Firebird™ Version 1.5

Release Notes v.1.5

31 October 2003

Contents

General Notes
New Features
Compatibility with Older Versions
Language Enhancements
- Data types
- Metadata
- DSQL
- PSQL
- Firebird 1.0.x

New Reserved Words
ISQL Features
External Functions (UDFs)
- in the ib_udf library
- in the fbudf library

New Configuration File—firebird.conf
- Filesystem-related parameters
- Resource-related
- Communications-related
-

firebird.conf parameters (continued)
- POSIX-specific
- Windows-specific
- Sort space
- Compatibility

DB File Aliasing
- Connecting using an aliased path
- Naming databases on Windows

Firebird Developer Teams
Installation Notes
- Windows 32-bit
- Linux/UNIX
- Solaris
- MacOS X
- FreeBSD

Further Information
Tools and Drivers
Documentation
Fixed Bugs

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 -
http://sourceforge.net/project/showfiles.php?group_id=9028

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

Please refer to the Documentation section for locations of recommended documentation.

New Features

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.

- 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.

Several important new language features have been added since version 1.0.x, including the SQL-92 case 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" 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 local system 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.

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.

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 SS 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.

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).
For compatibility with existing applications, a gateway library "gds32.dll" is included in the distribution kit. This library doesn't have any code but just redirects all calls to fbclient.dll. Hence, for old

applications, you must have both this version of gds32.dll and fbclient.dll in either the application program directory or OS system directory.

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

Platform	Module	Firebird 1.0	Firebird 1.5	Special notes
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.
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	
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	
Windows	Guardian	ibguard.exe	fbguard.exe	
Windows	Superserver binary	ibserver.exe	fbserver.exe	Not multi-processor capable.

Platform	Module	Firebird 1.0	Firebird 1.5	Special notes
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	Client library stub for compatibility	N/A	gds32.dll	
Windows	Configuration file	ibconfig	firebird.conf	
Windows	Local IPC port	InterBaseIPI	FirebirdIPI	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\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".

Compatibility

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

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.
The Operations Guide 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 File Names and Locations 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. 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 Db to 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 updated from 7.x	Some problems reported	Some problems reported
	8.0 clean install	Yes	Yes
Mandrake	8.x	No	No
	9.0	Yes	Yes, if force-patched with glibc-2.3.1-10mdk.i586.rpm
	9.1	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	Yes	Not known
	8.1	Yes	Yes

Language Enhancements

DATA TYPES

(1.5) New Native SQL Data Type

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

Example(s)
i)
```
DECLARE VARIABLE VAR1 BIGINT;
```
ii)
```
CREATE TABLE TABLE1 (FIELD1 BIGINT);
```

METADATA

(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 created named PRIMARY, FOREIGN KEY and UNIQUE constraints, the identifier of 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
a) a system-generated index to receive automatically the same identifier as the named constraint it enforces
b) 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
```
...
[ADD] CONSTRAINT [<constraint-identifier>]
<constraint-type> <constraint-definition>
[USING [ASC[ENDING] | DESC[ENDING]] INDEX <index_name>]
```

Caveat: 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

```
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.
Note(s):
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 contexts 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) 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 the NOT NULL constraint. This complies with SQL-99. Be cautious about using this if you plan to revert your database to Firebird 1.0.x or any InterBase version. The testing logic is as follows:

```
<unique constraint definition> ::=
<unique specification> ( <unique column list> )
<unique specification> ::= {UNIQUE | PRIMARY KEY}
```

1) If the <unique specification> specifies PRIMARY KEY, then let SC be the <search condition>:

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

ii) Otherwise, let SC be the <search condition>:

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

where UNIQUE means that if there are no two rows in (SELECT UCL FROM TN) such that the value of each column in one row is non-null and is not distinct from the value of the corresponding column in the other row, then the result is True; otherwise, the result is False.

The constraint allows existence of only those rows for which the aforementioned SC evaluates to True. It means that ***the PRIMARY KEY constraint doesn't allow NULLs*** whilst the UNIQUE constraint allows 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).

DSQL

(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

```
  <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
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.

Format

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

Syntax Rules

i) COALESCE (V1, V2) is equivalent to the following <case specification>:
   CASE WHEN V1 IS NOT NULL THEN V1 ELSE V2 END

ii) 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.

Format

```
<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.

```
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 identifer 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.

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 <identifer> 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. 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 a) extremely small (ideally, a singleton) and b) precisely controlled by the application code.

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

Syntax

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

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

The SELECT... WITH LOCK construct is available in DSQL and PSQL. It can succeed only in a top-level, single-table SELECT statement. It is not available in a subquery specification, nor for joined sets. It cannot be specified with the DISTINCT operator, a GROUP BY clause or any other aggregating operation. It cannot be used in or with a view, nor with an external table, nor with the output of a selectable stored procedure.

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.

TPB mode	***Behavior***
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. 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 transation 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.

When an UPDATE statement steps on 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.

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. 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 FOR UPDATE clause provides a technique to prevent usage of buffered fetches, optionally with the OF <column-names> to enable positioned updates. 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.

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

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.
NOTE: 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 ')'
                        ;
```

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.

Example
```
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

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.
The ordering of nulls in the result set can be controlled using the 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

```
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
```

PSQL (Stored procedure and trigger language)

(1.5) EXECUTE STATEMENT

Alex Peshkov

PSQL extension which takes a string which is a valid dynamic SQL statement and executes it as if it had been submitted to DSQL.
Available in triggers and stored procedures.

The syntax may have three forms.

Syntax 1
Executes <string> as 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 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))
RETURNING_VALUES (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
```

Additonal notes about 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 impossible, or perform even worse than EXECUTE STATEMENT.

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

CURRENT_CONNECTION

and

CURRENT_TRANSACTION

Each of these context variables returns the system identifier of the active connection or the current transaction context, respectively. Return type is INTEGER. Available in DSQL and PSQL. Because these values are stored on the database header page, they will be reset after a database restore.

Syntax

```
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

```
ROW_COUNT
```

Example

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

Note: this variable 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 ~~will~~ 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

```
SQLCODE
GDSCODE
```

Example

```
BEGIN
   ...
   WHEN SQLCODE -802 THEN
     EXCEPTION E_EXCEPTION_1;
   WHEN SQLCODE -803 THEN
     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

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 universal triggers (see METADATA, above).

Syntax

```
INSERTING
UPDATING
DELETING
```

Example

```
IF (INSERTING OR DELETING) THEN
  NEW.ID = 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
```
EXCEPTION <exception_name> <message_value>;
```

Examples
i)
```
EXCEPTION E_EXCEPTION_1 'Error!';
```
ii)
```
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
```
EXCEPTION;
```

Examples
i)
```
BEGIN
  ...
  WHEN SQLCODE -802 THEN
    EXCEPTION E_ARITH_EXCEPT;
  WHEN SQLCODE -802 THEN
    EXCEPTION E_KEY_VIOLATION;
  WHEN ANY THEN
    EXCEPTION;
END
```
ii)
```
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. The SQL-99 standard keyword LEAVE deprecates the existing BREAK. Available in triggers as well as stored procedures.

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

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

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

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

(1.5) SELECT FIRST can now take zero as argument

FB 1.5 allows zero as an argument of FIRST. An empty result set will be returned.

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

Parentheses are required for expression arguments and optional otherwise.

They 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.

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.

Available in SQL and DSQL except where otherwise indicated.

Examples:

```
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 correlated WHERE...IN (SELECT...) predicates 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 )
```

Character set enhancements

Added in 1.5

- Added WIN1251_UA collation (for both Russian and Ukrainian languages) for WIN1251 character set.
- Corrected default uppercasing for WIN1251
- Added ISO_HUN (for Hungarian language) for ISO8859_2 character set

New character sets (no non-binary collations) added

Blas Rodriguez 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, Pinyi, 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

Added in 1.0

- Added case insensitive Hungarian collation set, developed and tested by Sandor Szollosi (ssani@freemail.hu).

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

LANGUAGE EXTENSIONS CARRIED OVER FROM FIREBIRD 1.0.x

The following language extensions, introduced in Firebird 1.0.x, are reproduced here for your convenience.

(1.0) CURRENT_USER and CURRENT_ROLE

These two new context variables have been added to reference the USER and (if implemented[1]) the ROLE of the current connection context.

```
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 O 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. 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.

(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.

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

(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) 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.

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

Available in SQL and DSQL.

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

Internal function 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 option 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*> starts at 1, like other SQL commands.
Neither <*pos*> nor <*length*> can be query parameters.

Because <pos> and <length> are byte positions, the identifier 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.

Available in SQL and DSQL.

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

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

(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.
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
```

(1.0) New Comment marker

Claudio Valderrama

For use in scripts, DSQL, stored procedures and triggers.

Example
```
-- 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) Alter Trigger no longer increments the change count on table

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.

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

ISQL Features

“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.

User-defined Functions

In ib_udf

rpad (*instring, length, padcharacter***)**
Juan Guerrero

Right-pads the supplied string *instring* by appending *padcharacter*s 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 (*instring, length, padcharacter***)**
Juan Guerrero

Left-pads the supplied string *instring* by prepending *padcharacter*s 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 (*x, y***)**
Paul Vinkenoog

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

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. It should be noted 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 nonsense result or the fbudf code may perform an illegal operation.

New Configuration File – firebird.conf

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.

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

1) FIREBIRD environment variable
2) RootDirectory parameter in firebird.conf
3) Registry:
   HKLM\SOFTWARE\ SOFTWARE\Firebird Project\Firebird Server\Instances\DefaultInstance
   and looks for the field RootDirectory.
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 (/usr/local/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 (/usr/local/firebird)

Parameters

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

Filesystem-related parameters

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:

```
DatabaseFileAccess = Restrict userdata
```

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.

See the **CAUTION** below the next entry, UdfAccess.

UdfAccess

was *external_function_directory* in isc_config/ibconfig but syntax has changed.

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 files only if they are located in C:\Program Files\Firebird\userdata\ExternalModules:

```
ExternalFileAccess = 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 parameters

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.

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 installed 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:

$M_i = 2^I$
$A = M_1 + M_2 + M_3 . . .$

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

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

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

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 parameters

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.
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 here:

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_KEEPALIVE 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:

http://support.microsoft.com/default.aspx?kbid=140325

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

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 parameters

LockSignal

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

RemoteFileOpenAbility

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 on slow networks. 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 parameters

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 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.
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 withheld 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®.

Parameters for configuring temporary 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.
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 100000000;/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
```

NB No quoted paths as was required in Firebird 1.0

Compatibility parameters

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.

> 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 workarounds 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.

DB 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 database 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
```

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.

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 Rokytskyy	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; coordinator, Darwin builds; initial implementor of new memory model
Erik Kunze	Germany	Bug-finder and fixer; code-cleaner; SINIX-Z builds
Dmitry Sibiryakov	Russian Federation	Code cleanup; MinGW builds
Pavel Cisar	Czech Republic	Linux builds (Release 1.0); QA tools designer/coordinator
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

Developer	Country	Major tasks
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
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 Kukkuk Yakov Maryanov Christian Pradelli	Daniel Rail Volker Rehn David Ridgway David Rushby Pavel Shibanov Ruslan Strelba

Install Firebird 1.5 on Windows 32

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 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!

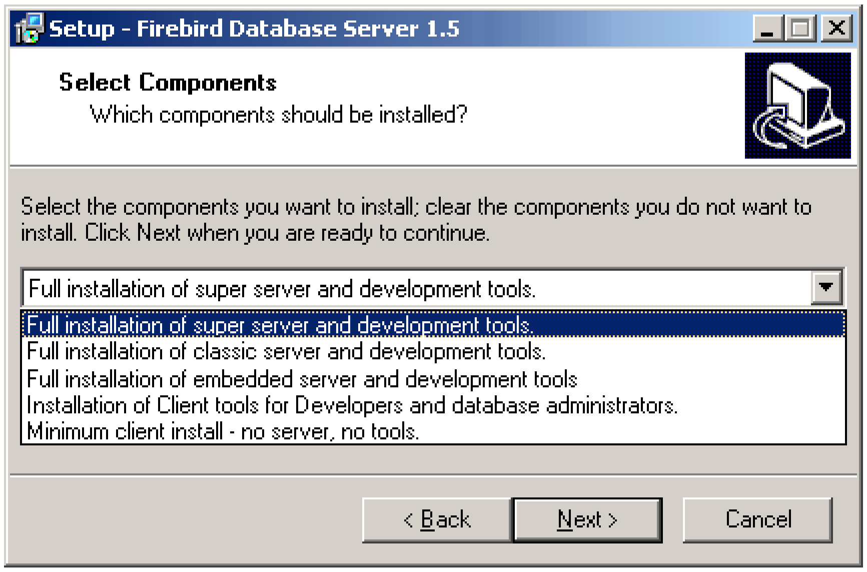
- The default root location of Firebird 1.5 will be C:\Program Files\Firebird. If your old version is already in a directory of that name and you want to use the 1.5 defaults, rename the existing directory

- 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

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.


Setup - Firebird Database Server 1.5

Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Full installation of super server and development tools.

Full installation of super server and development tools.
Full installation of classic server and development tools.
Full installation of embedded server and development tools
Installation of Client tools for Developers and database administrators.
Minimum client install - no server, no tools.

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 msvcp60.dll, or an old gds32.dll was already loaded when the installer started up.

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 <root>\bin (here and below <root> 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, run also instsvc.exe:
-

```
instsvc.exe install
```

- optionally, you should 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
```

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.

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 (fbembed.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 IB/FB 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 fbembed.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.

Open your firebird.conf and set RootDirectory to the root of this directory tree. (Don't forget to erase the "#" comment marker!)

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_utils.dll
  D:\my_app\gbak.exe
  D:\my_app\firebird.msg
  D:\my_app\intl\fbintl.dll
  D:\my_app\udf\fbudf.dll
```

firebird.conf:
```
  RootDirectory = D:\my_app
```

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

Uninstallation

The Firebird uninstall routine 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.

Other Notes

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

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
- move the database 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.

This leaves the security database isc4.gdb, which is considered writable by the code that should be simply validating a user's login, in order that isc4's header be updated for that transaction. Therefore, WinME probably makes a backup each time a user logs in.
We are trying to get an accurate problem description and a proven workaround to publish here. If you can help with the description and/or workaround, please post a message to the firebird-support list or to the firebird-devel newsgroup interface at news://news.atkin.com

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.

Install on UNIX / Linux

(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. Although the future is likely to be SuperServer, for the user just starting out with Firebird, the Classic server is likely to prove a better platform for initially experimenting with Firebird.

NOTES - READ THIS FIRST

1) You will need to be root user to install Firebird.
2) For SuperServer to install correctly you will need to add localhost to your /etc/hosts.equiv file.
3) If you require database access from any remote machines, you will also need to add the remote machine names into the /etc/hosts.equiv and /etc/hosts files.
4) Superserver won't install on Linuxen having a glibc package installed that is lower than glibc-2.2.93-5.
5) SS won't start if glibc is lower than GLIBC_2_3. CS should be OK with glibc-2.2.5 or higher.
6) For a rough evaluation of compatible Linuxen, refer to this table, 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.
7) For the rpm installs, ensure that the 'ed' editor package is installed on your system. If it is not present, the package will install but the installation scripts will fail.

Super Server edition installs are as shown below, except that the install files have a SS tag rather than a CS tag.

For linux rpm install

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

For linux .tar.gz install

```
$tar -xzf FirebirdCS-1.5.0-nnnn.tar.gz
```

```
$cd install
$./install.sh
```

* or `FirebirdSS-1.5.0-nnnn`

What the Linux Install will do
The Linux installations will
1. Attempt to stop any currently running server
2. If a previous installation of Firebird exists, then it and any associated files in /usr/lib /usr/include will be archived into the file /opt/interbase_<datetimestamp>.tar.gz and will be subsequently deleted.
3. Install the software into the directory /opt/interbase and libraries into /usr/lib and header files into /usr/include
4. Automatically add gds_db for port 3050 to /etc/services
5. Automatically add localhost.localdomain and HOSTNAME to /etc/host.equiv
6. SuperServer also installs a /etc/rc.d/init.d/firebird server start script. A new rcfirebird link is created in /usr/bin for the init.d script. This is preferred over direct execution of the Firebird initd script, because /usr/bin is on the search path for all users.
7. The /etc/rc.config Firebird entry is created on SuSE (SuSE specific configuration for service startup management).

Firebird should start automatically in runlevel 2, 3 and 5.

The .tar.gz packages do not support uninstallation.

The Classic install automatically sets up the xinetd entry if the /etc/xinetd.d directory is found, otherwise it will set up an inetd entry. Because some distributions locate xinetd in a different location than he /etc/xinetd.d directory, manual setup may be required in these conditions.

Testing your Linux installation

To test local access for your installation:

```
$cd /usr/local/firebird/bin
$isql -user sysdba -password <password*>

>connect / usr/local/firebird/examples/employee.gdb;

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

>quit;
```

To test remote access:

```
$cd / usr/local/firebird/bin
$isql -user sysdba -password <password*>

>connect '<hostname>:/opt/interbase/examples/employee.gdb';

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

>quit;
```

*If a password has been generated for you on installation, obtain it from the `/usr/local/firebird/SYSDBA.password` file.

Considerations for Linux

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

(XX = CS for Firebird Classic and SS for Firebird SuperServer.)

- XXchangeRunUser.sh—Create a new firebird unix user account and change the owner of the Firebird install and background tasks to run as to the firebird user.
- XXrestoreRootRunUser.sh—Restore the owner of the Firebird files, and the owner/user of the background tasks to the initial install default of root user.
  It is STRONGLY recommended for a secure Firebird installation that the server processes are NOT run as root. This does place some restrictions on who can initially create Firebird databases and where they can be created.
- changeDBAPassword.sh—Change the Firebird SYSDBA user password and, where necessary, Change the init script /etc/rc.d/init.d/firebird with the new password as well.

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.

Install Firebird Classic on MacOS X / Darwin

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

Build or Install Firebird on FreeBSD

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

Further Information

More information can be found about the Firebird database engine from:
http://firebird.sourceforge.net

or affiliated sites:
http://firebirdsql.org
http://www.ibphoenix.com
http://www.cvalde.com

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

with the word 'subscribe' in the Subject field.

Please do not use the firebird-devel list for posting support questions.

For technical support, please join the firebird-support list by going to
http://www.yahoogroups.com/groups/firebird-support

For InterClient and Java development and support, there is a specialized list:
http://www.yahoogroups.com/groups/Firebird-Java

The firebird-support list handles technical problems with Firebird and InterBase(R). Please take your Delphi and other client development environment questions to the appropriate forum.

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

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
news://news.atkin.com

Paid support worldwide for Firebird can be arranged through IBPhoenix (contact addresses and numbers are at http://www.ibphoenix.com). Several members of the Firebird team are also available for support and consultancy. Please contact them directly.

Database recovery services for Firebird or InterBase databases can be handled by IBPhoenix. For analyzing and possibly repairing corrupted databases yourself, try IBSurgeon at IBase.ru (www.ibase.ru) IBase.ru also provides recovery services in Russia and Europe.

Requests/offers for sponsored enhancements to Firebird can be taken directly to the FirebirdSQL Foundation Inc. 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.

General discussion about FB enhancements can be handled in the Firebird-priorities list (http://www.yahoogroups.com/community/Firebird-priorities.

IB-Architect (http://www.yahoogroups.com/community/ib-architect)is for **technical design discussions** ONLY. Support/conversion questions are definitely off-topic there.

Tools and Drivers

Database Desktop Client Programs

Several excellent choices of GUI admin desktops for Firebird are listed in the Contributed Downloads page at http://www.ibphoenix.com. Some are open source, some are freeware, others are established commercial products.

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 -
http://sourceforge.net/project/showfiles.php?group_id=9028

For advice and to participate in development and testing, join up with the Firebird-java forum at http://www.yahoogroups.com/community/firebird-java.

.NET: Firebird has an ongoing .NET driver project. Sources and binaries can be downloaded from the Firebird release pages at -
http://sourceforge.net/project/showfiles.php?group_id=9028

For advice and to participate in development and testing, join up with the Firebird .NET provider forum: http://lists.sourceforge.net/lists/listinfo/firebird-net-provider

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 http://www.ibphoenix.com. Lab for ongoing development of the open source ODBC/JDBC driver is in this forum: http://lists.sourceforge.net/lists/listinfo/firebird-odbc-devel

PHP: A group is 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

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

ftp://ftpc.inprise.com/pub/interbase/techpubs/ib_60_doc.zip

A structured Documentation Index is maintained on the Firebird community site at

http://firebird.sourceforge.net/index.php?op=doc

This is work-in-progress and all additions are welcome - send a message to firebird-docs@lists.sourceforge.net

Some installation guidelines and other HowTos may be found in the documentation area which can be linked to from
http://www.firebirdsql.org
or more directly from
http://sourceforge.net/projects/firebird

The main repository for user and technical issues is the IBPhoenix site -
http://www.ibphoenix.com
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:
http://www.borland.com/techpubs/interbase/

Bugfixes and Additions since Release 1.0

Tracker #	Description	Contributor
(no #)	Fixed minor inconsistencies in charsets naming.	P. Jacobi
(no #)	GSTAT crashed in some switch combinations.	D. Yemanov
(no #)	Fixed broken savepoint handling in BREAK|LEAVE/EXIT.	D. Yemanov
(no #)	Fixed optimizer to prefer single indices to composite ones and to prefer full-match unique indices.	A. Brinkman
(no #)	Enhanced Win32 install tools instsvc.exe and instreg.exe	O. Mascia
Improvement	Maximum number of indices per table increased from 64 to (DB_PAGE_SIZE/16)-2	A. Harrison, ported to 1.5 by N.Samofatov
721792	Long runnning connection caused mem leak in OS kernel device	N. Samofatov
775003	UDF log(x, y) in fact returned log(y, x)	P. Vinkenoog, N. Samofatov
774987	UDFs ltrim('') and rtrim('') returned NULL; rtrim forgot 1st char	P. Vinkenoog, N. Samofatov
(no #)	Fixed server crash caused by lost transaction context.	A. Peshkoff
(no #)	Fixed server crash for any combination of sub-select & between.	A. Peshkoff
736318	"<value> STARTING WITH <field>" fails when using indices.	D. Yemanov
(no #)	Non-existent deadlock is raised after execution of pre-(update/delete) triggers.	A. Peshkoff
Improvement	Make INSERTING/UPDATING/DELETING keywords non-reserved.	N. Samofatov
Improvement	Added new (more specific) error messages for some of v1.5 changes.	D. Yemanov, A. Brinkman, A. Peshkoff
Security fix	Added -login switch to instsvc allowing to install FB service as non-localsystem account.	A. Peshkoff
Improvement	Re-introduced trimming of VARCHAR fields in the remote protocols.	D. Yemanov
(no #)	Random server crash in the case of big queries being prepared.	D. Yemanov
Improvement	Configuration improvement – make path management in firebird.conf conform to the OS requirements.	A. Peshkoff

Tracker #	Description	Contributor
(no #)	Wrong UDF arguments of types DATE/TIME (dialect 3).	Oleg Loa
(no #)	Possible referential integrity violation.	Vlad Horsun, D. Yemanov
745090 and other RC 2 installation issues	Permissions problem for firebird.conf (SF #745090). Also generate aliases.conf on install; use rpmbuild to create Linux packages	Erik S. LaBianca, N. Samofatov
(no #)	Allow easy adjustment of LockSemCount on POSIX platforms. No need to use gds_drop or reboot machine to make new setting take effect	N. Samofatov
Improvement	Make FIRST/SKIP keywords non-reserved.	N. Samofatov
(no #)	Wrong attachment reference after exception in PSQL.	A. Peshkoff
(no #)	BREAK/LEAVE and EXIT statements are now available for usage in triggers.	D. Yemanov
(no #)	Possible index corruptions during garbage collection.	Vlad Horsun, D. Yemanov
(no #)	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) Only 27 unique filenames generated on win32 (which could cause unpredictable behavior in SS builds)	N. Samofatov
(no #)	Event manager change: disabled usage of definite aux port in CS builds due to known issues.	D. Yemanov

(no #)	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
		A. Brinkman

(no #)	Possible crashes on disconnect when event notification is used.	D. Yemanov
(no #)	Service manager changes: features of GSTAT/GSEC are not available via Services API in win32 CS (until v1.6 release).	D. Yemanov
(no #)	Wrong record statistics are reported when operation fails for some reason.	D. Yemanov
(no #)	stdin/stdout cannot be used to redirect console I/O in win32 build of GBAK.	A. Peshkoff

Tracker #	Description	Contributor
(no #)	Broken lock table resizing in CS. No more "lock manager out of room" (Win32 CS 1.5 RC1) or crashes (possible in all other CS builds of Interbase/Firebird).	N. Samofatov
Improvement	INTL improvement: make UPPER function work for WIN1251 charset without explicit collations.	N. Samofatov, D. Yemanov
BUGCHECK(291)	Possible database corruption when you modify/delete the same record in pre-trigger for which this trigger was called.	A. Peshkoff
(no #)	Buffer overrun in isc_database_info() call.	Oleg Loa
(no #)	Configuration manager change: now the server exits on missing / wrong firebird.conf with error report in system log.	A. Peshkoff
(no #)	Fixed Services API: enabled statistics Services API for POSIX CS builds.	N. Samofatov
(no #)	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 newly introduced tokens are made non-reserved	D. Yemanov
(no #)	Fixed Services API: partly enabled Services API for win32 CS builds.	D. Yemanov
(no #)	Wrong type of event delivery (unnecessary usage of OOB packets).	Jim Starkey, Paul Reeves
(no #)	Improved lock manager: deadlocks are now detected and reported as soon all blocking processes received notifications, i.e. instantly in all normal cases	N. Samofatov
(no #)	Server crashes in some Services API operations.	A. Brinkman
(no #)	Advanced security capabilities: implemented configurable access for databases, external tables and UDF libraries.	A. Peshkoff
(no #)	Fixed resource/memory leaks.	Mike Nordell, A. Peshkoff, N. Samofatov, D. Yemanov
(no #)	Buffer overrun with multidimensional arrays.	D. Yemanov
213460, 678718	Various issues with events used on multihomed hosts. NOTE Now it's also possible to setup a definite port for event processing.	D. Yemanov

Tracker #	Description	Contributor
(no #)	Fixed some resource leaks.	Mike Nordell, A. Peshkoff
(no #)	Fixed Services API: enabled Services API for posix CS builds. Notes: 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	N. Samofatov
(no #)	SQL enhancement: allow NULLs in unique constraints and indices (SQL-99 spec).	D. Yemanov, N. Samofatov
(no #)	Performance improvement: 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).	N. Samofatov
(no #)	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 deleted under savepoint.	N. Samofatov
(no #)	Improved EXECUTE STATEMENT. Now it's possible to return values from the dynamic SQL. Syntax: EXECUTE STATEMENT <value> INTO <var_list>; (singleton form) or FOR EXECUTE STATEMENT <value> INTO <var_list> DO <stmt_list>;	A. Peshkoff
(no #)	Server hangs during disconnect after mass updates.	D. Yemanov
(no #)	Improved optimizer: Subselects in SET clause of UPDATE now can use indices.	A. Brinkman
(no #)	"Context already in use" error in the case of DISTINCT with subqueries.	A. Brinkman
(no #)	Enhanced isc_database_info capability: list of currently active transactions is now available via isc_database_info call.	N. Samofatov
(no #)	Performance improvement: shortcut boolean evaluation. NOTE behaviour is controlled by "CompleteBooleanEvaluation" option of firebird.conf. Default is 0 (shortcut evaluation).	Mike Nordell
(Beta 2 bug)	Stack overflow during statement preparation.	D. Yemanov, Mike Nordell

Tracker #	Description	Contributor
(no #)	Performance improvement for IA32 CPU architecture: speed-up for index operations	Mike Nordell
(no #)	Change in universal triggers: allowed access to both (OLD and NEW) contexts in universal triggers.	D. Yemanov
(no #)	Improved optimizer: when an equal-node and other nodes (geq, leq, between...) are available for an index retrieval, then use the equal node always instead of the others.	A. Brinkman
(no #)	Long delays during connecting/disconnecting on WinXP.	A. Brinkman
(no #)	Generic cleanup: removed a lot of unused code.	Blas Rodriguez Somoza, Erik Kunze
523589	View is affecting the result of a query. Comment: Problem was that RSE's (inside a view) were not flagged as variant.	A. Brinkman
(no #)	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).	Blas Rodriguez Somoza
(no #)	The security database has been renamed to security.fdb.	D. Yemanov
(no #)	New configuration file: firebird.conf is finally published.	D. Yemanov
(no #)	New user-defined functions LPAD and RPAD added to IB_UDF library.	Juan Guerrero
(no #)	Sometimes GFIX didn't allow to specify "-user" and "-password" switches ("incompatible swiches" error).	D. Yemanov
(no #)	Security connection cache: connection to the security database is now cached, thus allowing to decrease time of subsequent database attachments.	D. Yemanov
Improvements	1. Reduce memory usage by the server. 2. Direct external I/O when the memory is not available for the sorting. Increased number of streams and predicates supported by the optimizer.	D. Yemanov
508594	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.	A. Brinkman
(no #)	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 NOTE Collations for the above charsets are not available yet.	Blas Rodriguez Somoza

Tracker #	Description	Contributor
(no #)	CREATE VIEW changes: disallowed PLAN subclause.	D. Yemanov
(no #)	Changed aggregate tracking behavior -- introduced backwards compatibility within aggregates. Deepest field inside aggregate determines where an aggregate-context should belong.	A. Brinkman
(no #)	Improved optimizer: better optimizations of "complex" JOIN queries (LEFT JOIN, views, SPs, etc).	A. Brinkman
(alpha 5 bug)	Major memory leaks are fixed.	D. Yemanov
(no #)	New API functions: IB7-compliant functions to return version of the client library -- isc_get_client_version(), isc_get_client_major_version(), isc_get_client_minor_version()	D. Yemanov
(no #)	Sort/merge improvement: merging (SORT MERGE plans) is now done via in-memory sorting module.	D. Yemanov
(no #)	New memory manager's internal has been changed to give us better performance.	N. Samofatov
(no #)	Win32 build 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.	D. Yemanov
(no #)	Trashed RDB$FIELD_LENGTH for views that contain concatenation of long CHAR/VARCHAR fields.	D. Yemanov

Improvement	Triggers improvement: added runtime action checks (INSERTING/UPDATING/DELETING predicates).
	
	Example:
	
	if (INSERTING) then
	
	new.OPER_TYPE = 'I';
	
	else
	
	new.OPER_TYPE = 'U';
		D. Yemanov

(no #)	Cursors (WHERE CURRENT OF clause) could not be used in triggers.	D. Yemanov
221921	ORDER BY has no effect.	A. Brinkman

Tracker #	Description	Contributor
213859	Subquery connected with 'IN' clause.	A. Brinkman
Improvement	Allowed arbitrary expressions in the ORDER BY clause.	N. Samofatov
(no #)	Engine crashed when UNIONS were used in a VIEW and that VIEW was used in the WHERE clause inside an subquery.	A. Brinkman
(no #)	Generic code cleanup: structures within Y-valve.	A. Peshkoff, N. Samofatov
Improvement	Single-line comments (--) are now allowed in any position of the SQL statement.	D. Yemanov
(no #)	"Request sycnhronization error" with BREAK statement.	D. Yemanov
625899	Bugcheck 291.	A. Peshkoff
(no #)	PSQL change: EXECUTE VARCHAR is renamed to EXECUTE STATEMENT.	A. Peshkoff
521952	No current record for fetch operation.	A. Brinkman
(no #)	QLI doesn't understand BIGINT datatype.	D. Yemanov
(no #)	Length of text variables inside procs/triggers wasn't copied to descriptor structure.	A. Brinkman
(no #)	FIRST/SKIP and ORDER BY changes -- 1. Implemented ORDER BY clause in subqueries. 2. Disallowed FIRST/SKIP for views. 3. Allowed zero as valid argument for FIRST.	D. Yemanov
(no #)	Buffer overflow (MAXPATHLEN) and rewritten local function dirname.	Erik Kunze
(no #)	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 "OldParameterOrdering" configuration manager parameter.	N. Samofatov
Improvement	Improved optimizer: let subqueries also use indices when their parent is a stored procedure.	A. Brinkman
(no #)	Removed request size limitation.	D. Yemanov
(no #)	Nulls first/last and collation handling in "order by" clause of unions	N. Samofatov

Tracker #	Description	Contributor
(no #)	Generic code cleanup; renamings, new safe macros, support for mingw.	Erik Kunze, Ignacio J. Ortega, D. Sibiryakov
(no #)	Explicit record locking implementation finalized. Should be stable and consistent now.	N. Samofatov
Improvement	Improved optimizer: better handling of AND nodes inside an OR node.	A. Brinkman
(no #)	Exceptions inside for/while loop in triggers are not handled correctly.	A. Peshkoff
623992	Double forward slash in connection string.	Paul Reeves, Mark O'Donohue
(no #)	Deadlock during some database operations.	A. Peshkoff
Improvement	Improved optimizer: if a few indices with much different selectivity could be used for index retrieval, only better of them are used while others are ignored.	D. Yemanov
(no #)	Quoted identifiers problem in plan expressions.	N. Samofatov
Improvement	CS architecture is now supported on Win32, but it still cannot be considered stable, so any feedback is welcome.	D. Yemanov
(no #)	Stored procedures are no longer recompiled before deletion.	N. Samofatov
(no #)	New collation for WIN1251 charset: WIN1251_UA for both Ukrainian and Russian languages.	D. Yemanov
(no #)	Client library change: API routines are no longer exported by ordinals.	D. Yemanov
Improvement	New configuration manager: enable the same plain file based configuration for all supported platforms.	D. Yemanov
Improvement	Improved optimizer: added better support for using indices with "OR". Pick the best available compound index from all "AND" nodes.	A. Brinkman
Improvement	Added support for explicit savepoint management in DSQL.	N. Samofatov
(no #)	Protocol cleanup: IPX/SPX network protocol is no longer supported.	Sean Leyne
(no #)	Obsolete platforms cleanup: 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	Sean Leyne
Improvement	Improved optimizer: added support for detecting use of index with sub-selects in aggregate select.	A. Brinkman
Improvement	Improved thread scheduler for Win32 SS: now the server should be more responsive under heavy load.	A. Peshkoff

Tracker #	Description	Contributor
Improvement	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]]	N. Samofatov
558364	Triggers fail to compile if PLAN used.	Ignacio J. Ortega
(no #)	Distributed (2PC) transaction cannot be properly rolled back due to network errors.	Vlad Horsun, Erik Kunze
(no #)	Generic cleanup: ISC_STATUS_LENGTH and MAXPATHLEN macros.	Erik Kunze
496784	When optimizer finds indexes for LEFT JOIN, work like INNER JOIN. Fixed problem which caused complex outer joins to produce wrong results.	N. Samofatov
(no #)	BLOB subtype is ignored in system domains generated for expression fields in views.	D. Yemanov
(no #)	Fixed installation bug: instreg.exe doesn't create "GuardianOptions" registry value.	D. Yemanov
(no #)	Resource leaks in DDL recursive procedure handling which caused some DDL to fail.	N. Samofatov
(no #)	Check constraint which uses only one table field is now dropped automatically when this field is dropped.	N. Samofatov
451927	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.	D. Yemanov
446240	Dynamic exception messages: allow to throw an exception with a message different to the one the exception was created with. Syntax: EXCEPTION name [value];	D. Yemanov
547383	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).	D. Yemanov
(no #)	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.	“Digitman”
(no #)	The server crashes during the garbage collection under heavy load.	N. Samofatov

Tracker #	Description	Contributor
Improvement	Deferred metadata compilation: solved a lot of causes of the well-known "object in use" error.	N. Samofatov
Improvement	New NULL order handling: allow user-defined ordering of NULLs.	N. Samofatov
(no #)	gstat showed wrong value for maxdup element.	D. Kuzmenko
(no #)	New registry key is used on win32: SOFTWARE\FirebirdSQL\Firebird.	--
451925	User-defined constraint index names: allows name of an index enforcing a constraint to be either constraint name or user-defined name.	D. Yemanov
Improvement	New RECREATE VIEW statement: shorthand for DROP VIEW / CREATE VIEW coupling of statements. Syntax: RECREATE VIEW name <view_definition>;	D. Yemanov
(no #)	Trigger which name starts with 'RDB$' cannot be altered or dropped at all.	D. Yemanov
(no #)	Renamed distribution files to make sure we're Firebird. Now they're fbserver, fbclient, firebird.msg etc. The client library is fbclient now and it should be used in all new FB-based projects. gds32 contains nothing but redirected exports and is provided for compatibility only.	Various
(Minor ODS upgrade)	Added new system indices (RDB$INDEX_41, RDB$INDEX_42, RDB$INDEX_43), now ODS version is 10.1.	D. Yemanov, N. Samofatov
451935	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>;	D. Yemanov
(no #)	Broken dependencies (like DB$34) appear in the database after metadata changes.	D. Yemanov
(no #)	Enhanced declaration of local variables: 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;	Claudio Valderrama
(no #)	Disabled BREAK statement for triggers (like EXIT) due to known internal limitations.	D. Yemanov

Tracker #	Description	Contributor
555839, 546274	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).	A. Brinkman
451917	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.	A. Brinkman
451917	New NULLIF internal function returns NULL for a sub-expression if it has a specific value, otherwise returns the value of the sub-expression.	A. Brinkman
451917	New CASE internal function allows the result of a column to be determined by a the results of a case expression.	A. Brinkman
545725	Automatic/background sweep hangs.	A. Peshkoff
(no #)	The server crashes when XSQLDA structures are not prepared for all statement parameters.	D. Yemanov
(no #)	PSQL: enabled support for empty BEGIN...END blocks.	D. Yemanov
567931	Partly fixed metadata security hole.	D. Yemanov
(no #)	BigInt arrays didn't work.	Artem Petkevych
437859	Implemented execute procedure and string concat, allowing any expression to be used as a SP parameter.	D. Yemanov
562417	Aggregate concatenated empty char.	D. Yemanov
Improvement	Readline (cmd history) support added to ISQL.	M. O'Donohue
446206	New BIGINT datatype allowing native SQL usage of 64-bit exact numerics (Dialect 3 only).	D. Yemanov
451922	Universal triggers allowing one trigger to be fired for a number of action types.	D. Yemanov
446238, 446243	New CONNECTION_ID and TRANSACTION_ID system available in PSQL. Return appropriate internal identifier stored on the database header page.	D. Yemanov
446180	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	D. Yemanov
(no #)	New plugin manager and INTL interface.	John Bellardo

Tracker #	Description	Contributor
Improvement	In-memory sorting: 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.	D. Yemanov
538201	Crash with extract from null as date.	Claudio Valderrama
446256	New EXECUTE VARCHAR PSQL extension statement allows execution of dynamic SQL statements in SPs/triggers. (Subsequently renamed to EXECUTE STATEMENT).	A. Peshkoff
(no #)	Major code cleanup.	Sean Leyne, Erik Kunze
(no #)	New memory manager.	John Bellardo
(no #)	New exception handling logic.	Mike Nordell, John Bellardo
(no #)	New autoconf-based build configuration.	John Bellardo, M. O'Donohue, Erik Kunze
(no #)	The code port from C to C++.	Mike Nordell, John Bellardo, M. O'Donohue