1.5) When the optimizer finds indexes for LEFT JOIN, they work like INNER JOIN. Fixed problem which caused complex outer joins to produce wrong results.

Unregistered bug

fixed by D. Yemanov

(1.5) BLOB subtype was ignored in system domains generated for expression fields in views.

Unregistered bug

fixed by D. Yemanov

(1.5) Fixed a Windows installation bug: instreg.exe wasn't creating the "GuardianOptions" registry value.

Unregistered bug

fixed by N. Samofatov

(1.5) Resource leaks in DDL recursive procedure handling which caused some DDL to fail.

Unregistered bug

fixed by N. Samofatov

(1.5) CHECK constraint which uses only one table field is now dropped automatically when this field is dropped.

Unregistered bug

fixed by N. Samofatov

(1.5) The server could crash during garbage collection under heavy load.

Unregistered bug

fixed by D. Kuzmenko

(1.5) gstat showed wrong value for maxdup element.

Unregistered bug

fixed by D. Yemanov

(1.5) Trigger whose name starts with 'RDB\$' can not be altered or dropped at all.

Unregistered bug

fixed by D. Yemanov

(1.5) Broken dependencies (like DB\$34) would appear in the database after metadata changes.

Unregistered bug

fixed by D. Yemanov

(1.5) Disabled BREAK statement for triggers (like EXIT) due to known internal limitations.

Bug # 545725

fixed by A. Peshkoff

(1.5) Automatic/background sweep could hang.

Unregistered bug

fixed by D. Yemanov

(1.5) The server would crash when XSQLDA structures were not prepared for all statement parameters.

Bug # 567931

Partly fixed by D. Yemanov

(1.5) A metadata security hole.

Unregistered bug

fixed by Artem Petkevych

(1.5) BigInt arrays didn't work.

Bug # 538201

fixed by Claudio Valderrama

(1.5) Crash with extract from null as date.

Unspecific

fixed by N. Samofatov

(1.5) Deferred metadata compilation: solved a lot of causes of the well-known "object in use" error.

Known Issues at v.1.5.1

The following outstanding issues were known at sub-release 1.5.1.

Issue with SQL Plans

PLANS for selectable procedures having multiple FOR loops are reported in the wrong order.

Example (from QMDB test):

```
create procedure proc1
returns (a integer)
as
begin
  for select a from table1 into :a do
    suspend;
  for select b from table2 into :a do
    suspend;
end ^^
...
select * from proc1^^
```

1.5.1 produces the PLAN:

```
PLAN (TABLE2 NATURAL) (TABLE1 NATURAL)
```

instead of

```
PLAN (TABLE1 NATURAL) (TABLE2 NATURAL)
```

UPDATE :: This issue was fixed in release v.1.5.2.