



Firebird 2.5 Release Notes

Helen Borrie (Collator/Editor)

2 July 2008 - Document v.025_05 - for Firebird 2.5 Alpha

Firebird 2.5 Release Notes

2 July 2008 - Document v.025_05 - for Firebird 2.5 Alpha
Helen Borrie (Collator/Editor)

Table of Contents

1. General Notes	1
Bug Reporting	1
Documentation	1
2. New in Firebird 2.5	2
Other New Features	2
Other SQL Language Additions	2
Data-handling Enhancements	2
Administrative Enhancements	2
API Additions	3
International Language Support	3
3. Changes to the Firebird API and ODS	4
ODS (On-Disk Structure) Changes	4
New ODS Number	4
API (Application Programming Interface) Extensions	4
Support for SQLSTATE Completion Codes	4
“Efficient Unprepare”	4
Cancel Operation Function	5
Shutdown Function for the Embedded Engine	5
4. Configuration Parameter Changes	6
Authentication	6
5. Changes in the Firebird Engine	7
New Threading Architecture	7
“Superclassic”	8
Thread-safe Client Library	9
Improvements	9
Immediate Detection of Disconnected Clients on Classic	9
Optimizations	9
DLL Loading for Windows Embedded Engine	10
UDFs Safeguard	10
Diagnostics	10
Metadata Improvements	10
6. Data Definition Language (DDL)	12
Quick Links	12
CREATE/ALTER/DROP USER	12
Syntaxes for Altering Views	13
Extension for CREATE VIEW	14
ALTER Mechanism for Computed Columns	14
Extension for GRANT and REVOKE	15
ALTER ROLE	16
Default COLLATION Attribute for a Database	16
ALTER CHARACTER SET Command	17
7. Data Manipulation Language (DML)	18
Quick Links	18
RegEx Search Support using SIMILAR TO	18
Hex Literal Support	23
New UUID Conversion Functions	24
Extension to LIST() Function	24
8. Procedural SQL (PSQL)	26

Quick Links	26
Autonomous Transactions	26
Borrow Database Column Type for a PSQL Variable	27
New Extensions to EXECUTE STATEMENT	28
Context Issues	28
External Queries from PSQL	30
EXECUTE STATEMENT with Dynamic Parameters	31
Examples Using EXECUTE STATEMENT	32
9. International Language Support (INTL)	36
Default COLLATION Attribute for a Database	36
ALTER CHARACTER SET Command	36
Other Improvements	36
Malformed UNICODE_FSS Characters Disallowed	36
Repair Switches for Malformed Strings	36
New Collations	37
10. Administrative Features	38
New RDB\$ADMIN System Role	38
Windows Domain Administrators	38
Monitoring Improvements	39
New MON\$ Metadata for ODS 11.2 Databases	39
Usage Notes	40
11. Command-line Utilities	42
gbak	42
Repair Switches for Malformed Strings	42
Preserve Character Set Default Collation	42
gpre (Precompiler)	42
Some Updates	42
12. Bugs Fixed	43
Firebird 2.5 Alpha 1	43
Core Engine/DSQL	43
Server Crashes	46
POSIX-specific	47
Windows-specific	48
Data Manipulation Language	48
Remote Interface/API	48
International Language Support	49
Database Monitoring/Administration	49
Security	50
Command-line Utilities	50
13. Firebird 2.5 Project Teams	52
Appendix A: Licence Notice	54

List of Tables

7.1. Character class identifiers	20
13.1. Firebird Development Teams	52

General Notes

Bug Reporting

- If you think you have discovered a new bug in this release, please make a point of reading the instructions for bug reporting in the article [How to Report Bugs Effectively](#), at the Firebird Project website.
- If you think a bug fix hasn't worked, or has caused a regression, please locate the original bug report in the Tracker, reopen it if necessary, and follow the instructions below.

Follow these guidelines as you attempt to analyse your bug:

1. Write detailed bug reports, supplying the exact build number of your Firebird kit. Also provide details of the OS platform. Include reproducible test data in your report and post it to our [Tracker](#).
2. You are warmly encouraged to make yourself known as a field-tester of this alpha by subscribing to the [field-testers' list](#) and posting the best possible bug description you can.
3. If you want to start a discussion thread about a bug or an implementation, please do so by subscribing to the [firebird-devel list](#). In that forum you might also see feedback about any tracker ticket you post regarding this alpha.

Documentation

You will find all of the README documents referred to in these notes in the doc sub-directory of your Firebird 2.5 Alpha 1 installation.

An automated "Release Notes" page in the Tracker provides lists and links for all of the Tracker tickets associated with this alpha. [Use this link](#).

--The Firebird Project

New in Firebird 2.5

The primary goal for Firebird 2.5 was to establish the basics for a new threading architecture that is almost entirely common to both the Superserver and Classic models, taking in lower level synchronization and thread safety generally.

Although SQL enhancements are not a primary objective of this release, for the first time, user management becomes accessible through SQL CREATE/ALTER/DROP USER statements and syntaxes for ALTER VIEW and CREATE OR ALTER VIEW are implemented. PSQL improvements include the introduction of autonomous transactions and ability to query another database via EXECUTE STATEMENT.

Other New Features

Other new features and improvements in this release include:

Other SQL Language Additions

- Regular expression support using the SIMILAR TO predicate
- ALTER COLUMN for computed columns
- Autonomous transactions within a PSQL module (stored procedure, trigger or dynamically executable PSQL block)
- Enhanced access to stored procedures in view definitions

Data-handling Enhancements

- New built-in functions for converting UUID CHAR(16) OCTETS strings to RFC4122-compliant format and vice versa
- Ability to pass 32-bit and 64-bit integers as hexadecimal in numeric literal and X-prefixed binary string literal formats

Administrative Enhancements

- New system role RDB\$ADMIN in the ODS 11.2 database allows SYSDBA to transfer its privileges to another user on a per-database basis
- More information in the monitoring tables
- Asynchronous cancellation of connections

API Additions

- Statements now return an SQL-2003 standard 5-alphanumeric SQLSTATE completion code
- New constant `DSQL_unprepare` available for use with `isc_dsql_free_statement` for efficient unpreparing of statements

International Language Support

- Default `COLLATE` clause for `CREATE DATABASE`
- `GBAK` restore switches `FIX_FSS_DATA` and `FIX_FSS_METADATA` to restore legacy databases with `UNICODE_FSS` data and/or metadata correctly without resorting to scripts and manual techniques
- Accent-insensitive collation for Unicode

Changes to the Firebird API and ODS

ODS (On-Disk Structure) Changes

On-disk structure (ODS) changes include the following:

New ODS Number

Firebird 2.5 creates databases with an ODS (On-Disk Structure) version of 11.2

API (Application Programming Interface) Extensions

Additions to the Firebird API include.-

Support for SQLSTATE Completion Codes

W. Oliver
D. Yemanov

Tracker reference [CORE-1761](#).

Statements now return an SQL-2003 standard 5-alphanumeric SQLSTATE completion code. Further information to come.

“Efficient Unprepare”

W. Oliver
D. Yemanov

Tracker reference [CORE-1741](#).

The new option *DSQL_unprepare* (numeric value 4) for the API routine *isc_dsql_free_statement()* allows the DSQL statement handle to survive the “unpreparing” of the statement.

Previously, the *isc_dsql_free_statement()* function supported only *DSQL_close* (for closing a named cursor) and *DSQL_drop* (which frees the statement handle).

The API addition is:

```
#define DSQL_close 1
#define DSQL_drop 2
#define DSQL_unprepare 4
```

Cancel Operation Function

Alex Peshkov

New *fb_cancel_operation()* API call, allowing cancellation of the current activity being performed by some kind of blocking API call in the given connection.

More information to come.

Shutdown Function for the Embedded Engine

Alex Peshkov

New *fb_shutdown()* API call. When used for the embedded engine, it terminates all the current activity, rolls back active transactions, disconnects active attachments and shuts down the engine gracefully.

Note

At this point (Alpha 1) it is implemented as a no-op for the other client libraries. This may change before the final release.

More information to come.

Configuration Parameter Changes

No new configuration parameters are added in this version. However, certain changes affect existing parameters, as follows:

Authentication

A. Peshkov

On Windows server platforms, since V.2.1, *Authentication* has been used for configuring the server authentication mode if you need it to be other than the default *mixed*.

The mode settings for v.2.5 are the same, viz.

- *trusted* makes use of Windows “trusted authentication” which, under the right conditions, may be the most secure way to authenticate on Windows.
- *native* sets the traditional Firebird server authentication mode, requiring users to log in using a user name and password defined in the security database.
- *mixed* allows both.

Under v.2.5, although the modes are unchanged, configuring 'mixed' or 'trusted' mode no longer confers SYS-DBA privileges to Windows domain administrators automatically by default. Please [read the notes in the Administrative Features chapter](#) regarding the new RDB\$ADMIN role in ODS 11.2 databases and auto-mapping SYSDBA privileges to domain administrators.

Changes in the Firebird Engine

The primary objective of this release was to refactor Firebird's threading architecture to take advantage of the symmetric multiprocessing (SMP) capabilities of multiprocessor hardware. This has a noticeable effect on the scalability of Superserver when multiple databases are being accessed simultaneously but its major effect is the emergence of the architectural “Superclassic” model that will underpin the fine-grained multi-threading under development for Firebird 3.

New Threading Architecture

Dmitry Yemanov
Vladyslav Khorsun
Alex Peshkov
also -
Nickolay Samofatov
Roman Simakov

For Superserver, the new architecture will be most obvious in two ways:

1. Superserver threads distributed evenly to available processors according the the database clients attach to.

Note

The default *CpuAffinity* setting still binds SuperServer to a single processor only. In order to scale better when working with multiple databases, this setting should be changed in `firebird.conf`.

The default value may change before the final release.

2. A slight improvement in scaling for single database usage on SMP hardware

It is with Classic that the effects are most evident:

1. Classic Server can now be multi-threaded. The one worker thread per process model remains but now it is possible to use additional threads for parallel tasks such as asynchronous shutdown, sweep, inter-process communications with the lock manager and more.
2. On POSIX, services in Classic also run in threads now, rather than in forked processes as previously.

Note

For Windows Classic, services became threadable in v.2.1.

3. The embedded library on POSIX, *libfbembed.so*, is now multi-thread-capable and thread-safe, so it can be used in multi-threaded applications.
4. Testing suggests that the performance of Classic in this version will be significantly faster than previous Classic versions.

“Superclassic”

This multi-threaded mode for Classic has been dubbed “Superclassic” for its capability to handle multiple worker threads—dedicated or pooled—inside a single server process. It shares all the usual Classic features, with a few differences:

- Safe, full shutdown is possible on any platform
- It performs better than Classic—by about 15-20%, according to TPC testing
- It uses fewer kernel resources (although not less memory)
- When a Superclassic process crashes, it takes all its connections with it
- Recognised limitations in the Services API for the Classic server, such as the inability to retrieve the list of attachments/active users, do not apply to SuperClassic.
- On POSIX, Superclassic does not require *[x]inetd*.

Usage Notes

Windows

On Windows, the same `fb_inet_server.exe` binary delivers either the Classic or the Superclassic working modes, according to switch settings. Classic is the default mode.

To use the Superclassic mode, add the `-m` (multi-threaded) switch to the command line, as follows:

When intending to run Superclassic *as an application*, use

```
fb_inet_server -a -m
```

When installing Superclassic to run *as a service*, you need to include an `-m` switch in the `instsvc.exe` command.

New Binary for POSIX

On POSIX, the new binary `fb_smp_server` is supplied for the Superclassic model. It contains the network listener, meaning it works similarly to `fbserver` with regard to attachment requests and does not require *[x]inetd*.

The multi-threaded engine used by `fb_smp_server` is the `libfbembed.so` library, in accordance with OSRI requirements. The Classic packages also include `fbguard` (the Guardian) which, for Classic, starts `fb_smp_server`, rather than `fbserver` as it does when the Superserver model is installed with Guardian.

Thread-safe Client Library

Dmitry Yemanov
Vladyslav Khorsun
Alex Peshkov

Tracker reference [CORE-707](#).

The client libraries, including the embedded one, can now be used in multi-threaded applications without any application-level synchronization.

Improvements

Improvements implemented include:

Immediate Detection of Disconnected Clients on Classic

Vladyslav Khorsun

The Classic server now detects immediately when a Classic process has been broken by a client disconnection. Its response is to terminate any pending activity, roll back the active transaction and close the network connection.

Tracker reference [CORE-818](#).

Optimizations

Important optimizations include:

Data Retrieval

Dmitry Yemanov

An optimization improves data retrieval performance for tables from which no fields are accessed.

Tracker reference [CORE-1598](#).

BLOB Memory Usage

Adriano dos Santos Fernandes

An optimization avoids memory consumption of `<page size>` bytes for each temporary BLOB created during assignment.

Tracker reference [CORE-1658](#).

DLL Loading for Windows Embedded Engine

Adriano dos Santos Fernandes

The root determination mechanism for the Windows embedded engine has been changed to avoid common problems that occur when an installation of the application structure encounters “DLL Hell”. Previously, the implicit root directory was the directory containing the user application's main executable file. Now it is the directory where the renamed *fbembed.dll* library is located.

Tracker reference [CORE-1814](#).

UDFs Safeguard

Adriano dos Santos Fernandes

Tracker reference [CORE-1937](#).

When a string UDF is written to return a pointer not allocated by the same runtime as the Firebird server is accessing, the presence of the `FREE_IT` keyword in its declaration corrupts memory and crashes the server. As a safeguard against such dysfunctional UDFs, the engine now

1. detects such UDFs and throws an exception
2. depends on the presence of the updated *ib_util* library in the path for all server models, including embedded

Diagnostics

Transaction Diagnostics

Claudio Valderrama

Better diagnostics and error reporting when TPB contents are malformed. More information to come.

Tracker reference [CORE-1600](#).

Access Privilege Error Messages

Alex Peshkov

Both table and column names are now reported when access privilege exceptions occur for a column.

Tracker reference [CORE-1234](#).

Metadata Improvements

Preserve Character Set Default Collation

Adriano dos Santos Fernandes

An improvement allows the current value of `RDB$DEFAULT_COLLATE_NAME` in the system table `RDB$CHARACTER_SETS` to survive the backup/restore cycle.

Tracker reference [CORE-789](#).

Chapter 6

Data Definition Language (DDL)

V.2.5 brings a few significant additions and enhancements to DDL.

Quick Links

- [CREATE/ALTER/DROP USER](#)
- [Syntaxes for Altering Views](#)
- [SSP Extension for CREATE VIEW](#)
- [ALTER Mechanism for Computed Columns](#)
- [GRANTED BY Extension for GRANT and REVOKE](#)
- [ALTER ROLE](#)
- [Default COLLATION Attribute for a Database](#)
- [ALTER CHARACTER SET](#)

CREATE/ALTER/DROP USER

Alex Peshkov

Tracker reference [CORE-696](#).

In v.2.5, Firebird finally has syntax to enable user accounts on the server to be managed by submitting SQL statements when logged in to a regular database.

Syntax Patterns

A user with SYSDBA privileges can add a new user:

```
CREATE USER <username> {PASSWORD 'password'}  
  [FIRSTNAME 'firstname']  
  [MIDDLENAME 'middlename']  
  [LASTNAME 'lastname'];
```

Note

The PASSWORD clause is required when creating a new user. It should be the initial password for that new user. (The user can change it later, using ALTER USER.)

A user with SYSDBA privileges can change one or more of the password and proper name attributes of an existing user. Non-privileged users can use this statement to alter only their own attributes.

```
ALTER USER <username>
  [PASSWORD 'password']
  [FIRSTNAME 'firstname']
  [MIDDLENAME 'middlename']
  [LASTNAME 'lastname'];
```

Note

At least one of PASSWORD, FIRSTNAME, MIDDLENAME or LASTNAME must be present.

ALTER USER does not allow the <username> to be changed. If a different <username> is required, the old one should be deleted (dropped) and a new one created.

A user with SYSDBA privileges can delete a user:

```
DROP USER <username>;
```

Restrictions

CREATE and DROP statements are available only for the SYSDBA or a user that has been granted the RDB \$ADMIN role in the security database. An ordinary user can ALTER his own password and elements of his proper name. An attempt to modify another user will fail.

Examples

```
CREATE USER alex PASSWORD 'test';
ALTER USER alex FIRSTNAME 'Alex' LASTNAME 'Peshkov';
ALTER USER alex PASSWORD 'IdQfA';
DROP USER alex;
```

Tip

Firebird 2.5 does not allow you to set up more than one security database on a server. From V.3.0, it is intended to be possible to have separate security databases for each database. For now, you can be connected to any database on the server (even *employee.fdb*) to update its one-and-only *security2.fdb*.

In future, it will be essential to send these requests from a database that is associated with the security database that is to be affected by them.

Syntaxes for Altering Views

Adriano dos Santos Fernandes

Previously, in order to alter a view definition, it was necessary to save the view definition off-line somewhere and drop the view, before recreating it with its changes. This made things very cumbersome, especially if there were dependencies. V.2.5 introduces syntaxes for ALTER VIEW and CREATE OR ALTER VIEW.

Tracker references are [CORE-770](#) and [CORE-1640](#).

ALTER VIEW

ALTER VIEW enables a view definition to be altered without the need to recreate (drop and create) the old version of the view and all of its dependencies.

CREATE OR ALTER VIEW

With CREATE OR ALTER VIEW, the view definition will be altered (as with ALTER VIEW) if it exists, or created if it does not exist.

Syntax Pattern

```
create [ or alter ] | alter } view <view_name>
  [ ( <field list> ) ]
as <select statement>
```

Example

```
create table users (
  id integer,
  name varchar(20),
  passwd varchar(20)
);

create view v_users as
  select name from users;

alter view v_users (id, name) as
  select id, name from users;
```

Extension for CREATE VIEW

Adriano dos Santos Fernandes

Tracker reference [CORE-886](#).

A selectable stored procedure can now be specified in the FROM clause of a view definition.

More information to come.

ALTER Mechanism for Computed Columns

Adriano dos Santos Fernandes

Tracker reference [CORE-1454](#).

A column defined as COMPUTED BY <expression> can now be altered using the ALTER TABLE...ALTER COLUMN syntax. This feature can be used only to change the <expression> element of the column definition to a different expression. It cannot convert a computed column to non-computed or vice versa.

Syntax Pattern

```
alter table <table-name>
  alter <computed-column-name>
  [type <data-type>]
  COMPUTED BY (<expression>);
```

Examples

```
create table test (
  n integer,
  dn computed by (n * 2)
);
commit;
alter table test
  alter dn computed by (n + n);
```

Extension for GRANT and REVOKE

Alex Peshkov

A GRANTED BY or GRANTED AS clause can now be optionally included in GRANT and REVOKE statements, enabling the grantor to be a user other than the CURRENT_USER (the default).

Syntax Pattern

```
grant <right> to <object>
  [ { granted by | as } [ user ] <username> ]
--
revoke <right> from <object>
  [ { granted by | as } [ user ] <username> ]
```

Tip

GRANTED BY and GRANTED AS are equivalent. GRANTED BY is the form recommended by SQL standard. We support GRANTED AS for compatibility with some other servers (Informix, for example).

Example

Logged in as SYSDBA:

```
create role r1; -- SYSDBA owns the role
/* next, SYSDBA grants the role to user1
   with the power to grant it to others */
grant r1 to user1 with admin option;
/* SYSDBA uses GRANTED BY to exercise
   user1's ADMIN OPTION */
grant r1 to public granted by user1;
```

In isql, we look at the effects:

```
SQL>show grant;
/* Grant permissions for this database */
GRANT R1 TO PUBLIC GRANTED BY USER1
GRANT R1 TO USER1 WITH ADMIN OPTION
SQL>
```

ALTER ROLE

Alex Peshkoff

Tracker reference [CORE-1660](#).

The new ALTER ROLE statement has a specialised function to control the assignment of SYSDBA permissions to Windows administrators during trusted authentication. It has no other purpose currently.

For usage details, see the topic in the Administrative Features chapter, entitled [Automatically Mapping RDB \\$ADMIN to a Windows User](#).

Default COLLATION Attribute for a Database

Adriano dos Santos Fernandes

Tracker references [CORE-1737](#) and [CORE-1803](#).

An ODS 11.2 or higher database can now have a default COLLATION attribute associated with the default character set, enabling all text column, domain and variable definitions to be created with the same collation order unless a COLLATE clause for a different collation is specified.

The COLLATION clause is optional. If it is omitted, the default collation order for the character set is used.

Tip

Note also that the default collation order for a character set used in a database can now also be changed, thanks to the introduction of syntax for [ALTER CHARACTER SET](#).

Syntax Pattern

```
create database <file name>
  [ page_size <page size> ]
  [ length = <length> ]
  [ user <user name> ]
  [ password <user password> ]
  [ set names <charset name> ]
  [ default character set <charset name>
    [ collation <collation name> ] ]
  [ difference file <file name> ]
```

Example

```
create database 'test.fdb'
```

```
default character set win1252
collation win_ptbr;
```

ALTER CHARACTER SET Command

Adriano dos Santos Fernandes

Tracker reference [CORE-1803](#).

New syntax introduced in this version, enabling the default collation for a character set to be set for a database.

The default collation is used when table columns are created with a given character set (explicitly, through a CHARACTER SET clause in the column or domain definition, or implicitly, through the default character set attribute of the database) and no COLLATION clause is specified.

Note

String constants also use the default collation of the connection character set.

Syntax Pattern

```
ALTER CHARACTER SET <charset_name>
SET DEFAULT COLLATION <collation_name>
```

Example

```
create database 'people.fdb'
  default character set win1252;

alter character set win1252
  set default collation win_ptbr;

create table person (
  id integer,
  name varchar(50) /* will use the database default
                  character set and the win1252
                  default collation */
);

insert into person
  values (1, 'adriano');
insert into person
  values (2, 'ADRIANO');

/* will retrieve both records
   because win_ptbr is case insensitive */
select * from person where name like 'A%';
```

Tip

[Another improvement](#) allows the current value of RDB\$DEFAULT_COLLATE_NAME in the system table RDB\$CHARACTER_SETS to survive the backup/restore cycle.

Chapter 7

Data Manipulation Language (DML)

In this chapter are the additions and improvements that have been added to the SQL data manipulation language subset in Firebird 2.5.

Quick Links

- [RegEx Search Support using SIMILAR TO](#)
- [Hex Literal Support](#)
- [New UUID Conversion Functions](#)
- [Extension to LIST\(\) Function](#)

RegEx Search Support using SIMILAR TO

Adriano dos Santos Fernandes

Tracker reference [CORE-769](#).

A new SIMILAR TO predicate is introduced to support regular expressions. The predicate's function is to verify whether a given SQL-standard regular expression matches a string argument. It is valid in any context that accepts Boolean expressions, such as WHERE clauses, CHECK constraints and PSQL IF() tests.

Syntax Patterns

```
<similar predicate> ::=
  <value> [ NOT ] SIMILAR TO <similar pattern> [ ESCAPE <escape character> ]

<similar pattern> ::= <character value expression: regular expression>

<regular expression> ::=
  <regular term>
  | <regular expression> <vertical bar> <regular term>

<regular term> ::=
  <regular factor>
  | <regular term> <regular factor>

<regular factor> ::=
  <regular primary>
  | <regular primary> <asterisk>
  | <regular primary> <plus sign>
  | <regular primary> <question mark>
  | <regular primary> <repeat factor>

<repeat factor> ::=
```

```

<left brace> <low value> [ <upper limit> ] <right brace>

<upper limit> ::= <comma> [ <high value> ]

<low value> ::= <unsigned integer>

<high value> ::= <unsigned integer>

<regular primary> ::=
    <character specifier>
    | <percent>
    | <regular character set>
    | <left paren> <regular expression> <right paren>

<character specifier> ::=
    <non-escaped character>
    | <escaped character>

<regular character set> ::=
    <underscore>
    | <left bracket> <character enumeration>... <right bracket>
    | <left bracket> <circumflex> <character enumeration>... <right bracket>
    | <left bracket> <character enumeration include>... <circumflex> <character enumeration excl
    <right bracket>

<character enumeration include> ::= <character enumeration>

<character enumeration exclude> ::= <character enumeration>

<character enumeration> ::=
    <character specifier>
    | <character specifier> <minus sign> <character specifier>
    | <left bracket> <colon> <character class identifier> <colon> <right bracket>

<character specifier> ::=
    <non-escaped character>
    | <escaped character>

<character class identifier> ::=
    ALPHA
    | UPPER
    | LOWER
    | DIGIT
    | SPACE
    | WHITESPACE
    | ALNUM

```

Note

1. <non-escaped character> is any character except <left bracket>, <right bracket>, <left paren>, <right paren>, <vertical bar>, <circumflex>, <minus sign>, <plus sign>, <asterisk>, <underscore>, <percent>, <question mark>, <left brace> and <escape character>.
2. <escaped character> is the <escape character> succeeded by one of <left bracket>, <right bracket>, <left paren>, <right paren>, <vertical bar>, <circumflex>, <minus sign>, <plus sign>, <asterisk>, <underscore>, <percent>, <question mark>, <left brace> or <escape character>.

Table 7.1. Character class identifiers

Identifier	Description	Note
ALPHA	All characters that are simple latin letters (a-z, A-Z)	Includes latin letters with accents when using accent-insensitive collation
UPPER	All characters that are simple latin uppercase letters (A-Z)	Includes lowercase letters when using case-insensitive collation
LOWER	All characters that are simple latin lowercase letters (a-z)	Includes uppercase letters when using case-insensitive collation
DIGIT	All characters that are numeric digits (0-9)	
SPACE	All characters that are the space character (ASCII 32)	
WHITESPACE	All characters that are whitespaces (vertical tab (9), newline (10), horizontal tab (11), carriage return (13), formfeed (12), space (32))	
ALNUM	All characters that are simple latin letters (ALPHA) or numeric digits (DIGIT)	

Usage Guide

Return true for a string that matches <regular expression> or <regular term>: <regular expression> <vertical bar> <regular term>

```
'ab' SIMILAR TO 'ab|cd|efg' -- true
'efg' SIMILAR TO 'ab|cd|efg' -- true
'a' SIMILAR TO 'ab|cd|efg' -- false
```

Match zero or more occurrences of <regular primary>: <regular primary> <asterisk>

```
' ' SIMILAR TO 'a*' -- true
'a' SIMILAR TO 'a*' -- true
'aaa' SIMILAR TO 'a*' -- true
```

Match one or more occurrences of <regular primary>: <regular primary> <plus sign>

```
' ' SIMILAR TO 'a+' -- false
'a' SIMILAR TO 'a+' -- true
'aaa' SIMILAR TO 'a+' -- true
```

Match zero or one occurrence of <regular primary>: <regular primary> <question mark>

```
' ' SIMILAR TO 'a?'      -- true
'a' SIMILAR TO 'a?'      -- true
'aaa' SIMILAR TO 'a?'    -- false
```

Match exact <low value> occurrences of <regular primary>: <regular primary> <left brace> <low value> <right brace>

```
' ' SIMILAR TO 'a{2}'    -- false
'a' SIMILAR TO 'a{2}'    -- false
'aa' SIMILAR TO 'a{2}'   -- true
'aaa' SIMILAR TO 'a{2}'  -- false
```

Match <low value> or more occurrences of <regular primary>: <regular primary> <left brace> <low value> <comma> <right brace>:

```
' ' SIMILAR TO 'a{2,}'   -- false
'a' SIMILAR TO 'a{2,}'   -- false
'aa' SIMILAR TO 'a{2,}'  -- true
'aaa' SIMILAR TO 'a{2,}' -- true
```

Match <low value> to <high value> occurrences of <regular primary> <regular primary> <left brace> <low value> <comma> <high value> <right brace>:

```
' ' SIMILAR TO 'a{2,4}'  -- false
'a' SIMILAR TO 'a{2,4}'  -- false
'aa' SIMILAR TO 'a{2,4}' -- true
'aaa' SIMILAR TO 'a{2,4}' -- true
'aaaa' SIMILAR TO 'a{2,4}' -- true
'aaaaa' SIMILAR TO 'a{2,4}' -- false
```

Match any (non-empty) character: <underscore>

```
' ' SIMILAR TO '_'       -- false
'a' SIMILAR TO '_'       -- true
'l' SIMILAR TO '_'       -- true
'al' SIMILAR TO '_'      -- false
```

Match a string of any length (including empty strings): <percent>

```
' ' SIMILAR TO '%'       -- true
'az' SIMILAR TO 'a%z'    -- true
'al23z' SIMILAR TO 'a%z' -- true
'azx' SIMILAR TO 'a%z'   -- false
```

Group a complete <regular expression> to use as one single <regular primary> as a sub-expression: <left paren> <regular expression> <right paren>

```
'ab' SIMILAR TO '(ab){2}' -- false
'aabb' SIMILAR TO '(ab){2}' -- false
```

```
'abab' SIMILAR TO '(ab){2}' -- true
```

Match a character identical to one of <character enumeration>: <left bracket> <character enumeration>... <right bracket>

```
'b' SIMILAR TO '[abc]' -- true
'd' SIMILAR TO '[abc]' -- false
'9' SIMILAR TO '[0-9]' -- true
'9' SIMILAR TO '[0-8]' -- false
```

Match a character not identical to one of <character enumeration>: <left bracket> <circumflex> <character enumeration>... <right bracket>

```
'b' SIMILAR TO '[^abc]' -- false
'd' SIMILAR TO '[^abc]' -- true
```

Match a character identical to one of <character enumeration include> but not identical to one of <character enumeration exclude>: <left bracket> <character enumeration include>... <circumflex> <character enumeration exclude>...

```
'3' SIMILAR TO '[:DIGIT:]^3' -- false
'4' SIMILAR TO '[:DIGIT:]^3' -- true
```

Match a character identical to one character included in <character class identifier>. Refer to the table of [Character Class Identifiers](#), above. May be used with <circumflex> to invert the logic (see above): <left bracket> <colon> <character class identifier> <colon> <right bracket>

```
'4' SIMILAR TO '[:DIGIT:]' -- true
'a' SIMILAR TO '[:DIGIT:]' -- false
'4' SIMILAR TO '[^[:DIGIT:]]' -- false
'a' SIMILAR TO '[^[:DIGIT:]]' -- true
```

Examples

```
create table department (
  number numeric(3) not null,
  name varchar(25) not null,
  phone varchar(14)
  check (phone similar to '\([0-9]{3}\) [0-9]{3}-[0-9]{4}' escape '\')
);

insert into department
  values ('000', 'Corporate Headquarters', '(408) 555-1234');
insert into department
  values ('100', 'Sales and Marketing', '(415) 555-1234');
insert into department
  values ('140', 'Field Office: Canada', '(416) 677-1000');

insert into department
  values ('600', 'Engineering', '(408) 555-123'); -- check constraint violation
```

```
select * from department
  where phone not similar to '\\([0-9]{3}\\) 555\\-#' escape '\\';
```

Hex Literal Support

Bill Oliver
Adriano dos Santos Fernandes

Tracker reference [CORE-1760](#).

Support for hexadecimal numeric and binary string literals has been introduced.

Syntax Patterns

```
<numeric hex literal> ::=
  { 0x | 0X } <hexit> [ <hexit>... ]

<binary string literal> ::=
  { x | X } <quote> [ { <hexit> <hexit> }... ] <quote>

<digit> ::=
  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<hexit> ::=
  <digit> | A | B | C | D | E | F | a | b | c | d | e | f
```

Numeric Hex Literals

- The number of <hexit> in the string cannot exceed 16.
- If the number of <hexit> is greater than eight, the constant data type is a signed BIGINT. If it is eight or less, the data type is a signed INTEGER.

Tip

That means 0xF0000000 is -268435456 and 0x0F0000000 is 4026531840.

Binary String Literals

The resulting string is defined as a CHAR($n/2$) CHARACTER SET OCTETS, where n is the number of <hexit>.

Examples

```
select 0x10, cast('0x0F0000000' as bigint)
  from rdb$database;
select x'deadbeef'
  from rdb$database;
```

New UUID Conversion Functions

Adriano dos Santos Fernandes

Tracker references [CORE-1656](#) and [CORE-1682](#).

Two new built-in functions, `UUID_TO_CHAR` and `CHAR_TO_UUID`, enable conversion between a UUID in the form of a CHAR(16) OCTETS string and the RFC4122-compliant form.

CHAR_TO_UUID()

The function `CHAR_TO_UUID()` converts the CHAR(32) ASCII representation of a UUID (XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX) to the CHAR(16) OCTETS representation, optimized for storage.

Syntax Model

```
CHAR_TO_UUID( <string> )
```

Example

```
select char_to_uuid('93519227-8D50-4E47-81AA-8F6678C096A1')
  from rdb$database;
```

UUID_TO_CHAR()

The function `UUID_TO_CHAR()` converts a CHAR(16) OCTETS UUID (as returned by the `GEN_UUID()` function) to the CHAR(32) ASCII representation (XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX).

Syntax Model

```
UUID_TO_CHAR( <string> )
```

Example

```
select uuid_to_char(gen_uuid())
  from rdb$database;
```

Extension to LIST() Function

Adriano dos Santos Fernandes

Tracker reference [CORE-1453](#)

A string expression is now allowed as the delimiter argument of the LIST() function.

Example

```
SELECT
  DISCUSSION_ID,
  LIST(COMMENT, ASCII_CHAR(13))
FROM COMMENTS
GROUP BY DISCUSSION_ID;
```

Chapter 8

Procedural SQL (PSQL)

Several significant changes appear in Firebird's procedural language (PSQL), the language set for triggers, stored procedures and dynamic executable blocks, especially with regard to new extensions to the capabilities of EXECUTE STATEMENT. This release also heralds the arrival of the “autonomous transaction”.

Quick Links

- [Autonomous Transactions](#)
- [Borrow Database Column Type for a PSQL Variable](#)
- [New Extensions to EXECUTE STATEMENT](#)
 - [Context Issues](#)
 - [Authentication](#)
 - [Transaction Behaviour](#)
 - [Inherited Access Privileges](#)
 - [External Queries from PSQL](#)
 - [EXECUTE STATEMENT with Dynamic Parameters](#)
 - [Examples Using EXECUTE STATEMENT](#)

Autonomous Transactions

Adriano dos Santos Fernandes

Tracker reference [CORE-1409](#).

This new implementation allows a piece of code to run in an autonomous transaction within a PSQL module. It can be handy for a situation where you need to raise an exception but do not want the database changes to be rolled back.

The new transaction is initiated with the same isolation level as the one from which it is launched. Any exception raised in a block within the autonomous transaction will cause changes to be rolled back. If the block runs through until its end, the transaction is committed.

Warning

Because the autonomous transaction is independent from the one from which is launched, you need to use this feature with caution to avoid deadlocks.

Syntax Pattern

```
IN AUTONOMOUS TRANSACTION
DO
  <simple statement | compound statement>
```

Example of Use

```
create table log (  
  logdate timestamp,  
  msg varchar(60)  
);  
  
create exception e_conn 'Connection rejected';  
  
set term !;  
  
create trigger t_conn on connect  
as  
begin  
  if (current_user = 'BAD_USER') then  
  begin  
    in autonomous transaction  
    do  
    begin  
      insert into log (logdate, msg) values (current_timestamp, 'Connection rejected');  
    end  
  
    exception e_conn;  
  end  
end!  
  
set term ;!
```

Borrow Database Column Type for a PSQL Variable

Adriano dos Santos Fernandes

Tracker reference [CORE-1356](#).

This feature extends the implementation in v.2 whereby domains became available as “data types” for declaring variables in PSQL. Now it is possible to borrow the data type of a column definition from a table or view for this purpose.

Syntax Pattern

```
data_type ::=  
  <builtin_data_type>  
  | <domain_name>  
  | TYPE OF <domain_name>  
  | TYPE OF COLUMN <table or view>.<column>
```

Note

TYPE OF COLUMN gets only the type of the column. Any constraints or default values defined for the column are ignored.

Examples

```
CREATE TABLE PERSON (  
  ID INTEGER,
```

```

NAME VARCHAR(40)
);

CREATE PROCEDURE SP_INS_PERSON (
  ID TYPE OF COLUMN PERSON.ID,
  NAME TYPE OF COLUMN PERSON.NAME
)
AS
DECLARE VARIABLE NEW_ID TYPE OF COLUMN PERSON.ID;
BEGIN
  INSERT INTO PERSON (ID, NAME)
  VALUES (:ID, :NAME)
  RETURNING ID INTO :NEW_ID;
END

```

New Extensions to EXECUTE STATEMENT

Unusually for our release notes, we begin this chapter with the full, newly extended syntax for the EXECUTE STATEMENT statement in PSQL and move on afterwards to explain the various new features and their usage.

```

[FOR] EXECUTE STATEMENT <query_text> [( <input_parameters> )]
  [ON EXTERNAL [DATA SOURCE] <connection_string>]
  [WITH {AUTONOMOUS | COMMON} TRANSACTION]
  [AS USER <user_name>]
  [PASSWORD <password>]
  [WITH CALLER PRIVILEGES]
  [INTO <variables>]

```

Note

The order of the optional clauses is not fixed so, for example, a statement based on the following model would be just as valid:

```

[ON EXTERNAL [DATA SOURCE] <connection_string>]
[WITH {AUTONOMOUS | COMMON} TRANSACTION]
[AS USER <user_name>]
[PASSWORD <password>]
[WITH CALLER PRIVILEGES]

```

Clauses cannot be duplicated.

Context Issues

If there is no ON EXTERNAL DATA SOURCE clause present, EXECUTE STATEMENT is normally executed within the CURRENT_CONNECTION context. This will be the case if the AS USER clause is omitted, or it is present with its <user_name> argument equal to CURRENT_USER.

However, if <user_name> is not equal to CURRENT_USER, then the statement is executed in a separate connection, established without Y-Valve and remote layers, inside the same engine instance.

Note

In the absence of an AS USER <user_name> clause, CURRENT_USER is the default.

Authentication

Where server authentication is needed for a connection that is different to CURRENT_CONNECTION, e.g., for executing an EXECUTE STATEMENT command on an external datasource, the AS USER and PASSWORD clauses are required. However, under some conditions, the PASSWORD may be omitted and the effects will be as follows:

1. On Windows, for the CURRENT_CONNECTION (i.e., no external data source), trusted authentication will be performed *if it is active* and the AS USER parameter is missing, null or equal to CURRENT_USER.
2. If the external data source parameter is present and its <connection_string> refers to the same database as the CURRENT_CONNECTION, the effective user account will be that of the CURRENT_USER.
3. If the external data source parameter is present and its <connection_string> refers to a different database than the one CURRENT_CONNECTION is attached to, the effective user account will be the operating system account under which the Firebird process is currently running.

In any other case where the PASSWORD clause is missing, only *isc_dpb_user_name* will be presented in the DPB (attachment parameters) and native authentication will be attempted.

Transaction Behaviour

The new syntax has an optional clause for setting the appropriate transaction behaviour: WITH AUTONOMOUS TRANSACTION and WITH COMMON TRANSACTION. WITH COMMON TRANSACTION is the default and does not need to be specified. Transaction lifetimes are bound to the lifetime of CURRENT_TRANSACTION and are committed or rolled back in accordance with the CURRENT_TRANSACTION.

The behaviour for WITH COMMON TRANSACTION is as follows:

- a. Causes any transaction in an external data source to be started with the same parameters as CURRENT_TRANSACTION; otherwise
- b. Executes the statement inside the CURRENT_TRANSACTION; or
- c. May use another transaction that is started internally in CURRENT_CONNECTION.

The WITH AUTONOMOUS TRANSACTION setting starts a new transaction with the same parameters as CURRENT_TRANSACTION. That transaction will be committed if the statement is executed without exceptions or rolled back if the statement encounters an error.

Inherited Access Privileges

Vladyslav Khorsun

Tracker reference [CORE-1928](#).

By design, the original implementation of EXECUTE STATEMENT isolated the executable code from the access privileges of the calling stored procedure or trigger, falling back to the privileges available to the

CURRENT_USER. In general, the strategy is wise, since it reduces the vulnerability inherent in providing for the execution of arbitrary statements. However, in hardened environments, or where privacy is not an issue, it could present a limitation.

The introduction of the optional clause WITH CALLER PRIVILEGES now makes it possible to have the executable statement inherit the access privileges of the calling stored procedure or trigger. The statement is prepared using any additional privileges that apply to the calling stored procedure or trigger. The effect is the same as if the statement were executed by the stored procedure or trigger directly.

Important

The WITH CALLER PRIVILEGES option is not compatible with the ON EXTERNAL DATA SOURCE option.

External Queries from PSQL

Vladyslav Khorsun

Tracker reference [CORE-1853](#).

EXECUTE STATEMENT now supports queries against external databases by inclusion of the ON EXTERNAL DATA SOURCE clause with its <connection_string> argument.

The <connection_string> Argument

The format of <connection_string> is the usual one that is passed through the API function `isc_attach_database()`, viz.

```
[<host_name><protocol_delimiter>]database_path
```

Character Set

The connection to the external data source uses the same character set as is being used by the CURRENT_CONNECTION context.

Access Privileges

If the external data source is on another server then the clauses AS USER <user_name> and PASSWORD <password> will be needed.

The clause WITH CALLER PRIVILEGES is a no-op if the external data source is on another server.

MORE INFORMATION REQUIRED. ROLES?

Note

Use of a two-phase transaction for the external connection is not available in V.2.5.

EXECUTE STATEMENT with Dynamic Parameters

Vladyslav Khorsun
Alex Peshkov

Tracker reference [CORE-1221](#).

The new extensions provide the ability to prepare a statement with dynamic input parameters (placeholders) in a manner similar to a parameterised DSQL statement. The actual text of the query itself can also be passed as a parameter.

Syntax Conventions

The mechanism employs some conventions to facilitate the run-time parsing and to allow the option of “naming” parameters in a style comparable with the way some popular client wrapper layers, such as Delphi, handle DSQL parameters. The API's own convention, of passing unnamed parameters in a predefined order, is also supported. However, named and unnamed parameters cannot be mixed.

The New Binding Operator

At this point in the implementation of the dynamic parameter feature, to avoid clashes with equivalence tests, it was necessary to introduce a new assignment operator for binding run-time values to named parameters. The new operator mimics the Pascal assignment operator:“:=”.

Syntax for Defining Parameters

```
<input_parameters> ::=
  <named_parameter> | <input_parameters>, <named_parameter>

<named_parameter> ::=
  <parameter name> := <expression>
```

Example for named input parameters

For example, the following block of PSQL defines both <query_text> and named <input_parameters> (<named_parameter>):

```
EXECUTE BLOCK AS
  DECLARE S VARCHAR(255);
  DECLARE N INT = 100000;
  BEGIN
  /* Normal PSQL string assignment of <query_text> */
  S = 'INSERT INTO TTT VALUES (:a, :b, :a)';

  WHILE (N > 0) DO
  BEGIN
  /* Each loop execution applies both the string value
     and the values to be bound to the input parameters */
```

```
EXECUTE STATEMENT (:S) (a := CURRENT_TRANSACTION, b := CURRENT_CONNECTION)
WITH COMMON TRANSACTION;
N = N - 1;
END
END
```

Example for **unnamed input parameters**

A similar block using a set of unnamed input parameters instead and passing constant arguments directly:

```
EXECUTE BLOCK AS
DECLARE S VARCHAR(255);
DECLARE N INT = 100000;
BEGIN
S = 'INSERT INTO TTT VALUES (?, ?, ?)';

WHILE (N > 0) DO
BEGIN
EXECUTE STATEMENT (:S) (CURRENT_TRANSACTION, CURRENT_CONNECTION, CURRENT_TRANSACTION);
N = N - 1;
END
END
END
```

Note

Observe that, if you use both <query_text> and <input_parameters> then the <query_text> must be enclosed in parentheses, viz.

```
EXECUTE STATEMENT (:sql) (p1 := 'abc', p2 := :second_param) ...
```

Examples Using EXECUTE STATEMENT

The following examples offer a sampler of ways that the EXECUTE STATEMENT extensions might be applied in your applications.

Test Connections and Transactions

A couple of tests you can try to compare variations in settings:

Test a) :Execute this block few times in the same transaction - it will create three new connections to the current database and reuse it in every call. Transactions are also reused.

```
EXECUTE BLOCK
RETURNS (CONN INT, TRAN INT, DB VARCHAR(255))
AS
DECLARE I INT = 0;
DECLARE N INT = 3;
DECLARE S VARCHAR(255);
BEGIN
SELECT A.MON$ATTACHMENT_NAME FROM MON$ATTACHMENTS A
```

```

WHERE A.MON$ATTACHMENT_ID = CURRENT_CONNECTION
INTO :S;

WHILE (i < N) DO
BEGIN
  DB = TRIM(CASE i - 3 * (I / 3)
    WHEN 0 THEN '\\.\' WHEN 1 THEN 'localhost:' ELSE '' END) || :S;

  FOR EXECUTE STATEMENT
  'SELECT CURRENT_CONNECTION, CURRENT_TRANSACTION
  FROM RDB$DATABASE'
  ON EXTERNAL :DB
  AS USER CURRENT_USER PASSWORD 'masterkey' -- just for example
  WITH COMMON TRANSACTION
  INTO :CONN, :TRAN
  DO SUSPEND;

  i = i + 1;
END
END

```

Test b) : Execute this block few times in the same transaction - it will create three new connections to the current database on every call.

```

EXECUTE BLOCK
  RETURNS (CONN INT, TRAN INT, DB VARCHAR(255))
AS
  DECLARE I INT = 0;
  DECLARE N INT = 3;
  DECLARE S VARCHAR(255);
BEGIN
  SELECT A.MON$ATTACHMENT_NAME
  FROM MON$ATTACHMENTS A
  WHERE A.MON$ATTACHMENT_ID = CURRENT_CONNECTION
  INTO :S;

  WHILE (i < N) DO
  BEGIN
    DB = TRIM(CASE i - 3 * (I / 3)
      WHEN 0 THEN '\\.\'
      WHEN 1 THEN 'localhost:'
      ELSE '' END) || :S;

    FOR EXECUTE STATEMENT
    'SELECT CURRENT_CONNECTION, CURRENT_TRANSACTION FROM RDB$DATABASE'
    ON EXTERNAL :DB
    WITH AUTONOMOUS TRANSACTION -- note autonomous transaction
    INTO :CONN, :TRAN
    DO SUSPEND;

    i = i + 1;
  END
END

```

Input Evaluation Demo

Demonstrating that input expressions evaluated only once:

```
EXECUTE BLOCK
  RETURNS (A INT, B INT, C INT)
AS
BEGIN
  EXECUTE STATEMENT (
    'SELECT CAST(:X AS INT),
           CAST(:X AS INT),
           CAST(:X AS INT)
     FROM RDB$DATABASE'
    (x := GEN_ID(G, 1))
    INTO :A, :B, :C;

  SUSPEND;
END
```

Insert Speed Test

Recycling our earlier examples for input parameter usage for comparison with the non-parameterised form of EXECUTE STATEMENT:

```
RECREATE TABLE TTT (
  TRAN INT,
  CONN INT,
  ID INT);

-- Direct inserts:

EXECUTE BLOCK AS
  DECLARE N INT = 100000;
BEGIN
  WHILE (N > 0) DO
  BEGIN
    INSERT INTO TTT VALUES (CURRENT_TRANSACTION, CURRENT_CONNECTION, CURRENT_TRANSACTION);
    N = N - 1;
  END
END

-- Inserts via prepared dynamic statement
-- using named input parameters:

EXECUTE BLOCK AS
  DECLARE S VARCHAR(255);
  DECLARE N INT = 100000;
BEGIN
  S = 'INSERT INTO TTT VALUES (:a, :b, :a)';

  WHILE (N > 0) DO
  BEGIN
    EXECUTE STATEMENT (:S)
      (a := CURRENT_TRANSACTION, b := CURRENT_CONNECTION)
    WITH COMMON TRANSACTION;
    N = N - 1;
  END
END

-- Inserts via prepared dynamic statement
-- using unnamed input parameters:
```

```
EXECUTE BLOCK AS
DECLARE S VARCHAR(255);
DECLARE N INT = 100000;
BEGIN
    S = 'INSERT INTO TTT VALUES (?, ?, ?)';

    WHILE (N > 0) DO
    BEGIN
        EXECUTE STATEMENT (:S) (CURRENT_TRANSACTION, CURRENT_CONNECTION, CURRENT_TRANSACTION);
        N = N - 1;
    END
END
```

Chapter 9

International Language Support (INTL)

Adriano dos Santos Fernandes

Some improvements appear in this release to tighten and enhance Firebird's handling capabilities for international language environment.

Default COLLATION Attribute for a Database

Databases of ODS 11.2 and higher can now optionally be created with a default collation associated with the default character set. For details, please see [Default COLLATION Attribute for a Database](#) in the DDL chapter.

ALTER CHARACTER SET Command

DDL syntax has been introduced to enable the default collation for a character set to be set at database level. For details, please see [ALTER CHARACTER SET Command](#) in the DDL chapter.

Other Improvements

DDL syntax has been introduced to enable the default collation for a character set to be set at database level. For details, please see [ALTER CHARACTER SET Command](#) in the DDL chapter.

Malformed UNICODE_FSS Characters Disallowed

Tracker reference [CORE-1600](#).

Malformed characters are no longer allowed in data for UNICODE_FSS columns.

Repair Switches for Malformed Strings

New restore switches were added to the *gbak* utility code for the purpose of repairing malformed UNICODE_FSS data and metadata by restoring a backup of the affected database. Details are in the [gbak section](#) of the Utilities chapter.

New Collations

UNICODE_CI_AI

Tracker reference [CORE-824](#).

UNICODE_CI_AI: case-insensitive, accent-insensitive collation added for UNICODE.

Chapter 10

Administrative Features

Certain improvements to Firebird's administrative features will be welcomed by many.

New RDB\$ADMIN System Role

Alex Peshkov

A new pre-defined system role RDB\$ADMIN has been added for transferring SYSDBA privileges to another user. Any user, when granted the role in a particular database, acquires SYSDBA-like rights when attaching to that database with the RDB\$ADMIN role specified.

To assign it, SYSDBA should log in to that database and grant this role to the user, in the same way he would grant any other role to a user.

The following example transfers SYSDBA privileges to users named User1 and Admins\ADMINS. The second user in our example is typical for Windows trusted authentication:

```
GRANT RDB$ADMIN TO User1;  
GRANT RDB$ADMIN TO "Admins\ADMINS";
```

Note

For Windows trusted authentication, a database can be set up to provide the RDB\$ADMIN role to Windows Administrators automatically. This is described in more detail presently.

Windows Domain Administrators

On POSIX hosts, the *root* user always had SYSDBA privileges, but the same was not possible for a domain administrator on Windows until Firebird 2.1. There, a configuration parameter, *Authentication*, was introduced whereby a user logged in as a Windows domain administrator could automatically gain server access with SYSDBA privileges through trusted user authentication. The mechanism for achieving that has changed with the introduction of the new system role and associated behaviour in v.2.5.

Automatically Mapping RDB\$ADMIN to a Windows User

The situation has not changed for the *root* user on POSIX but, on Windows, a domain administrator must now be granted the RDB\$ADMIN role in order to get SYSDBA access. By default, the SYSDBA must perform this GRANT manually for any user, including a domain administrator. However, the SYSDBA can configure it to happen automatically for Windows Administrators if the *Authentication* parameter in *firebird.conf* is 'mixed' or 'trusted'. A new ALTER ROLE syntax is implemented for this specialised purpose.

Auto-mapping Syntax

To configure a database to auto-grant the RDB\$ADMIN role to Administrators, use the following statement:

```
ALTER ROLE RDB$ADMIN  
SET AUTO ADMIN MAPPING;
```

To revert to the default setting, preventing administrators from getting SYSDBA privileges automatically, issue this statement:

```
ALTER ROLE RDB$ADMIN  
DROP AUTO ADMIN MAPPING;
```

Escalating RDB\$ADMIN Scope

Because *security2.fdb* is created as (or should be upgraded to) an ODS 11.2 database, it has the pre-defined RDB\$ADMIN role, too. SYSDBA can grant RDB\$ADMIN in *security2.fdb* to a user if that user needs the same rights as SYSDBA to administer all other users through *gsec* or the Services API, i.e., create and drop users or alter any user.

The auto-mapping facility described above is also applicable, if required.

Important

If the user attaches with a user database role passed in the DPB (connection parameters), it will not be replaced with RDB\$ADMIN, i.e., he/she will not get SYSDBA rights.

Monitoring Improvements

Dmitry Yemanov

Firebird 2.5 sees the enhancement of the “MON\$” database monitoring features introduced in V.2.1, with new tables delivering data about context variables and memory usage in ODS 11.2 and higher databases. Also, in these databases, it becomes possible to terminate a client connection from another connection through the MON\$ structures.

New MON\$ Metadata for ODS 11.2 Databases

Note

Please refer to the V.2.1 documentation for the ODS 11.1 metadata.

MON\$MEMORY_USAGE (current memory usage)

- MON\$STAT_ID (statistics ID)
- MON\$STAT_GROUP (statistics group)
 - 0: database
 - 1: attachment
 - 2: transaction
 - 3: statement
 - 4: call

- **MON\$MEMORY_USED** (number of bytes currently in use)
High-level memory allocations performed by the engine from its pools.
Can be useful for tracing memory leaks and for investigating unusual memory consumption and the attachments, procedures, etc. that might be responsible for it.
- **MON\$MEMORY_ALLOCATED** (number of bytes currently allocated at the OS level)
Low-level memory allocations performed by the Firebird memory manager.
These are bytes actually allocated by the operating system, so it enables the physical memory consumption to be monitored.

Note

Not all records have non-zero values. On the whole, only **MON\$DATABASE** and memory-bound objects point to non-zero “allocated” values. Small allocations are not allocated at this level, being redirected to the database memory pool instead.

- **MON\$MAX_MEMORY_USED** (maximum number of bytes used by this object)
- **MON\$MAX_MEMORY_ALLOCATED** (maximum number of bytes allocated from the operating system by this object)

MON\$CONTEXT_VARIABLES (known context variables)

- **MON\$ATTACHMENT_ID** (attachment ID)
Contains a valid ID only for session-level context variables.
Transaction-level variables have this field set to NULL.
- **MON\$TRANSACTION_ID** (transaction ID)
Contains a valid ID only for transaction-level context variables.
Session-level variables have this field set to NULL.
- **MON\$VARIABLE_NAME** (name of context variable)
- **MON\$VARIABLE_VALUE** (value of context variable)

Usage Notes

Examples

“Top 10” statements ranked according to their memory usage:

```
SELECT FIRST 10
  STMT.MON$ATTACHMENT_ID,
  STMT.MON$SQL_TEXT,
  MEM.MON$MEMORY_USED
FROM MON$MEMORY_USAGE MEM
  NATURAL JOIN MON$STATEMENTS STMT
  ORDER BY MEM.MON$MEMORY_USED DESC
```

To enumerate all session-level context variables for the current connection:

```
SELECT
  VAR.MON$VARIABLE_NAME ,
  VAR.MON$VARIABLE_VALUE
FROM MON$CONTEXT_VARIABLES VAR
  WHERE VAR.MON$ATTACHMENT_ID = CURRENT_CONNECTION
```

Terminating a Client

The MON\$ structures are, by design, read-only. Thus, user DML operations on them are prohibited. However, a mechanism is built in to allow deleting (only) of records in the MON\$STATEMENTS and MON\$ATTACHMENTS tables. The effect of this mechanism makes it possible, respectively, to cancel running statements and, for ODS 11.2 databases, to terminate client sessions.

To cancel all current activity for a specified connection:

```
DELETE FROM MON$STATEMENTS
  WHERE MON$ATTACHMENT_ID = 32
```

To disconnect all clients except the “Me” connection:

```
DELETE FROM MON$ATTACHMENTS
  WHERE MON$ATTACHMENT_ID <> CURRENT_CONNECTION
```

Note

- A statement cancellation attempt becomes a void operation (“no-op”) if the client has no statements currently running.
 - Upon cancellation, the execute/fetch API call returns the *isc_cancelled* error code.
 - Subsequent operations are allowed.
- Any active transactions in the connection being terminated will have their activities cancelled immediately and they are rolled back.
 - Once terminated, the client session receives the *isc_att_shutdown* error code.
 - Subsequent attempts to use this connection handle will cause network read/write errors.

Command-line Utilities

gbak

Repair Switches for Malformed Strings

Adriano dos Santos Fernandes

Tracker reference [CORE-1831](#).

The *gbak* utility has two new restore switches intended to repair malformed UNICODE_FSS character data and metadata, respectively, when restoring the backup of an affected database.

Switch Syntax

```
-FIX_FSS_D(ATA) <charset> -- fix malformed UNICODE_FSS data  
-FIX_FSS_M(ETADATA) <charset> -- fix malformed UNICODE_FSS metadata
```

Preserve Character Set Default Collation

Adriano dos Santos Fernandes

An improvement allows the current value of RDB\$DEFAULT_COLLATE_NAME in the system table RDB\$CHARACTER_SETS to survive the backup/restore cycle.

Tracker reference [CORE-789](#).

gpre (Precompiler)

Some Updates

Stephen Boyd
Adriano dos Santos Fernandes

Tracker reference [CORE-1527](#).

GPRES now supports the IS NOT DISTINCT predicate and CASE/NULLIF/COALESCE/SUBSTRING functions, as well as the whole set of CURRENT_* context variables.

Bugs Fixed

Firebird 2.5 Alpha 1

The following bugs are reported as fixed in the Alpha 1 release:

Core Engine/DSQL

([CORE-1421](#)) SuperServer could not shut down immediately if the shutdown request followed a failed login attempt.

fixed by A. Peshkov

~ ~ ~

([CORE-1907](#)) Dropping and adding a domain constraint in the same transaction would leave incorrect dependencies.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1905](#)) Hash sign (#) in filenames in aliases.conf was being handled incorrectly.

fixed by C. Valderrama

~ ~ ~

([CORE-1887](#)) Newly created databases had wrong access rights.

fixed by A. Peshkov

~ ~ ~

([CORE-1869](#)) Roles granting/revoking logic differed between v.2.0 and v.2.1.

fixed by A. Peshkov

~ ~ ~

([CORE-1841](#)) If some VIEW used derived tables and long table names/aliases, it was possible to overflow RDB\$VIEW_RELATIONS.RDB\$CONTEXT_NAME.

fixed by V. Khorsun

~ ~ ~

([CORE-xxxx](#)) Text

fixed by A. Peshkov

~ ~ ~

([CORE-1840](#)) Each DDL execution would cause a small memory leak.

fixed by D. Yemanov

~ ~ ~

([CORE-1838](#)) SET STATISTICS INDEX on an index for a GTT could wrongly change the index id by the maximum available number for the database page size.

fixed by V. Khorsun

~ ~ ~

([CORE-1830](#)) Possible index corruption with multiple updates of the same record in the same transaction with savepoints being used.

fixed by V. Khorsun

~ ~ ~

([CORE-1817](#)) The RelaxedAliasChecking parameter was having no effect with regard to RDB\$DB_KEY.

fixed by V. Khorsun

~ ~ ~

([CORE-1811](#)) Parser reacted incorrectly to the unquoted usage of the keyword "VALUE".

fixed by D. Yemanov

~ ~ ~

([CORE-1798](#)) RDB\$DB_KEY was being evaluated as NULL in INSERT ... RETURNING.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1797](#)) OLD/NEW.RDB\$DB_KEY returned incorrect result in triggers.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1784](#)) Error with EXECUTE PROCEDURE inside EXECUTE STATEMENT.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1777](#)) Conflicting table reservation specifications were being allowed in the TPB.

fixed by C. Valderrama

~ ~ ~

([CORE-1775](#)) Security checking was performing poorly during prepare.

fixed by V. Khorsun

~ ~ ~

([CORE-1770](#)) Bugcheck 291 in DDL.

fixed by A. Peshkov

~ ~ ~

([CORE-1735](#)) Behaviour problem with SET DEFAULT action argument in referential integrity triggers.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1731](#)) Sometimes engine would hang for several minutes, using 1000% CPU load but with no I/O activity.

fixed by V. Khorsun

~ ~ ~

([CORE-1730](#)) Problems arose if one of the directories specified in the TempDirectories config setting was not available.

fixed by D. Yemanov

~ ~ ~

([CORE-1724](#)) Common table expressions could not be used in computed columns nor in quantified predicates (IN / ANY / ALL).

fixed by V. Khorsun

~ ~ ~

([CORE-1694](#)) Bug in CREATE/ALTER database trigger, where comments were in Russian.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1693](#)) Error in EXECUTE STATEMENT inside CONNECT / TRANSACTION START triggers.

fixed by A. dos Santos Fernandes, D. Yemanov

~ ~ ~

([CORE-1689](#)) “There are <n> dependencies” error message shows the wrong count of dependent objects

fixed by C. Valderrama

~ ~ ~

([CORE-1357](#)) DummyPacketInterval mechanism was broken.

fixed by D. Yemanov

~ ~ ~

([CORE-1307](#)) Switch -s of fb_inet_server was not being processed correctly

fixed by A. Peshkov

~ ~ ~

([CORE-479](#)) Grants would overwrite previous entries in RDB\$SECURITY_CLASSES.

fixed by A. Peshkov

~ ~ ~

Server Crashes

([CORE-1930](#)) Possible server crash if procedure was altered to have no outputs and dependent procedures were not recompiled

fixed by V. Khorsun

~ ~ ~

([CORE-1919](#)) Memory corruptions in EXECUTE STATEMENT could crash the server.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1884](#)) Random crashes using stored procedures with expressions as default values for input parameters.

fixed by V. Khorsun

~ ~ ~

([CORE-1839](#)) Server could crash when sorting by a field that was calculated using a recursive CTE.

fixed by A. Peshkov

~ ~ ~

([CORE-1793](#)) Server crashes at prepare of query with unused parameterized CTE.

fixed by V. Khorsun

~ ~ ~

([CORE-1512](#)) Server crashes due to the wrong parsing of the DEFAULT clause.

fixed by D. Yemanov

~ ~ ~

POSIX-specific

([CORE-1909](#)) Garbage in firebird.log on linux/amd64

fixed by A. Peshkov

~ ~ ~

([CORE-1885](#)) CREATE COLLATION caused lost connection under Posix.

fixed by A. dos Santos Fernandes, A. Peshkov

~ ~ ~

([CORE-1854](#)) Value of CURRENT_USER might not be in upper case when using Unix OS authentication.

fixed by A. Peshkov

~ ~ ~

([CORE-1826](#)) changeRunUser.sh and restoreRootRunUser.sh scripts were not changing the run user in the init.d scripts.

fixed by A. Peshkov

~ ~ ~

([CORE-1818](#)) Temporary files used for temporary page spaces were not deleted after use on Posix platforms.

fixed by A. Peshkov

~ ~ ~

([CORE-1807](#)) Server was being assigned to a non-canonical port after abnormal termination.

fixed by A. Peshkov

~ ~ ~

([CORE-1766](#)) Owner and group of isc_monitor1 file on Linux classic server were incorrect.

fixed by A. Peshkov

~ ~ ~

([CORE-1671](#)) atexit() calls in client libraries cause segfaults if the libraries were used in dlopen'ed modules.

fixed by A. Peshkov

~ ~ ~

Windows-specific

([CORE-1820](#)) Setup program was failing to detect a running server.

fixed by P. Reeves, D. Yemanov

~ ~ ~

([CORE-1105](#), [CORE-1390](#), [CORE-1566](#) & [CORE-1639](#)) Aliases would not work properly for XNET connections.

fixed by D. Yemanov

~ ~ ~

Data Manipulation Language

([CORE-1910](#)) Invalid fields were accepted in the insert clause for MERGE.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1859](#)) Arithmetic overflow or division by zero could occur in MAX() function.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1828](#)) Error with ABS() function in dialect 1.

fixed by A. dos Santos Fernandes

~ ~ ~

Remote Interface/API

([CORE-1868](#)) Client library was crashing inside `isc_dsqli_free_statement()`.

fixed by A. Peshkov

~ ~ ~

([CORE-1763](#)) The client library was not setting the options `SO_KEEPALIVE` nor `TCP_NODELAY` for the socket at connection.

fixed by V. Khorsun

~ ~ ~

([CORE-1755](#) and [CORE-1756](#)) A couple of crash scenarios could occur in `isc_start_transaction()`.

fixed by D. Kovalenko, A. Peshkov

~ ~ ~

([CORE-1726](#)) Failure in isc_service_start().

fixed by A. Peshkov

~ ~ ~

([CORE-1079](#)) Every attach of fbclient/fbembed library to the host process would leak 64KB of memory.

fixed by A. Peshkov

~ ~ ~

International Language Support

([CORE-1802](#)) Some issues were reported concerning maximum key size using the PXW_CS collation.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1774](#)) A problem appeared with collate ES_ES_CI_AI.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1254](#)) Problem with DISTINCT and insensitive collations.

fixed by A. dos Santos Fernandes

~ ~ ~

Database Monitoring/Administration

([CORE-1890](#)) Database monitoring process could hang under high load.

fixed by D. Yemanov

~ ~ ~

([CORE-1881](#)) Database monitoring could crash the server or badly affect its page locking logic.

fixed by D. Yemanov

~ ~ ~

([CORE-1728](#)) Database monitoring would not work after a fresh Linux install.

fixed by A. Peshkov

~ ~ ~

Security

([CORE-1845](#)) Some standard calls would show the server installation directory to regular users.

fixed by A. Peshkov

~ ~ ~

([CORE-1810](#)) Some issues appeared concerning usernames with '.' characters.

fixed by A. Peshkov

~ ~ ~

Command-line Utilities

isql

([CORE-1891](#)) SHOW VIEW would show nonsense information for view fields with expressions.

fixed by A. dos Santos Fernandes

~ ~ ~

([CORE-1875](#)) Errors in scripts with CURRENT_DATE.

fixed by V. Khorsun

~ ~ ~

([CORE-1862](#)) Extracted script was unusable with interdependent selectable procedures in FB 2.1

fixed by C. Valderrama

~ ~ ~

([CORE-1782](#)) Isql would crash when fetching data for a column having an alias longer than 30 characters.

fixed by D. Yemanov

~ ~ ~

([CORE-1749](#)) DDL statement with AUTODDL ON was not showing statistics.

fixed by D. Yemanov

~ ~ ~

([CORE-1507](#)) ISQL linecount facility in scripts goes out of sync after an INPUT command.

fixed by C. Valderrama

~ ~ ~

([CORE-1363](#)) ISQL would crash when the string converted from a double was longer than 23 bytes.

fixed by C. Valderrama

~ ~ ~

gsec

([CORE-1680](#)) Gsec DISPLAY was showing only a few of the first users when the security databases contained more than 50 users.

fixed by A. Peshkov

~ ~ ~

gbak

([CORE-1911](#)) Backup and restore were not thread-safe when using the Services API.

fixed by C. Valderrama

~ ~ ~

([CORE-1843](#)) Gbak with Service Manager would not allow paths with spaces.

fixed by A. Peshkov

~ ~ ~

([CORE-1703](#)) Delays/lockups when the gbak output was redirected to another process.

fixed by D. Yemanov

~ ~ ~

nbackup

([CORE-1876](#)) Incremental backups with NBACKUP were broken in v.2.1.

fixed by N. Samofatov

~ ~ ~

Firebird 2.5 Project Teams

Table 13.1. Firebird Development Teams

Developer	Country	Major Tasks
Dmitry Yemanov	Russian Federation	Full-time database engineer/implementor, core team leader
Alex Peshkov	Russian Federation	Full-time security features coordinator; buildmaster; porting authority
Claudio Valderrama	Chile	Code scrutineer; bug-finder and fixer; ISQL enhancements; UDF fixer, designer and implementor
Vladyslav Khorsun	Ukraine	Full-time DB engineer, SQL feature designer/implementor
Arno Brinkman	The Netherlands	Indexing and Optimizer enhancements; new DSQL features
Adriano dos Santos Fernandes	Brazil	New international character-set handling; text and text BLOB enhancements; new DSQL features; code scrutineering
Nickolay Samofatov	Russian Federation	Engine contributions
Roman Simakov	Russian Federation	Engine contributions
Bill Oliver	U.S.A.	Vulcan fork development, engine contributions
Paul Beach	France	Release Manager; HP-UX builds; MacOS Builds; Solaris Builds
Pavel Cisar	Czech Republic	QA tools designer/coordinator
Philippe Makowski	France	QA tester
Paul Reeves	France	Win32 installers and builds
Roman Rokytskyy	Germany	Jaybird implementor and co-coordinator
Evgeny Putilin	Russian Federation	Java stored procedures implementation
Jiri Cincura	Czech Republic	Developer and coordinator of .NET providers
Vladimir Tsvigun	Ukraine	Developer and coordinator of ODBC/JDBC driver for Firebird
Stephen Boyd	Canada	GPRES contributions

Firebird 2.5 Project Teams

Developer	Country	Major Tasks
Paul Vinkenoog	The Netherlands	Coordinator, Firebird documentation project; documentation writer and tools developer/implementor
Norman Dunbar	U.K.	Documentation writer
Pavel Menshchikov	Russian Federation	Documentation translator
Tomneko Hayashi	Japan	Documentation translator
Umberto (Mimmo) Masotti	Italy	Documentation translator
Helen Borrie	Australia	Release notes editor; Chief of Thought Police
also		
Université du Littoral Côte d'Opale Masters students	France	QA tests development

Appendix A: Licence Notice

The contents of this Documentation are subject to the Public Documentation License Version 1.0 (the “License”); you may only use this Documentation if you comply with the terms of this License. Copies of the License are available at <http://www.firebirdsql.org/pdfmanual/pdl.pdf> (PDF) and <http://www.firebirdsql.org/manual/pdl.html> (HTML).

The Original Documentation is entitled *Firebird 2.5 Release Notes*.

The Initial Writer of the Original Documentation is: Helen Borrie. Persons named in attributions are Contributors.

Copyright (C) 2004–2008. All Rights Reserved. Initial Writer contact: helebor at users dot sourceforge dot net.