Firebird Version 1

Release Notes

November 2001

Contents

General Notes
Compatibility with Older Versions
New Features
Language Enhancements
External Functions (UDFs)
- ib_udf library
- FBUDF UDF Library (using Descriptors)

API Enhancements
Installation Notes
- Windows 32-bit
- Linux/UNIX
- Solaris
- MacOS X
- FreeBSD

Further Information
Documentation
Fixed Bugs
Known Issues

General Notes

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

This release was built from code which has been subjected to extensive cleanup, bug-fixing and beta testing during the year between the release of the code and this release. The On-Disk Structure is still ODS 10. Certain enhancements requiring changes to the ODS have been deferred to a future release.

A list of bugs fixed appears later in this document.

Several new language features have been added. For syntax, please refer to the Language Enhancements section later in this document.

The Firebird binaries can be downloaded through the FirebirdSQL.org domain -
http://www.firebirdSQL.org

Please refer to the Documentation section at the end of these release notes for locations of recommended documention.

Compatibility

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 and use a restored copy for experiments.

Please note the changes to functions in the ib_udf user-defined function library, described in the Language Enhancements section following this.

Any existing InterBase database will be 100% upwardly compatible with this software provided you follow the usual rules when migrating databases currently operating under an InterBase 4.x or 5.x server (ODS lower than 10), viz.

- Create a TRANSPORTABLE InterBase Backup file using the version of the program gbak (gbak.exe in Windows) that is correct for the ODS of the existing database. It is recommended that you give this file a ".gbk" extension. For Windows versions 4 and 5, the program ibmgr32.exe provides a graphical interface for gbak.exe in the Tasks | Backup menu. Do NOT use this program to work on a Firebird or an InterBase 6 (ODS 10) database.

- If necessary, the backup file can be zipped up and moved via transportable media. Remember that files saved to and then retrieved from a CD-ROM will need to have their Read-Only attribute unset.

- Using the version of gbak from your Firebird installation, restore the gbk file to the disk location where your Firebird database will be run. It must be located on the same physical machine as the ibserver program.

- Your converted database is now ready to run as a Dialect 1 Firebird database.

  The Operations Guide from the documentation set contains the command syntax for the gbak backup and restore program.

  Some utility programs, such as IBConsole, provide a graphical user interface tool for ODS 10 backup and restore. The recommended tool is IBBackup, a freeware binary by Martin Schmid <M.Schmid@EQUITANIA.de>, which can be downloaded from http://www.equitania.de/interbase/downloads/ibBackup.zip.

Version Strings for Firebird Releases

Version strings remain the same for Firebird 1.0 and return much as they have before, i.e. "LI-V6.2.0.368 Firebird".

This prepares the way for two connection mode options in the forthcoming Firebird 2:

- In native Firebird mode the server will listen on a different port (probably 3051) and return a version string like "LI-V2.0.0.XXX Firebird".

- It will have a "compatibility option" by which it listens on port 3050 and returns a compatibility version string (nominally "LI-V6.2.0.368 Firebird". This compatibility string will probably be user-configurable via Registry setting/conf file.

New items in the isc_database_info parameter - isc_vendor info and isc_info_fb_version - will provide an additional means for a client application to detect whether it is connecting to a Firebird or Borland Interbase database and to which Firebird version it is connecting.

isc_vendor info, and the isc_info_fb_version one as well.

New Features

Enhanced Dependency Checking

UDFs and generators are now tracked, so that it will not be possible to drop one if it is in use by a procedure or computed column.

Large Database File Support

(Sean Leyne, David Trudgett)

The logical database can comprise a primary file and numerous secondary files (the standing record for a single FB/IB database is 980GB!). However, this enhancement concerns the size of a single physical database file.

The maximum file size limit involves three separate factors:
- the implementation (coding) of the FB/IB engine I/O functions
- operating system support for 64-bit file operations
- filesystem limits on file size

Engine support for 64-bit file I/O

All Firebird/IB versions prior to FB v1.0 (including IB v6.0) used both 32-bit API/OS calls and 32-bit (integer) file pointer structures. The engine's capability was thus designed to handle files with a maximum file size of 4Gb. In the Unix code the file pointer structures were defined as signed integers, thus limiting Unix and Linux implementations to file sizes of 2Gb.

MS Windows

By design, all versions of Win32 (Win9x/ME/NT/2000) provide full support for 64-bit pointer file operations, provided the right file pointer structure is passed. To add large file support for Firebird, it was necessary only to change the file pointer structure to LARGE_INTEGER and to modify the I/O calls themselves slightly.

Unix

For Unix, 64-bit file operations are supported with the more recent versions of glibc (v2.4?). This support only ships with the latest distributions (Red Hat 7.x and Mandrake 8.x), or by ugrading older installations. Because 64-bit support is not guaranteed for all platforms, two versions of the Firebird Unix ports (for 32-bit and 64-bit I/O respectively) have been built.

David Trudgett notes: I'm not sure if that version you quoted is correct. Red Hat 7.1 ships with

GLIBC 2.2.2-10 and is reputed to work fine with large files. I use it myself on my workstation, but haven't tested it with a file greater than 2Gb yet.

MacOS X 10.0
John Bellardo: 64 bit I/O wasn't working correctly in MacOS X 10.0 (and you thought our version string was confusing :-) After updating to the just-released MacOS X 10.1 my tests are passing. I guess it was one of the items fixed between 10.0 and 10.1.

Filesystems

Although the engine can now support 64-bit operations, it is important to remember that not all filesystems support large files.

In the case of Win32, the following *file* size (not to be confused with partition/volume) limits apply:

FAT16 for Win9x/ME, max file size = 2GB - 1byte
FAT16 for WinNT/2000, max file size = 4GB - 1byte
FAT32 for Win9x/ME/2000, max file size = 4GB - 1byte
NTFS for WinNT/2000, max file size = 16,384GB - 1byte

For Linux and Unix flavors, because there are so many different filesystems, you should consult the following URL for the exact details:

http://www.suse.de/~aj/linux_lfs.html

Regarding Linux, at least three factors impact on maximum file size:

1. VFS (virtual filesystem) support
2. filesystem support
3. system library support (glibc)

The Linux kernel accesses different filesystems via the VFS (virtual filesystem) abstraction layer. In the past, the VFS itself had a (signed) 32 bit limit, which meant that *no* Linux files on any type of filesystem could be bigger than 2Gb.

I'm not sure when the VFS limitation disappeared, but it is definitely gone now (probably replaced by a 64 bit limit. I don't know much about (2) and (3) except that in the latest Linux distributions, they are no longer a problem either.

In the case of the Ext2 filesystem (the default native filesystem on most Linux systems), I believe it has either always supported large files or has done so for quite a while.

[Note to the reader, in case it's not obvious! Please treat the above info as an indication, not as gospel. I'm only the monkey repeating what I've heard from sources that I believe to be reasonably authoritative. - David Trudgett]

Support for 16kb pages is enabled

The engine and GBAK routines were modified to allow for PAGE_SIZE beyond the previous 8kb maximum. It will allow for a greater database size provide more options for tuning databases to fit the hardware environment.

Win32 Forced Writes Default = ON

With the introduction of IB 6.0 the default FORCED WRITES setting for newly created databases, under Win32, was changed to OFF, ostensibly to improve database performance but at the expense of database integrity/recovery. The default has been reverted to ON.

Expressly Define the Location(s) of External Files

The new configuration-level setting for locating EXTERNAL FILES is specific to platform. It is currently implemented only for Windows. only available on Windows. Here's how it works:

In the ibconfig file make an entry for each (existing) physical directory location where you want Firebird to search for external files. Notice that you must enclose the path in double quotes in this environment:

EXTERNAL_FILE_DIRECTORY "H:\test"
EXTERNAL_FILE_DIRECTORY "H:\external"

There is no limit on the number of directories that will be searched. Firebird will treat a series of entries as a search list.
Then -

```
isql> create table foo1 external file 'foo1.txt' (afield(char1), crlf char(2));
```

Using our example entries, Firebird will look in both h:\test and h:\external to see if the file foo1.txt exists. If it does, the external file table will be created and will point to the file in the right directory (although the rdb$external_file field in rdb$relations will store just foo1.txt, with no directory details).

When the table foo1 is accessed, the file will be picked up automatically in the proper external file directory.

If the file doesn't exist, it gets created by default in the default (current) directory, wherever you are when isql or the relevant utility is run. In this case, the directory name will be stripped from the external file definition in rdb$relations because you have already defined EXTERNAL_FILE_DIRECTORY. The assumption is that, if you are not specifying a path, then you are referring to an existing file in an external file directory that has been previously defined. It will pay you in this case, therefore, to run isql from the directory where you want the file to be found by the database engine.

If EXTERNAL_FILE_DIRECTORY isn't defined in the config file, the external file gets the full path treatment as as per ISC_expand_filename. If you

```
isql> create table foo1 external file 'h:\files\foo1.txt' (
    afield(char1),
    crlf char(2));
```

i.e. specify the directory name for the external file. If the file exists it gets defined as normal or, if doesn't exist, gets created as normal, in the directory specified.

Limit raised on External Table handles (NT)

The former limit of the number of external tables that can be in use simultaneously on NT is the default number of fopens allowed minus some files we read, or about 510.

The Firebird change increases that total number of fopens on NT to the highest value allowed on the operating system, 2048. The actual limit on the number of external tables that can be open

simultaneously is untested, but believed to be about 2040, depending on the number of databases open, the number of files per database, etc.

External files now closed when external table is dropped

A small change to the module dfw.e now closes external files when the external table that attached to them is dropped. This gets rid of the annoying problem of dropping the table and being unable to delete the associated file without shutting down the server.

CPU_AFFINITY Configuration Parameter

With Firebird SuperServer on Windows, there is a problem with Windows continually swapping the server process back and forth between processors on SMP machines. This ruins performance. Until now, to set ibserver's affinity to a single CPU, it was necessary to run the server as an application and to run a utility (IB_Affinity.exe) on top of the running server program.

This new configuration parameter can be added to ibconfig to remove the need for any external program to change CPU affinity on an SMP Windows system. It will take effect when the service starts up.

The CPU affinity is a config file parameter CPU_AFFINITY, taking one integer, that is the CPU mask.

Example:

CPU_AFFINITY 1
only runs on the first CPU (CPU 0).
CPU_AFFINITY 2
only runs on the second CPU (CPU 1).
CPU_AFFINITY 3
runs on both first and second CPU.

Connect to a Server Listening on a Specific Port

Feature id 1468, SF ID 447400

To connect to a server listening on a port that is not the standard 3050, you can now include the alternative port in the connection string. The syntax is as follows:

```
SQL> CREATE DATABASE "localhost/3050:/tmp/gurk.gdb";
```

Notice the slash instead of a colon between the server name and the port. You still need the colon before the physical path string.

Connection now defaults to Port 3050

The client and server will now use TCP port 3050 if no entry in the Services file is found. This should solve a common client/server installation problem.

Added -NONAGLE switch for Linux ports

Disabling the TCP/IP Nagle Algorithm typically improves speed on slow networks.

The Nagle TCP/IP algorithm was designed to avoid problems with small packets, called tinygrams, on slow networks. The algorithm says that a TCP/IP connection can have only one outstanding small segment that has not yet been acknowledged. The definition of small varies but usually it is defined as less than the segment size which on ethernet is about 1500 bytes.

By default, the socket library will use an internal algorithm known as Nagle's algorithm for buffering bytes on write before actually sending the data in order to minimise actual physical writes.

The presence of the new switch on Linux allows developers to determine, for themselves, the possible pro's and con's of using this alternative packet handling approach.

Additions and Changes to *isc_info_database*

Three new items added:

isc_info_db_provider, values are:

```
enum info_db_provider
{
isc_info_db_code_rdb_eln,
isc_info_db_code_rdb_vms,
isc_info_db_code_interbase,
isc_info_db_code_firebird
};
```

isc_info_db_class, values are:

```
enum info_db_class
{
isc_info_db_class_access = 1,
....
isc_info_db_class_cache,
isc_info_db_class_classic_access,
isc_info_db_class_server_access
};
```

isc_info_firebird_version, values are set in inf.c

One item renamed:

```
isc_info_isc_version
```

which was isc_info_version.

The old name is #defined to the new, so everything works exactly as it always has. In the future we can change isc_info_version to point to isc_info_firebird_version.

Inf.c has code to handle the new items.

GDS_VERSION was redefined in license.h to reflect Firebird simultaneously being 6.2 and 1.0. Also defined was FB_VERSION which we can be cut over to eventually.

The list of implementations in utl.c was cut over from InterBase/xxx to Firebird/xxx. The implementation numbers in common.h and ibase.h were changed to match the list of implementations in utl.c

GBAK has new COUNTER option for -V(erbose) switch

The -V (Verbose) option of GBAK now allows for a 'counter' value to be specified -- i.e. GBAK ... -V 20000

This value will provide additional user feedback, as the data rows are backed-up or restored. Once

GBAK has processed the defined number of rows (either backed-up/restored or re-indexed), GBAK will print a message indicating the running row count value.

By default, the counter value is 10,000.

isql double quotes requirement on-x and -a options relaxed

When extracting object names for an script with either option -x or -a, isql always took the easy path and put double quotes around identifiers. This was deemed annoying by several users. It will now use double quotes only if the name cannot be expressed without them.

An identifier doesn't need to be surrounded in double quotes if it contains:
a) Only ASCII A-Z (uppercase only)
b) 0..9 digits
c) underscore (_) and $ provided that in the first position, only a) is met.

Blank and zero length identifiers always need double quotes.

Language Enhancements

CURRENT_USER and CURRENT_ROLE

These two new context variables have been added to reference the USER and (if implemented) the ROLE of the current transaction 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 0 AS
BEGIN
  INSERT INTO USER_LOG VALUES(
    GEN_ID(GEN_USER_LOG, 1),
    CURRENT_TIMESTAMP,
    'ATABLE',
    NEW.ID,
    'I',
    CURRENT_USER,
    CURRENT_ROLE);
END
```

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>;
```

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
```

However, there are a number of anomalies.
First, DSQL has a method that checks for aggregates correctly used. This is invalid:

```
select rdb$relation_name from rdb$relations
group by rdb$owner_name
```

It gives "invalid column reference" because the statement doesn't meet the rules for aggregation.

But this statement goes undetected:

```
select rdb$relation_name from rdb$relations
group by strlen(rdb$owner_name)
```

even though it is totally nonsense. The basic rule is that when you group by something, this "something" is the only scalar expression that can appear in the SELECT list. All the other fields should be aggregate values.

Secondly, even if we solve the first anomaly, we are left with this subtle case:

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

Clearly it's invalid again. The two scalar expressions should be the same. This is equivalent to saying:

```
select A, count(*)
group by B
```

DSQL lacks code to check this problem. The only way to check it is to do recursive comparison of A and B to see if they involve the same expression, involving UDF parameters. The reason it doesn't check is that previously the engine didn't accept grouping by UDFs.

Various other "simple" cases should be rejected, too.

A side effect of the changes enabling grouping by UDFs is that, whereas you originally couldn't call built-in Firebird functions in GROUP BY. Now, by creating a dummy UDF wrapper, you can:

```
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)
```

At least be aware that some serious issues exist and that it's up to you (the programmer) to make sure your GROUP BY UDF clauses are capable of returning valid results - because the DSQL does not yet have the capability to distinguish valid from absurd. Simply put, the functionality lacks enough logic at this point to ensure correct udf usage in these scenarios.

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.

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.

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

Retrieves the first *m* rows of the selected output set. The optional SKIP clause will cause the first *n* rows to be discarded and return an output set of *m* rows starting at *n + 1*. In the simplest form, *m* and *n* are integers but any Firebird expression that evaluates to an integer is valid. 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. [Variable binding is not fully tested yet.]

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;
```

A Gotcha with SELECT FIRST

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!

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.

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.

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.

New PLANONLY option for statements

Support for PLANONLY setting, allows for a statement/query to be submitted to the engine and the plan retrieved, without executing the statement/query.

Allow FK indexes to be disabled

It is now possible to set FOREIGN KEY indexes inactive using ALTER INDEX. This should resolve some performance issues related to maintaining indexes with low selectivity.

Case Insensitive Hungarian Collation Set

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

New Comment marker for Scripts

The engine will now accept the single-line comment marker "--" in scripts, e.g.

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

The engine will ignore the '--' and everything between it and the next end-of-line marker OR the next end-of-statement marker, whichever comes first.

So, for example, you can't take this statement

```
CREATE GENERATOR MY_GEN;
```

and "comment it out" by placing the '--' symbol at the beginning of the line:

```
--CREATE GENERATOR MY_GEN;
```

because the engine encounters the end-of-statement marker (semi-colon in this case) before it encounters the end-of-line marker. For the server, it is equivalent to:

```
/* CREATE GENERATOR MY_GEN */;
```

API Enhancements

Enhancements to isc_database_info items set
Funded by Jason Wharton, CPS (IB Objects)

In the API, four request buffer items have been added to the isc_database_info items structure, for calling it to retrieve transaction statistics without needing to initialize and call the Services API, viz.

Item	
65	isc_info_oldest_transaction
66	isc_info_oldest_active
67	isc_info_oldest_snapshot
68	isc_info_next_transaction

All items return integers. Please refer to the API Guide for examples of calling isc_database_info(). The example found there fits perfectly, since it retrieves integer values and our new items are integers, too.

Replace the items declared in the example with:

```
char db_items[] = {
isc_info_oldest_transaction,
isc_info_oldest_active,
isc_info_oldest_snapshot,
isc_info_next_transaction,
isc_info_end};
```

The example has a loop with a switch(). Inside the loop, change the "case" to use our new values. There should be 4 cases plus the default one instead of the 2 cases + default in the example.

External Functions (UDFs)

In the "standard" library, ib_udf.dll, ib_udf.so

SUBSTR(<*string expr*>, <*pos1*>, <*pos2*>)

Returns a string consisting of the characters from <*pos1*> to <*pos2*> inclusively. If <*pos2*> is past the end of the string, the function will return all characters from <pos1> to the end of the string. NOTE that this behavior is different from that of the external function SUBSTR in Borland and previous Firebird versions of ib_udf.dll UDF library, which returns NULL when <*pos2*> is past the end of the string.

```
UPDATE ATABLE
SET COLUMNB = SUBSTR(COLUMNB, 4, 32765)
WHERE...
```

SUBSTRLEN(<*string expr*>, <*pos*>, <*length*>)

Returns a string of size <*length*> starting at <*pos*>. The length of the string will be the lesser of <length> or the number of characters from <pos> to the end of the input string.

```
UPDATE ATABLE
SET COLUMNB = SUBSTRLEN(COLUMNB, 4, 99)
WHERE...
```

ascii_char()

Claudio Valderrama fixed an old bug in the declaration for ascii_char supplied in ib_udf.sql. It has been corrected so that it returns a one-character C string instead of (erroneously) an InterBase CHAR(1) type:

```
DECLARE EXTERNAL FUNCTION ascii_char
       INTEGER
       RETURNS CSTRING(1) FREE_IT
       ENTRY_POINT 'IB_UDF_ascii_char' MODULE_NAME 'ib_udf';
```

In the new Firebird UDF library, FBUDF.dll

Development funded by Craig L. Leonardi

Distributed with this release is FBUDF.dll, the first Firebird UDF library using BY DESCRIPTOR syntax to pass arguments, which provides more versatility. In declaring the functions in SQL, notice the differences between the multiple declarations that map to the same function. See, for example, that INULLIF() is mapped to by both INULLIF() and I64NULLIF().

Author Claudio Valderrama comments that the library is still in beta and warns that it will probably exhibit some bugs. So far it has not been compiled on any other platform but Windows. Bug reports and comments are welcome.

The source and the DDL for declarations are in the Firebird CVS tree. To find them, select 'Browse the CVS tree' from http://sourceforge.net/projects/firebird, click on 'Browse CVS Repository' and then select Developers | Latest sources | interbase | extlib | fbudf.*.

NVL() functions for both exact precision ('invl') and string ('snvl') parameters

These functions attempt to mimic the NVL function of Oracle, to output an actual value when the column has a NULL value. They take two arguments, the first being the expression being tested for NULL, the second the value to output if the first argument is NULL. NVL will return the first argument if it's not null and the second argument if the first one is null. If both are null, you get null.

The pair of parameters should be compatible, either two numeric values (smallint, int, int64) or two string values (char, varchar, cstring). The engine does not honor the parameter types when using the technique exercised by FBUDF, so mixing a numeric and a string as arguments will yield wrong results.

NULLIF() for string ('snullif'), integer ('inullif') and INT64 ('i64nullif') parameters

NULLIF should take two arguments, returning NULL if they are equivalent, or the result of the first expression if they are not equivalent. Because of a shortcoming in the engine which prevents NULL being returned from a UDF, each of these three functions returns a zero-equivalent. This non-standard behaviour makes it not useful for "casting" certain values as NULL in order to have aggregate functions ignore nulls.

NOTE that the function call to both the integer and int64 functions is the same ('inullif').

Day-of-Week functions - one returning a short string ('SDOW'), the other a longer one ('DOW'), from a timestamp input. The return strings can be localized.

Several functions to **add segments of time to a timestamp** - 'addDay', 'AddWeek', etc.

A RIGHT() function (like RString in BASIC) to return the rightmost n characters from an input string.

A GetExactTimestamp() function returning the system timestamp with milliseconds precision.

Truncate()and i64truncate() truncate 32-bit and 64-bit integers respectively, taking scaled (exact-precision) numerics of any range (up to 9 in Dialect 1 or up to 19 in Dialect 3) and returning the whole-number portion. They do not work with float or double types.

```
truncate(14.76) returns 14
round(14.22) returns 14
```

Round() and i64Round() take 32-bit and 64-bit integers respectively, accepting scaled (exact-precision) numerics of any range (up to 9 in Dialect 1 or up to 19 in Dialect 3) and returning the nearest whole number. You cannot specify the number of decimal places.

```
round(14.76) returns 15
round(14.22) returns 14
```

String2blob() converts a char or varchar type to a blob. It is like the function that exists in freeUdfLib, but is much simpler internally.

INSTALLATION NOTES

Install on Windows 32

Clean any existing Firebird installation off your system by selecting in Settings|Control Panel|Add/Remove Programs..

After unzipping, run the downloaded self-install, FBWin32Setup.exe.

By default the installer script will install to a c:\Program Files\Firebird\ root. The script gives you the option to install to a different root if you wish.

- For a **client-only install**, take the default options from here on
- For a **full server and client install**, make sure to check the item

  ☐ Server for Windows

  and also, if you want it

  ☐ INTERSOLV InterBase ODBC Driver which is the ODBC driver for InterBase 5.x

Startup Adjustments
(Andy Canfield)

If the installer fails to set up to start the server automatically at boot-up, do this (changing the drectory references appropriately if you installed to a different root):

- Use Windows Explorer to go to C:\Program Files\Firebird\bin
- Right click on "ibguard.exe" and pick Copy from the pop-up menu
- If you wish the server to start on every boot, go to C:\WINDOWS\StartMenu\Programs\StartUp
- If you wish to run the server manually, go to C:\WINDOWS\StartMenu\Programs\Firebird
- From the menu, pick Edit|Paste Shortcut.
- If you wish, rename the new shortcut icon as "Firebird Server" or something else useful.
- Go into the Properties applet of the shortcut and
  - change the default directory to C:\WINDOWS (C:\WINNT\System32 on NT)
- If you want to set up manual startup, e.g. in the Firebird page of your Start Menu, add the application switch "-a" to the "Target" field, e.g.

  Target: "c:\Program Files\Firebird\bin\ibserver.exe -a"

- Use the "Start" menu to test it. The server icon should appear on your task tray.

Considerations for Windows
You can store both InterBase and Firebird on the same Windows computer. However, in order to switch from one to the other you must shutdown the one, alter the registry, and start the other.
The registry keys are in
[HKEY_LOCAL_MACHINE\Software\Borland\InterBase\CurrentVersion]
The two keys to change are RootDirectory and ServerDirectory.
RootDirectory must be either

C:\Program Files\Borland\Interbase\
or C:\Program Files\Firebird\
Similarly ServerDirectory must be either
C:\Program Files\Borland\Interbase\bin\
or C:\Program Files\Firebird\bin\

When you run the Firebird version of ibserver, use the Firebird version of ibguard. Run the InterBase version of ibguard with the InterBase version of ibserver.

Similarly, use the correctly versioned Windows client program gds32.dll (normally installed in your Windows (in 9x and ME) or \WINNT\System32 directory (NT, 2K).

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 ib-support list or to the firebird-devel newsgroup interface at news://news.atkin.com

Install on UNIX / Linux

(Mark O'Donohue)

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:
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 file.

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 -Uvh FirebirdCS-1.0.0-0a.Beta2.i386.rpm
```

For linux .tar.gz install

```
$tar -xzf FirebirdCS-1.0.0-0a.Beta2.tar.gz
$cd install
$./install.sh
```

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. Change the default sysdba password (a .tar.gz will prompt the user for the new password, and a rpm will generate a random one and leave it in the file /opt/interbase/SYSDBA.password).
5. SuperServer also installs a /etc/rc.d/init.d/firebird server start script.

Testing your Linux installation

To test local access for your installation:

```
$cd /opt/interbase/bin
$isql -user sysdba -password <password*>

>connect /opt/interbase/examples/employee.gdb;

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

```
>quit;
```

To test remote access:

```
$cd /opt/interbase/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 /opt/interbase/SYSDBA.file.

Considerations for Linux

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

(Replace XX in the two scripts with 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 do NOT be run as root.

Doing so however 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.

//end of notes on Linux/Unix

Install Firebird Classic & SuperServer on Solaris 2.7 Sparc

(Neil McCalden)

This is a development release of the Firebird for Solaris compiled from a cvs snapshot as of 7-Jan-2001 at 1845hrs GMT. Both classic and super server versions have passed the standard set of TCS tests.

The programs identify themselves with the version 'SO-T0.9.4.49 Firebird Test1' when called with -z. The T stands for Test, which comes before Beta which is a reflection of the relative youth of the Firebird builds than the maturity of the core product.

Basic install steps (Classic and Super Server):

As root, extract the accompanying .tar in to the directory of your choice.

Create the links :-

```
ln -s /ExtractDirPath/interbase /usr/interbase
ln -s /usr/interbase /opt/interbase

cd /usr/interbase
./install
```

This will create links for header files, libraries and update the /etc/services and /etc/inetd.conf files.

Extra steps for Super Server

Add localhost to /etc/hosts.equiv
Create an interbas or firebird user and group
Create script in /etc/init.d|rc3.d to start server
- for an example see /usr/interbase/bin/firebird

If you have an earlier version of InterBase(tm) installed it is probably installed in /opt/interbase. You will need to remove the package or rename the directory as appropriate.

Note the examples files are not included in this release, these are available from the file downloads section on firebird.sourceforge.net.

//end of notes on Solaris 2.7 Sparc

Install Firebird Classic on MacOS X / Darwin

(John Bellardo)

Last Updated 12 January 2001

*** CAUTION ***
Firebird is compiled from the source tree as of mid-September, so it has all the bugs and security problems from back then. I'm working on getting the changes into the source tree so we will have up to date versions.

Firebird currently requires a new kernel image to function correctly, so this process is not for the faint of heart. Read all the directions carefully before you do anything, and don't skip steps.

With that said, here is what you need to do:

1. Download and uncompress the following files:
    - Firebird framework: Tar-GZip File

- Darwin Kernel: GZip File

2. As root / Administrator copy the Firebird.framework directory (the "firebird framework" from now on) into
"/System/Library/Frameworks".

3. Open up Terminal.app.

4. type "su" and hit return, then enter the root password at the password prompt. If you don't get a password prompt, don't worry, everything is good.

5. type "source ", then drag-and-drop the fix_permissions file into the terminal app window and hit return. This is needed for security reasons.

6. type "cp ", then drag-and-drop the mach_kernel.firebird file into the terminal app window, then type " /mach_kernel.firebird" and hit return.

7. type "cp /mach_kernel /mach_kernel.backup" and hit return.
This step makes a backup of your current kernel. This is VERY IMPORTANT or else you won't be able to revert back to your old kernel.

8. type "cp /mach_kernel.firebird /mach_kernel" and hit return.

9. restart your computer. See the section below if you have problems booting.

Assuming your computer has started correctly, you are done.

There are only command line programs available to access the databases (currently). All the command line executables are located in
/System/Library/Frameworks/Firebird .framework/Resources/bin.

To run, for example, isql:

1. Open Terminal.app

2. type
"/System/Library/Frameworks/Firebird.framework/Resources/bin/isql"
and hit return

Don't move these programs to a different location. To work correctly, Firebird needs to find them where they are.

When compiling a program using the Firebird API specify the "-framework Firebird" option to the compile command and use '#include ' for the headers instead of '#include' in your source code.

This does NOT INSTALL THE INTERNET SERVER. Any development version of an internet server should not be used unless you know what you are doing.

This DOES include all local database access, and the code to allow you to access other Firebird interbase servers. If you want to enable the internet server send me email (address below) and I will send you instructions. This is extra important because this version of the source code still contains the back door (it won't for long) and there is no fix for Darwin.

Switching between your kernels
There are many reasons to switch your kernel back to the old one, that is why you made a backup. The firebird kernel does not have AirPort support, and doesn't support some other devices because it is based on the Darwin source, not an official Apple release.

To restore your kernel:

1. Open Terminal.app

2. type "su" and hit return, then enter your root password at the password prompt . If you don't get a password prompt, don't worry, everything is good.

3. type "cp /mach_kernel.backup /mach_kernel" and hit return

4. restart your machine

You should now boot into your old kernel. Firebird will no longer work, but everything else you lost access to by booting into the firebird kernel will.

To switch back to the firebird kernel:

1. Open Terminal.app

2. type "su" and hit return, then enter your root password at the password prompt . If you don't get a password prompt, don't worry, everything is good.

3. type "cp /mach_kernel.firebird /mach_kernel" and hit return

4. restart your machine

If you can not boot into MacOS X (because, for example, the firebird kernel does not work with your computer) follow these instructions to restore your backup kernel from OS < X.

1. Boot into MacOS < X

2. Open your MacOS X / Darwin Partition

3. There should be mach_kernel.backup, mach_kernel.firebird, and mach_kernel files

4. delete the mach_kernel file

5. duplicate the "mach_kernel.backup" file and name it "mach_kernel"

6. reboot your computer in MacOS X

If this doesn't work for you, Darwinfo is a great resource for more detailed directions.

Feedback is really important. I'm fixing bugs as I find them, but I am just one man. If you find a bug, problem or missing feature, let me know.

If you need more help send me email (but please read the interbase documentation or visit the ib-support list instead of asking me user support questions).

Also, if you want to know how to enable the server, drop me a line and I will tell you.

Thanks for you interest in the Darwin port of Firebird!!

John Bellardo
<bellardo@cs.ucsd.edu>

Build or Install Firebird on FreeBSD

(Geoffrey Speicher)

The recommended way is to build and install the port (as root):

```
# cd /usr/ports/databases/firebird
# make install
```

An alternative is to install the package (the downloadable version) with pkg_add:

```
# pkg_add FirebirdCS-0.9-4.FreeBSD.FreeBSD.tar.gz
```

Next, do the following:

```
chown root /usr/local/firebird/lib
```

This is due to the fact that ldconfig will ignore directories not owned by root, and subsequent references to the shared library will fail.

In either case, you'll need to put the following into the file /usr/local/etc/rc.d/firebird.sh :

```
#!/bin/sh

case $1 in
start)
    [ -d /usr/local/firebird/lib ] &&
         /sbin/ldconfig -m /usr/local/firebird/lib
esac
```

Then be sure to 'chmod 755 firebird.sh'.

The shared library reference catches everyone who adds Firebird as part of the mod_php4 port. See PR ports/25907 for a fix that should be committed.

Further Information

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

or affiliated sites:

http://www.ibphoenix.com
http://www.interbase2000.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 this list for posting support questions.

For technical support, please join the ib-support list by going to

http://www.yahoogroups.com/groups/ib-support

The open source community (IBDI) also operates several other discussion lists on various aspects of Firebird development. For details, please refer to the Mail Lists and Newsgroups section of the IBDI site at http://www.interbase2000.org.

The Firebird developers' list and the IBDI lists, along with some other lists of interest to Firebird and InterBase developers, are mirrored as newsgroups at

news://news.atkin.com

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

http://www.interbase2000.org/ib_doc.htm.

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

Some additional documentation may be discovered by visiting the Borland techpubs area:
http://www.borland.com/techpubs/interbase/

Bugs Fixed Since Release of the Source Code

SFID 448062.- **CREATE DOMAIN ... CHECK (condition)** left rdb$fields.rdb$validation_source being CHECK (condition) but ALTER DOMAIN ... ADD CONSTRAINT CHECK (condition) leaves rdb$fields.rdb$validation_source being CONSTRAINT CHECK (condition) that's inconsistent. Now, the first format is always stored; the extra CONSTRAINT word is always omitted.
Fix funded by Jason Wharton, CPS (IB Objects)

SFID 227760.- **Zero length identifiers are now forbidden**.

The XSQLVAR struct used to communicate data to the client application doesn't convey information to distinguish between no ident (NULL) and blank ident. Therefore, blank field names (zero-length names) have been forbidden. Also, the server internally trims the trailing blanks, so a name like " " (only blanks) becomes a zero-length identifier and is forbidden, too. Same correction was done for both CREATE and ALTER syntax and for every different object type. In addition, several places where the engine couldn't handle embedded or leading blanks in dialect 3 identifiers (legit use) have been fixed.
Fix funded by Jason Wharton, CPS (IB Objects)

SFID 428889.- **Column position was treated as zero-based but should be 1-based**

The syntax

```
alter table...alter column...position <n>;
```

is 1-based in Firebird. At a logical level, it is handled the same way as ORDER BY <n>, regardless of the fact that, internally, the engine uses the C convention and starts at zero.

Hence, to put some field in the first position, the command is:

```
alter table tbl
alter column cln position 1;
```

An incompatibility now exists because Firebird has now been corrected to conform with both the standard and the IB6 documentation. To quote from EmbedSQL.pdf, chapter 5:

...
The ALTER TABLE ALTER command allows you to change the column position and name as well. For example, the following statement moves a column, EMP_NO, from the third position to the second position in the EMPLOYEE table:

```
ALTER TABLE EMPLOYEE ALTER EMP_NO POSITION 2;
```

...

The example shown above, taken from the manual, works in Firebird. In IB it does nothing, since 2 is the third position for InterBase 6, whose syntax still has the bug of treating column position as if it were zero-based.

SFID 228526.- **Ambiguous JOIN statements are now rejected**

InterBase does not prevent you from submitting a statement like this:

```
SELECT A.FIELDA,  B.FIELDA
FROM A JOIN B
ON FIELDX = FIELDY
WHERE FIELDA="99"
ORDER BY FIELDA
```

Such statements return unpredictable output sets. Now Firebird will return an error if there are any unqualified column identifiers in join statements.

SFID 223133.- **Ambiguous self join produce bizarre results**
Also fixed.

SFID 460261.- **Blob API had problems with blanks embedded in names**

The following API calls would fail if a dialect 3 name with embedded blanks is presented:

isc_blob_default_desc
isc_blob_lookup_desc
isc_blob_set_desc

The culprit, the function get_name() in blob.e, was fixed.

SFID 436462.- **Rows affected incorrect with BEFORE UPDATE trigger on views**

On updatable views that have Before Update triggers, updating one row returned 3 rows affected. Fixed.

SFID 444463.- **BEFORE triggers were firing after checks**

Before triggers (insert and update) were firing after CHECK constraints, which allowed to change fields to values that would fail table check constraints. This was wrong and could make data unrestorable.

SFID 229009.- **CREATE VIEW not returning syntax error**

IB6 would allow CREATE VIEW command where number of view columns does not correspond to number of columns in select statement. According to SQL92 rules, such statement should return syntax error. Now it does.

SFID 458888.- **REFERENCES privileges were absurdly constrained/could cause crashes**

The bug reported was that a dependent table required REFERENCES privileges to be declared explicitly for its foreign key trigger to get permissions on the table it was referring to. This was fixed so that a table which owns an internal trigger gets implicit REFERENCES rights to the other table. The code that checks REFERENCES in "run-time" (at DML time) has been commented out and the response to the trigger's access request has been softened so that the conditions for the reported bug can not arise.

Analysis of this bug uncovered two more bugs, which have also been fixed:

- REFERENCES checking at "design-time" (DDL) was subject to being bypassed, so code was added to reinforce it even if the bypass should occur. Now it becomes mandatory for the creator of a foreign key to have REFERENCES rights on the master table or be the owner of the master table.

- fixes for the first bug also cured a bug that had been discussed but not logged. The REFERENCES checking bug was also indirectly responsible for a situation reported, where backing up and restoring a complex database would cause a crash immediately upon selecting from virtually any table. The analysis of the bug, which was complicated, uncovered a "deadly circle" of re-entrant REFERENCES checking.

SFID 446237.- **'Column not found' error occurred where it should not**

Eliminated.

SFID 229860.- **Wrong error message**

The message ' DATE data type is now called TIMESTAMP' was being returned on unrelated errors, such as typos. Fixed.

SFID 460624.- **isql extraction of procedure parameters was broken in Dialect 3**

Fixed a bug in isql where it would not extract procedure parameters' names correctly in dialect 3, even if the names required double quotes and these were suppiled. An incorrect parameter declaration in the routine that does the double quotes was also fixed.

SFID 451798**.- FIRST is applied before aggregation**
SFID 451810.- **SKIP is off by 1**

Both fixed.

SFID 412417.- **Error altering from CHAR to VARCHAR**

Altering CHAR to VARCHAR column was adding 2 bytes to field length. Fixed.

SFID 231998.- **space before CASTed numeric expression in dialect 1**

Eliminated.

SFID 212177.- **error with non-english column defaults**

GBAK would raise a transliteration error during restore with defaults stored in a database that was using a non-English character set.

SFID 221589.- **numeric fields and mathematical operations**

```
select field1 * field2 from mytable
or
select field1 * (1+field2/100) from mytable
```

where both fields were numeric (9,2) produced incorrect results.

SFID 450301.- **SUBSTRING did not work**

The new function was not working when used in a where-clause (e.g. with in or =) or in string-concatenation. Fixed.

SFID 223059.- **Updating VARCHAR did not clear old data**

When IB 6 updates the string stored in a VARCHAR, it does not zero rest of the string but concatenates the old value to the update value. Because VARCHARs contain length of the string, the client application will never notice any problem (i.e. it will always see correct result), but the gdb file can grow faster than expected (because such additional data can't be rle compressed), and database can get slower (because less useful data fits onto page).

Fixed in Firebird.

SFID 223512.- **DROP VIEW was causing the underlying table to be dropped**

Fixed.

SFID 226456.- **SELECT/PLAN did not understand delimited SQL index names**

Fixed.

SFID 419964.- **buffer overflow in remote/interface.c li**

This was caused by the version string buffer being only 64 bytes long. It has now been increased.

SFID 453686.- **Unable to create a dialect 3 database with Firebird 1 betas**

This problem has been occurring using clients (e.g. IBConsole and IBExpert) that need to connect to a major version number greater than 5. It is fixed in Firebird RC 1.

SFID 233124.- **Connection lost during execution of bad code**

Eliminated.

SFID 425799.- **Renamed domain left behind dimensions**

With a domain being an array, renaming the domain caused rdb$field_dimensions to be left unchanged; hence the connection between a domain and its dimensions specification was broken.
Fixed.

SFID 450405.- **Tricky role could defeat basic SQL security**

It was a convoluted example: but the loophole has now been closed.

SFID 462800.- **Non-unique pair in RDB$FORMATS**

Fixed.

SFID 447377.- **GDS error ...can't find TIP**

There is a bug in InterBase 5.6, 6.01, and early Firebird betas that causes the lookup of a transaction inventory page to fail if there are more than 32767 transaction pages. That makes the maximum safe transaction id for a database with:

```
        1024 byte pages   131,596,287.
        2048 byte pages   265,814,016.
        4096 byte pages   534,249,472.
        8192 byte pages 1,071,120,384.
```

Although those are large numbers, there was a particular database exceeded 131 million transactions in six months. Attempts to attach to the database failed with the error gds internal consistency check, can't find TIP.

Suggestions:
1) don't use a 1024 byte page size.
2) do check your next transaction number from time to time.
3) if you see the next transaction number approaching the limit, backup and restore the database.

SFID 229231.- **REVOKE is sensitive about the case of user names**

Fixed.

SFID 421260.- **Character length not filled for UDFs**

In the case of ODS 10, it was written only as a stub. Now completed.

SFID 227375.- **Grouping on derived fields processing NULL data kills InterBase**

This bug occurred when a SELECT on a view that calculated a derived column's value by subtracting one value from another attempted to GROUP BY this derived column. It now works properly in Firebird.

SFID 425949.- **Engine CRASH Error**

This statement

```
select count(*),adresy.rdb$db_key from adresy
```

can crash InterBase. adresy can be any table. Fixed in Firebird.

SFID 228467.- **Security bug with a hardcoded user with full rights to isc4.gdb**

This was the security vulnerability that was discovered to affect all versions of InterBase from Version 4.x forward. It was fixed in Firebird and also later by Borland in InterBase 6 source code and binary versions after January 2001.

SFID 229121.- **TEMP directory filling up**

Fixed.

SFID 213708 **-502 Declared cursor already exists**

Occurred in Microfocus programs connecting locally to IB6 Classic on AIX and remotely to IB6 Superserver on WINNT. Fixed.

SFID 214298.- **Select count(*) expression anomaly when table was empty**

select count(*) + 1 returned zero when table had no rows but returned correct result if count(*) was greater than 0. Fixed.

SFID 216579.- **Generators in COMPUTED BY columns would return wrong results**

Using generators in COMPUTED BY columns would return wrong results and render the database unusable. Fixed.

SFID 222476.- **AVG and SUM returned empty field names in dialect 3**

Fixed.

SFID 216733.- **Too Many Generators Could Corrupt Database**

The number of generators you can have is dependent on (page size - unknown overhead) / size of generator. IB allows you to create generators past this limit with no complaint, but these generators will return random data and corrupt the database if incremented.

IB seems to limit generators to one page, but no range checking is done. This is particularly bad on databases with small page sizes which migrate from ODS 9 to ODS 10, since the size of generates doubles from 32 bit to 64 bit, seriously reducing the limit. On a 1024 page size, this limit is somewhere less than 128 generators.

Fixed? or resolved by knowledge?

SFID 227717.- **COBOL programs randomly return a -901 request sync. error**
COBOL programs randomly return a -901 request synchronization error. Alternative title: COBOL programs lose SQLCODE values during UPDATE:

```
UPDATE SET ... WHERE x=..
```

('Mass Update') error codes returned by the update are not returned to the program, instead the program will see a -901 request synchronization error.

The error was caused by bad code generated by GPRE. Fixed.

SFID 212328.- **IB Guardian leaked handles**

Fixed.

SFID 212263.- **command line isql ignores -user / -password with -a or -x**

Fixed.

SFID 421262.- **ISQL reports UDF BLOB parameter BY VALUE**

The BLOB UDFs in ibudf have been fixed in the Firebird distribution.

SFID 222563 isql extracts wrong store proc parameters with UNICODE

ISQL was reading rdb$field_length instead of rdb$character_length for procedure's parameters. In the case of table's fields, the correct information is read and presented. The engine itself was doing the right thing.

Fixed.

Known Issues

Core Engine

SFID
479483 Bad treatment of FIRST/SKIP in subselect
224810 DISTINCT propagates outside a VIEW
442140 Grant for Roles on Views not working
222376 Horrible plan with a lot of OR conditions
223514 IB crashes with two procedures intermixed.
217042 IB doesn't validate weird constructions
419065 Join on diffrent datatypes
221925 Left joining table to SP: ORDER BY makes fields NULL
223058 Multi-hop server ability broken Confirmed Bug
228135 NULL is returned as zero through a left join in simple VIEW
219525 No current record for fetch operation
221921 ORDER BY has no effect
225283 ORDER BY on a VIEW turns values in fields into NULL
213460 Registering Events w/certain configuration crashes IB Server
233025 Server hangs when executing Stored Proc more than once

221649 Unique index allowed on NULLABLE field
211781 Win32: Server don't close thread handles
233644 cannot specify PLAN in UPDATE statement

DSQL

SF ID
223516 Missing types in rdb$types

451944 Trigger Activate/Deactivate increases meta counter

GBAK

228431 gbak cannot restore backup made by IB v5

GPRE
416228 gpre generates invalid isc_vtov calls
223513 Ambiguity between tables and views in isql's SHOW commands.
223126 Misplaced collation when extracting metadata with isql
225219 isql -a: wrong order with domains based on table's fields
450404 isql uppercases role in the command line

InterClient

227414 Sometimes Interbase/Interserver won't grant connections

Security Issues

229237 Blank passwords poorly supported
229894 Client program can log on as any user.
222375 Grants overwrite previous rdb$security_classes entries
223128 SYSDBA can grant non existent roles