



Firebird 5.0.3 Release Notes

Firebird Project: Core Developers, Mark Rotteveel

Version 0503-02, 11 July 2025

Support Firebird's Future!

Your contributions are vital to ensuring Firebird continues to thrive as a powerful, open-source relational database. By supporting the Firebird Foundation, you're directly contributing to the ongoing development, innovation, and stability of Firebird.

Why Your Support Matters

Firebird relies on stable funding to maintain its core infrastructure, deliver critical updates, and introduce new features. Your contributions help secure the resources needed to plan long-term development and keep Firebird at the forefront of database technology.

Exclusive Benefits for Supporters

By supporting Firebird through a **subscription or sponsorship**, you not only help sustain its development but also receive exclusive benefits:

- **Associates:** Enjoy early access to [EmberWings Magazine](#), discounts on certification exams, special offers, and more.
- **Partners:** Gain advanced benefits, including exclusive technical content, participation in release planning, and enhanced perks.
- **High-Level Sponsors:** Receive priority access, tailored recognition, and a customized collaboration experience.

Flexible Ways to Contribute

Choose the level of involvement that works best for you:

- Regular contributions via monthly (starting at €10) or annual subscription programs (starting at €100).
- One-time donations to show your support with memorable gifts as tokens of our gratitude.

If you rely on Firebird for your applications, consider supporting its development. Every contribution — big or small — makes a difference!

[Learn how you can support Firebird today.](#)

Table of Contents

1. General Notes	6
Compatibility with Older Versions	6
Sub-release V5.0.3	6
V5.0.3 Improvements	6
Sub-release V5.0.2	8
V5.0.2 Improvements	8
Sub-release V5.0.1	9
V5.0.1 Improvements	9
Bug Reporting	10
Documentation	11
2. New In Firebird 5.0	12
Summary of New Features	12
Complete In Firebird 5.0 Release Candidate 2	12
Complete In Firebird 5.0 Release Candidate 1	13
Complete In Firebird 5.0 Beta 1	16
3. Changes in the Firebird Engine	20
Quick Links	20
Support for parallel operations	20
Inline minor ODS upgrade	21
More cursor-related details in the plan output	22
Denser compression of records	23
Compiled statement cache	23
SQL and PSQL profiler	23
Package routines	26
Snapshot tables	29
Auxiliary views	32
RDB\$BLOB_UTIL package	35
Package routines	35
Examples	37
4. Changes to the Firebird API and ODS	40
ODS (On-Disk Structure) Changes	40
New Minor ODS Number	40
New System Tables	40
New Columns in System Tables	40
Application Programming Interfaces	40
Main API Extensions	40
Extensions to various getInfo() Methods	41
Services API Extensions	41

Inline blobs transmission in the wire protocol	42
5. Reserved Words and Changes	44
New Keywords in Firebird 5.0	44
Non-reserved	44
6. Configuration Additions and Changes	45
Parameters for Parallel Operations	45
MaxParallelWorkers	45
ParallelWorkers	45
Other Parameters	45
MaxStatementCacheSize	45
OnDisconnectTriggerTimeout	45
DefaultProfilerPlugin	45
OptimizeForFirstRows	45
OuterJoinConversion	46
SubQueryConversion	46
Changed configuration parameters	46
WireCryptPlugin	46
Replication Configuration Additions and Changes	46
cascade_replication	46
Allow macros in replication.conf	46
Removed configuration parameters	46
RemotePipeName	46
TcpLoopbackFastPath	47
7. Security	48
System privilege PROFILE_ANY_ATTACHMENT	48
Trace events before a valid security context is established	48
8. Management Statements	50
SET OPTIMIZE	50
9. Data Definition Language (DDL)	51
Quick Links	51
Support for partial indices	51
COMMENT ON MAPPING	52
10. Data Manipulation Language (DML)	53
Quick Links	53
SKIP LOCKED clause	53
Support for WHEN NOT MATCHED BY SOURCE in the MERGE statement	55
Support multiple rows for DML RETURNING	56
Allow parenthesized query expressions	56
Support PLAN and ORDER BY on MERGE	56
Support PLAN, ORDER BY and ROWS on UPDATE OR INSERT	57
OPTIMIZE FOR Clause	57

Changes to literals	58
Full SQL standard character string literal syntax	58
Full SQL standard binary string literal syntax	58
Improvements to IN	59
New Expressions and Built-in Functions	59
UNICODE_CHAR and UNICODE_VAL	59
QUARTER added to EXTRACT, FIRST_DAY and LAST_DAY	60
11. Procedural SQL (PSQL)	61
Allow subroutines to access variables/parameters defined at the outer/parent level	61
12. Monitoring & Command-line Utilities	62
Monitoring	62
<i>nbackup</i>	63
Clean Backup History	63
<i>isql</i>	63
Unify display of system procedures and functions & packages with other system objects	63
Display statement BLR	63
Replication information added to SHOW output	64
Display per-table statistics	64
<i>gbak</i>	65
Parallel backup/restore	65
<i>gfix</i>	66
Parallel sweep and ICU dependencies rebuild	66
ODS upgrade	66
13. Compatibility Issues	67
Migrating the security database from Firebird 4.0 to 5.0	67
SQL	67
Multi-row RETURNING behaviour	67
Removal of WNET protocol	67
Removal of QLI	68
Removal of fbclient_bor.lib	68
14. Bugs Fixed	69
Firebird 5.0.3 Release: Bug Fixes	69
Core Engine	69
Server Crashes/Hang-ups	71
Utilities	71
Firebird 5.0.2 Release: Bug Fixes	72
Core Engine	72
Firebird 5.0.1 Release: Bug Fixes	77
Core Engine	77
Server Crashes/Hangups	82
Utilities	84

Firebird 5.0 Release Candidate 2: Bug Fixes	85
Core Engine	85
Server Crashes/Hangups	87
Utilities	88
Firebird 5.0 Release Candidate 1: Bug Fixes	89
Core Engine	89
Server Crashes/Hangups	94
Utilities	95
Firebird 5.0 Beta 1 Release: Bug Fixes	95
Core Engine	95
Server Crashes/Hangups	97
Utilities	97
15. Firebird 5.0 Project Teams	98
Appendix A: Licence Notice	99

Chapter 1. General Notes

Thank you for choosing Firebird 5.0. We cordially invite you to test it hard against your expectations and engage with us in identifying and fixing any bugs you might encounter.

ODS (On-Disk Structure) 13.1 is introduced. It's a minor ODS upgrade, so databases in ODS 13.0 (created by Firebird 4.0) may still be opened with a Firebird 5.0 server (with some new features being unavailable), but databases in older ODS cannot be opened.



Databases created in pre-release builds of Firebird 5.0 may be inaccessible in the 5.0.0 final release and higher and have to be recreated. The ODS was changed a few times during the development cycle and the Firebird Project generally does not guarantee ODS being stable before the Beta stage.

In addition, 5.0.0 RC1 had an ODS change affecting the profiler, specifically with nullability and default values of its procedures and functions. If you intend to use (or have already used) the profiler, it is advisable to recreate the database.

Also, if you used the profiler in a pre-RC1 build, and don't intend to recreate the database, you will need to drop the tables and views prefixed with `PLG$PROF` due to changes in these tables (which are not considered part of the ODS). The next use of the profiler will recreate these tables and views.

The engine library is still named `engine13.dll` (Windows) and `libEngine13.so` (POSIX). The security database is named `security5.fdb`. Binaries layout and configuration are unchanged from Firebird 4.



That said, you can copy the Firebird engine library from the Firebird 3.0 distribution package (named `engine12.dll` (Windows) and `libEngine12.so` (POSIX), and located inside the `/plugins` sub-directory) to continue working with databases in ODS 12 without needing a backup/restore. However, new features introduced with Firebird 4.0 and Firebird 5.0 will not be accessible.

Compatibility with Older Versions

Known incompatibilities are detailed in the [Compatibility Issues](#) chapter.

Sub-release V.5.0.3

Bugs reported and fixed prior to the version 5.0.3 release are listed [HERE](#).

V.5.0.3 Improvements

#8598 — Don't fire referential integrity triggers if primary or unique keys haven't changed

Implemented by Vlad Khorsun

[#8523](#) — Avoid internal exception in *fbclient* during first `isc_attach_database`

Implemented by Vlad Khorsun

[#8522](#) — Avoid internal exception in *fbclient* during `isc_detach_database`

Implemented by Vlad Khorsun

[#8513](#) — Makes `MON$COMPILED_STATEMENTS` and `MON$STATEMENTS` share blobs with text and plan content of the same statement

Implemented by Vlad Khorsun

[#8447](#) — Avoid index scan for lower/upper bounds containing NULL keys

Implemented by Dmitry Yemanov

[#8433](#) — Improve code of class `BePlusTree`

Implemented by Vlad Khorsun

[#8421](#) — Add pointers tree to `TempSpace` class

Implemented by Andrey Kravchenko, Vlad Khorsun

[#8318](#) — Wire protocol improvement: send small blobs inline

For more details, see [Inline blobs transmission in the wire protocol](#).

Implemented by Vlad Khorsun



This enhancement was sponsored by the donations collected in the 21st Firebird Developers Day, many thanks to everyone involved!

[#8278](#) — Avoid index lookup for a NULL key if the condition is known to always be FALSE in this case

Implemented by Dmitry Yemanov

[#6413](#) — Data pages of newly *gbak* restored databases should be marked as "swept" [CORE6164]

Implemented by Vlad Khorsun

Sub-release V.5.0.2

Bugs reported and fixed prior to the version 5.0.2 release are listed [HERE](#).

V.5.0.2 Improvements

[#8356](#) — Make *trace* use HEX representation for parameter values of types [VAR]CHAR CHARACTER SET OCTETS and [VAR]BINARY

Implemented by Vlad Khorsun

[#8353](#) — Report unique usernames for `isc_info_user_names`

Implemented by Vlad Khorsun

[#8323](#) — Add `AUTO RELEASE TEMP BLOBID` transaction option

Implemented by Ilya Eremin

[#8310](#) — Collect network statistics and make it available for user applications

Implemented by Vlad Khorsun

[#8307](#) — Wire protocol improvement: prefetch blob info and some data when opening a blob

Implemented by Vlad Khorsun



This enhancement was sponsored by the donations collected in the 21st Firebird Developers Day, many thanks to everyone involved!

[#8291](#) — NULLs should be skipped during index navigation when there's no lower bound and matched conditions are known to ignore NULLs

Implemented by Dmitry Yemanov

[#8273](#) — Reorganize public headers

Implemented by Adriano dos Santos Fernandes

[#8256](#) — *Win_SSPI* plugin should try *Negotiate* before *NTLM*

Implemented by Vlad Khorsun

[#8197](#) — Add generated files for OO API for C language to distribution

Implemented by Alexander Peshkov

[#8161](#) — Cardinality estimation should use primary record versions only

Implemented by Vlad Khorsun

[#7269](#) — Database restore must make every effort on activating deferred indexes

Implemented by Vlad Khorsun, Dima

Sub-release V.5.0.1

Bugs reported and fixed prior to the version 5.0.1 release are listed [HERE](#).

V.5.0.1 Improvements

[#8181](#) — Ensure the standalone CS listener on Linux uses the *SO_REUSEADDR* socket option

Implemented by Dmitry Yemanov

[#8165](#) — Added shutdown handler for *Classic Server*

Implemented by Alexander Peshkov, Alexander Zhdanov

[#8104](#) — More efficient evaluation of expressions like *RDB\$DB_KEY <= ?* after mass delete

Implemented by Vlad Khorsun

[#8066](#) — Make protocol schemes case-insensitive

Implemented by Vlad Khorsun

[#8061](#) — Unnest IN/ANY/EXISTS sub-queries and optimize them using semi-join algorithm

See also the configuration parameter [SubQueryConversion](#).

Implemented by Dmitry Yemanov

[#8042](#) — Improve conflict resolution on replica when table has both primary and unique keys

Implemented by Vlad Khorsun

[#8030](#) — Better cardinality estimation when empty data pages exist

Implemented by Vlad Khorsun

[#8010](#) — Remove `gfix -cache` option

Implemented by Vlad Khorsun

[#7978](#) — Update Windows distributions with *zlib* version 1.3.1

Implemented by Vlad Khorsun

[#7928](#) — Make *TempCacheLimit* setting to be per-database (not per-attachment) for *SuperClassic*

Implemented by Vlad Khorsun

Bug Reporting

Bugs fixed in this release are listed and described in the chapter entitled [Bugs Fixed](#).

- 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 has not 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.
 2. Include reproducible test data in your report and post it to our [Tracker](#).
-

Documentation

You will find all the README documents referred to in these notes — as well as many others not referred to — in the doc subdirectory of your Firebird 5.0 installation.

— *The Firebird Project*

Chapter 2. New In Firebird 5.0

Summary of New Features

Firebird 5.0 introduces many improvements without any changes in architecture or operation, the most important are:

- Parallel (multi-threaded) operation for backup/restore, sweep and index creation;
- Partial indices;
- SKIP LOCKED clause for SELECT WITH LOCK, UPDATE and DELETE statements;
- Inline minor ODS upgrade;
- Compiled statement cache;
- PSQL and SQL profiler;
- Support for WHEN NOT MATCHED BY SOURCE for MERGE statement;
- Support multiple rows for DML RETURNING;
- New built-in functions and packages;
- Denser record-level compression;
- Network support for scrollable cursors;

The following list summarises the features and changes, with links to the chapters and topics where more detailed information can be found.

Complete In Firebird 5.0 Release Candidate 2

Allow to configure Firebird in POSIX using relative directories with options --with-fb

[#7918](#) by *Adriano dos Santos Fernandes*

Add backward compatibility option that disables joins transformation

[#7910](#) by *Dmitry Yemanov*

For more details, see [OuterJoinConversion](#).

Resolve performance issue with time zones

[#7854](#) by *Adriano dos Santos Fernandes, Vlad Khorsun*

Resolve difficulty returning the product version with the legacy connection

[#7819](#) by *Vlad Khorsun*

Extend RDB\$GET_CONTEXT('SYSTEM', '*') with other info from MON\$ATTACHMENTS

[#7818](#) by *Vlad Khorsun*

Don't update database-level statistics on every page cache operation

[#7814](#) by *Vlad Khorsun*

Improve SKIP LOCKED implementation

[#7810](#) by *Vlad Khorsun*

For more details, see [SKIP LOCKED clause](#).

Update Windows distribution with new *zlib* version 1.3 (released 2023-08-18)

[#7755](#) by *Vlad Khorsun*

Complete In Firebird 5.0 Release Candidate 1**Avoid truncation of the access path information inside the `PLG$PROF_RECORD_SOURCES` table to 255 characters**

[#7752](#) by *Adriano dos Santos Fernandes*

MacOS: build *libicu* and static *libc++* using *vcpkg*

[#7720](#) by *Adriano dos Santos Fernandes*

Better processing and optimization of `IN <list>` predicates

[#7707](#) by *Dmitry Yemanov*

For more details, see [Improvements to IN](#).

Make trace config parser resolve symlinks in database file path in trace configuration

[#7692](#) by *Vlad Khorsun*

Allow tracing database events even if the trace/audit is configured using a symbolic link instead of a real database file path.

Profiler should not miss query's top-level access paths nodes

[#7688](#) by *Adriano dos Santos Fernandes*

Add `LEVEL` column to `PLG$PROF_RECORD_SOURCES` and `PLG$PROF_RECORD_SOURCE_STATS_VIEW`

[#7687](#) by *Adriano dos Santos Fernandes*

Column `LEVEL` is added to the profiler record source tables/views. It returns the actual indentation level that allows proper reconstruction of multi-line access path nodes.

Add overload `FbVarChar::set` function for non null-terminated string

[#7685](#) by *Adriano dos Santos Fernandes*

Use *ParallelWorkers* setting from *firebird.conf* as default for all parallelized operations

[#7682](#) by *Vlad Khorsun*

This allows utilities started via the Service Manager to use the server-side *ParallelWorkers* setting without a need to specify the number of workers explicitly.

Make boot build on Windows a bit more user-friendly

[#7680](#) by *Vlad Khorsun*

Make the profiler store aggregated requests by default, with an option for the detailed store

[#7652](#) by *Adriano dos Santos Fernandes*

Make the profiler data aggregated at the statement level by default (with `REQUEST_ID = 0`), thus making the flushing faster. A detailed (non-aggregated) data may be requested explicitly by specifying the `'DETAILED_REQUESTS'` value in the `PLUGIN_OPTIONS` parameter of the `RDB$PROFILER.START_SESSION` function.

Getting the current DECFLOAT ROUND/TRAPS settings

[#7642](#) by *Alexander Peshkov*

New context variables `DECFLOAT_ROUND` and `DECFLOAT_TRAPS` are added to the `SYSTEM` namespace of the `RDB$GET_CONTEXT` function. They return the current settings of the corresponding session options.

Run as application not specifying switch -a

[#7637](#) by *Vlad Khorsun*

Command-line switch `-a` is no longer required on Windows to start the Firebird server as an application.

Include Performance Cores only in default affinity mask

[#7634](#) by *Vlad Khorsun*

Some modern CPUs may contain two sets of cores - Efficient (E) and Performance (P). By default Firebird runs on all available cores and that may cause performance degradation in CPU-bound tasks. Now, E-cores are excluded from the default affinity mask. If the *AffinityMask* configuration setting is used explicitly, please pay attention which types of cores are specified to be enabled.

Allow nested parenthesized joined table

[#7576](#) by *Mark Rotteveel*

Optimize creation of expression and partial indices

[#7559](#) by *Dmitry Yemanov*

Add support for -parallel in combination with gfix -icu

[#7550](#) by *Vlad Khorsun*

`gfix -icu` rebuilds indexes and thus can benefit from parallelization. Now it's allowed to use the `-icu` and `-parallel` switches together, to override the *ParallelWorkers* setting used by default.

Compiler warnings raise when build cloop-generated *Firebird.pas* in RAD Studio 11.3

[#7542](#) by *Vlad Khorsun*

RDB\$GET/SET_CONTEXT() — enclosing in apostrophes or double quotes of a missed namespace/variable will make output more readable

[#7539](#) by *Vlad Khorsun*

Add ability to query current value of parallel workers for an attachment

[#7536](#) by *Vlad Khorsun*

- New column `MON$PARALLEL_WORKERS` `INTEGER` was added into `MON$ATTACHMENTS` table.
- New variable `PARALLEL_WORKERS` is now available in the `SYSTEM` context of the `RDB$GET_CONTEXT` function.
- New tag `fb_info_parallel_workers` (value 149) is available in `IAttachment::getInfo()` and `isc_database_info()` API calls.

Reduce output of the `SHOW GRANTS` command

[#7506](#) by *Artyom Ivanov*

Firebird performance issue — unnecessary index reads

[#7494](#) by *Vlad Khorsun*

Index scan algorithm has been improved to avoid unnecessary record reads for partial lookups in compound indices.

`SHOW SYSTEM` command: provide list of functions belonging to system packages

[#7475](#) by *Alexander Peshkov*

Make Android port (client/embedded) work inside apps

[#7469](#) by *Adriano dos Santos Fernandes*

Add `COMPILE` trace events for procedures/functions/triggers

[#7466](#) by *Dmitry Yemanov*

Allows to trace parse/compile events for the stored modules, corresponding elapsed time and also plans for queries inside those `PSQL` modules.

Add `REPLICA MODE` to the output of the `isql SHOW DATABASE` command

[#7425](#) by *Dmitry Yemanov*

Surface internal optimization modes (all rows vs first rows) at the SQL and configuration levels

[#7405](#) by *Dmitry Yemanov*

For more details, see [OptimizeForFirstRows](#), [OPTIMIZE FOR](#) Clause, and [SET OPTIMIZE](#).

Use Windows private namespace for kernel objects used in server-to-server IPC

[#7213](#) by *Vlad Khorsun*

This improvement allows to synchronize Firebird processes across different Windows sessions.

Add ability to add comment to mapping (`COMMENT ON MAPPING ... IS ...`)

[#7046](#) by *Alexander Peshkov*

For more details, see [COMMENT ON MAPPING](#).

ISQL showing publication status

[#7001](#) by *Dmitry Yemanov*

Add support for QUARTER to EXTRACT, FIRST_DAY and LAST_DAY functions

[#5959](#) by *Adriano dos Santos Fernandes*

For more details, see [QUARTER added to EXTRACT, FIRST_DAY and LAST_DAY](#).

Complete In Firebird 5.0 Beta 1

Parallel (multi-threaded) operations

[#1783](#), [#3374](#), [#7447](#) by *Vlad Khorsun*

Such operations as logical backup/restore, sweeping and CREATE INDEX statement execution can be executed in parallel by multiple threads, thus decreasing the total operation time.

See chapters [Support for parallel operations](#), [Parallel backup/restore](#) and [Parallel sweep and ICU dependencies rebuild](#) for more details.

Support for partial indices

[#7257](#) by *Dmitry Yemanov*

The CREATE INDEX DDL statement has been extended to support [partial indices](#), i.e. an index may now declare a condition that defines the subset of records to be indexed.

SKIP LOCKED clause

[#7350](#) by *Adriano dos Santos Fernandes*

New clause [SKIP LOCKED](#) was introduced for statements SELECT WITH LOCK, UPDATE and DELETE. It allows to skip the already locked records while reading the table.

Inline minor ODS upgrade

[#7397](#) by *Dmitry Yemanov*

An ability to [upgrade the database](#) to the latest minor ODS version has been introduced, it does not require a backup/restore cycle.

Compiled statement cache

[#7144](#) by *Adriano dos Santos Fernandes*

A per-attachment [cache of compiled SQL statements](#) has been implemented.

PSQL and SQL profiler

[#7086](#) by *Adriano dos Santos Fernandes*

A built-in ability to [profile SQL and PSQL statements](#) has been added, making it possible to measure execution time at different levels.

Support for WHEN NOT MATCHED BY SOURCE in the MERGE statement

[#6681](#) by *Adriano dos Santos Fernandes*

The MERGE statement has been extended to support the [WHEN NOT MATCHED BY SOURCE](#) clause.

Built-in functions UNICODE_CHAR and UNICODE_VAL

[#6798](#) by Adriano dos Santos Fernandes

New [built-in functions UNICODE_CHAR and UNICODE_VAL](#) have been added to allow conversion between Unicode code point and character.

RDB\$BLOB_UTIL new system package

[#281](#) by Adriano dos Santos Fernandes

New [system package RDB\\$BLOB_UTIL](#) has been added to allow various operations with BLOBs in the PSQL modules.

Support multiple rows being returned by DML with the RETURNING clause

[#6815](#) by Adriano dos Santos Fernandes

The RETURNING clause, if used in DSQL queries, now [allows multiple rows to be returned](#).

Optimize the record-level RLE algorithm for a denser compression of shorter-than-declared strings and sets of subsequent NULLs

[#4723](#) by Dmitry Yemanov

The built-in [compression algorithm has been improved](#) to allow denser compression of records.

More cursor-related details in the plan output

[#7441](#) by Dmitry Yemanov

Execution plan now contains [more information about cursors](#).

Other improvements are briefly listed below, please follow the tracker references for more information.

Unify display of system procedures & packages with other system objects

[#7411](#) by Alexander Peshkov

System procedures and packages are now shown by [SHOW SYSTEM](#).

Simplify client library build

[#7399](#) by Adriano dos Santos Fernandes

Performance improvement for BLOB copying

[#7382](#) by Adriano dos Santos Fernandes

Cost-based choice between nested loop join and hash join

[#7331](#) by Dmitry Yemanov

Create Android packages with all necessary files in all architectures (x86, x64, arm32, arm64)

[#7293](#) by Adriano dos Santos Fernandes

Unify release filenames

[#7284](#) by Adriano dos Santos Fernandes

Improve ICU version mismatch diagnostics

[#7169](#) by Adriano dos Santos Fernandes

Provide ability to see in the trace log events related to missing security context

[#7165](#) by Alexander Peshkov

For more details, see [Trace events before a valid security context is established](#).

ResultSet.getInfo() new API method

[#7083](#) by Dmitry Yemanov

Network support for scrollable cursors

[#7051](#) by Dmitry Yemanov

Add table MON\$COMPILED_STATEMENTS and also column MON\$COMPILED_STATEMENT_ID to both MON\$STATEMENTS and MON\$CALL_STACK tables

[#7050](#) by Adriano dos Santos Fernandes

Results of negation must be the same for each datatype (SMALLINT / INT / BIGINT / INT128) when argument is minimum value for this type

[#7025](#) by Alexander Peshkov

Transform OUTER joins into INNER ones if the WHERE condition violates the outer join rules

[#6992](#) by Dmitry Yemanov

Add way to retrieve statement BLR with Statement.getInfo() and ISQL's SET EXEC_PATH_DISPLAY BLR

[#6910](#) by Adriano dos Santos Fernandes

For *isql*, see [Display statement BLR](#).

SIMILAR TO should use index when pattern starts with non-wildcard character (as LIKE does)

[#6873](#) by Adriano dos Santos Fernandes

Add column MON\$SESSION_TIMEZONE to the table MON\$ATTACHMENTS

[#6794](#) by Adriano dos Santos Fernandes

Allow parenthesized query expression for standard-compliance

[#6740](#) by Adriano dos Santos Fernandes

For more details, see [Allow parenthesized query expressions](#).

System table with keywords

[#6713](#) by Adriano dos Santos Fernandes

Support full SQL standard character string literal syntax

[#5589](#) by *Adriano dos Santos Fernandes*

For more details, see [Full SQL standard character string literal syntax](#).

Support full SQL standard binary string literal syntax

[#5588](#) by *Adriano dos Santos Fernandes*

For more details, see [Full SQL standard binary string literal syntax](#).

Allow subroutines to access variables/parameters defined at the outer/parent level

[#4769](#) by *Adriano dos Santos Fernandes*

For more details, see [Allow subroutines to access variables/parameters defined at the outer/parent level](#).

Avoid data retrieval if the WHERE clause always evaluates to FALSE

[#1708](#) by *Dmitry Yemanov*

Chapter 3. Changes in the Firebird Engine

Quick Links

- [Support for parallel operations](#)
- [Inline minor ODS upgrade](#)
- [More cursor-related details in the plan output](#)
- [Compiled statement cache](#)
- [Denser compression of records](#)
- [SQL and PSQL profiler](#)
- [RDB\\$BLOB_UTIL package](#)

Support for parallel operations

Vlad Khorsun

Tracker ticket: [#7447](#)

The Firebird engine can now execute some tasks using multiple threads in parallel. Currently, parallel execution is implemented for the sweep and the index creation tasks. Parallel execution is supported for both automatic and manual sweep.

To handle a task with multiple threads, the engine runs additional worker threads and creates internal worker attachments. By default, parallel execution is not enabled. There are two ways to enable parallelism in a user attachment:

1. set the number of parallel workers in DPB using new tag *isc_dpb_parallel_workers*,
2. set the default number of parallel workers using new setting *ParallelWorkers* in *firebird.conf*.

The *gfix* utility has a new command-line switch, *-parallel*, that allows to set the number of parallel workers for the sweep task.

For example, the following will run sweep on the given database and asks the engine to use 4 workers:

```
gfix -sweep -parallel 4 <database>
```

gfix uses DPB tag *isc_dpb_parallel_workers* when attaches to <database>, if switch *-parallel* is present.

A similar option was also added to *gbak*.

The new *firebird.conf* setting *ParallelWorkers* sets the default number of parallel workers that can be used by any user attachment running parallelizable task. The default value is 1 and means no use of additional parallel workers. The value in the DPB has a higher priority than the setting in

firebird.conf.

To control the number of additional workers that can be created by the engine, there are two new settings in firebird.conf:

ParallelWorkers

Sets the default number of parallel workers used by a user attachments. Can be overridden by attachment using tag *isc_dpb_parallel_workers* in DPB.

MaxParallelWorkers

Limits the maximum number of simultaneously used workers for the given database and Firebird process.

Internal worker attachments are created and managed by the engine itself. The engine maintains per-database pools of worker attachments. The number of threads in each pool is limited by the value of the *MaxParallelWorkers* setting. The pools are created by each Firebird process independently.

In SuperServer architecture worker attachments are implemented as light-weight system attachments, while in Classic and SuperClassic they look like usual user attachments. All worker attachments are embedded into the creating server process. Thus, in Classic architectures there are no additional server processes. Worker attachments are present in monitoring tables. Idle worker attachments are destroyed after 60 seconds of inactivity. Also, in Classic architectures, worker attachments are destroyed immediately after the last user connection detaches from the database.

Examples:

Set in firebird.conf *ParallelWorkers = 4*, *MaxParallelWorkers = 8* and restart Firebird server.

1. Connect to test database not using *isc_dpb_parallel_workers* in DPB and execute a `CREATE INDEX ... SQL` statement. On commit, the index will be created and the engine will use three additional worker attachments. In total, four attachments in four threads will work on index creation.
2. Ensure auto-sweep is enabled for test database. When auto-sweep runs on that database, it will also use three additional workers (and run within four threads).
3. More than one task at a time can be parallelized: make two attachments and execute a `CREATE INDEX ...` in each of them (of course indices to be built should be different). Each index will be created using four attachments (one user and three worker) and four threads.
4. Run `gfix -sweep <database>` without specifying switch `-parallel`: sweep will run using four attachments in four threads.
5. Run `gfix -sweep -parallel 2 <database>`: sweep will run using two attachments in two threads. This shows that value in DPB tag *isc_dpb_parallel_workers* overrides value of setting *ParallelWorkers*.

Inline minor ODS upgrade

Dmitry Yemanov

Tracker ticket: [#7397](#)

This feature allows to upgrade the existing database to the newest ODS version without backup/restore, provided that the database belongs to the same major ODS version.

For example, a database created by Firebird 4.0 uses ODS 13.0 and thus can be upgraded to the ODS 13.1 used by Firebird 5.0.

Notes

- The upgrade must be done manually, using *gfix -upgrade* command
- It requires exclusive access to the database, an error is thrown otherwise
- The system privilege `USE_GFIX_UTILITY` is required
- An upgrade is transactional, all changes are reverted if any error happens
- After the upgrade, Firebird 4.0 can no longer open the database

Usage

```
gfix -upgrade <database>
```

See also [ODS upgrade by gfix](#).



This is a one-way modification, downgrading backward is impossible. So please make a database copy before upgrading, just to have a recovery point if something goes wrong during the process.

More cursor-related details in the plan output

Dmitry Yemanov

Tracker ticket: [#7441](#)

Detailed plan output now distinguishes between user-specified SELECT statements (reported as *select expressions*), PSQL declared cursors and sub-queries. Both legacy and detailed plans now also include information about cursor's position (line/column) inside their PSQL module.

Examples:

Legacy plan

```
-- line 23, column 2
PLAN (DISTRICT INDEX (DISTRICT_PK))
-- line 28, column 2
PLAN JOIN (CUSTOMER INDEX (CUSTOMER_PK), WAREHOUSE INDEX(WAREHOUSE_PK))
```

Detailed plan

```
Select Expression (line 23, column 2)
-> Singularity Check
-> Filter
```

```

-> Table "DISTRICT" Access By ID
    -> Bitmap
        -> Index "DISTRICT_PK" Unique Scan
Select Expression (line 28, column 2)
    -> Singularity Check
        -> Nested Loop Join (inner)
            -> Filter
                -> Table "CUSTOMER" Access By ID
                    -> Bitmap
                        -> Index "CUSTOMER_PK" Unique Scan
            -> Filter
                -> Table "WAREHOUSE" Access By ID
                    -> Bitmap
                        -> Index "WAREHOUSE_PK" Unique Scan

```

Line/column numbers (as well as PSQL declared cursors) cannot be seen directly in the plan for user-specified SQL queries, except if the query is EXECUTE BLOCK. However, they are accessible in the MON\$EXPLAINED_PLAN column in either MON\$STATEMENTS or MON\$COMPILED_STATEMENTS tables.

Denser compression of records

Dmitry Yemanov

Tracker ticket: [#4723](#)

Starting with ODS 13.1, the engine uses an advanced RLE compression method (with variable-length counter) that stores repeating byte sequences more effectively, thus reducing the storage overhead. This improves compression for long VARCHAR fields (especially UTF8 encoded) that are filled only partially.

Compiled statement cache

Adriano dos Santos Fernandes

Tracker ticket: [#7144](#)

The engine now maintains a per-attachment cache of compiled SQL statements. By default, caching is enabled, the caching threshold is defined by the *MaxStatementCacheSize* parameter in firebird.conf. It can be disabled by setting *MaxStatementCacheSize* to zero.

The cache is maintained automatically; cached statements are invalidated when required (usually when some DDL statement is executed).

SQL and PSQL profiler

Adriano dos Santos Fernandes

Tracker ticket: [#7086](#)

The profiler allows users to measure performance cost of SQL and PSQL code. It's implemented with a system package in the engine passing data to a profiler plugin.

This documentation treats the engine and plugin parts as a single thing, in the way the default profiler (Default_Profiler) is going to be used.

The RDB\$PROFILER package can profile execution of PSQL code, collecting statistics of how many times each line was executed along with its minimum, maximum and accumulated execution times (with nanoseconds precision), as well as open and fetch statistics of implicit and explicit SQL cursors.

To collect profile data, a user must first start a profile session with RDB\$PROFILER.START_SESSION. This function returns a profile session ID which is later stored in the profiler snapshot tables to be queried and analyzed by the user. A profiler session may be local (same attachment) or remote (another attachment).

Remote profiling just forwards commands to the remote attachment. So, it's possible that a client profiles multiple attachments simultaneously. It's also possible that a locally or remotely started profile session have commands issued by another attachment.

Remotely issued commands require that the target attachment is in an idle state, i.e. not executing others requests. When the target attachment is not idle, the call blocks waiting for that state.

If the remote attachment is from a different user, the calling user must have the system privilege PROFILE_ANY_ATTACHMENT.

After a session is started, PSQL and SQL statements statistics are collected in memory. A profile session collects data only of statements executed in the same attachment associated with the session. Data is aggregated and stored per requests (i.e. a statement execution). When querying snapshot tables, the user may do extra aggregation per statement, or use the auxiliary views that do that automatically.

A session may be paused to temporarily disable statistics collecting. It may be resumed later to return statistics collection in the same session.

A new session may be started when a session is already active. In that case, it has the same semantics of finishing the current session with RDB\$PROFILER.FINISH_SESSION(FALSE), so snapshots tables are not updated.

To analyze the collected data, the user must flush the data to the snapshot tables, which can be done by finishing or pausing a session (with FLUSH parameter set to TRUE), or calling RDB\$PROFILER.FLUSH. Data is flushed using an autonomous transaction (a transaction started and finished for the specific purpose of profiler data update).

Below is a sample profile session and queries for data analysis.

1. Preparation — create table and routines that will be analyzed

```
create table tab (  
    id integer not null,
```

```

        val integer not null
    );

set term !;

create or alter function mult(p1 integer, p2 integer) returns integer
as
begin
    return p1 * p2;
end!

create or alter procedure ins
as
    declare n integer = 1;
begin
    while (n <= 1000)
    do
        begin
            if (mod(n, 2) = 1) then
                insert into tab values (:n, mult(:n, 2));
            n = n + 1;
        end
    end!

set term ;!

```

2. Start profiling

```

select rdb$profiler.start_session('Profile Session 1') from rdb$database;

set term !;

execute block
as
begin
    execute procedure ins;
    delete from tab;
end!

set term ;!

execute procedure rdb$profiler.finish_session(true);

execute procedure ins;

select rdb$profiler.start_session('Profile Session 2') from rdb$database;

select mod(id, 5),
       sum(val)
from tab

```

```

where id <= 50
group by mod(id, 5)
order by sum(val);

execute procedure rdb$profiler.finish_session(true);

```

3. Data analysis

```

set transaction read committed;

select * from plg$prof_sessions;

select * from plg$prof_psql_stats_view;

select * from plg$prof_record_source_stats_view;

select preq.*
  from plg$prof_requests preq
 join plg$prof_sessions pses
    on pses.profile_id = preq.profile_id and
       pses.description = 'Profile Session 1';

select pstat.*
  from plg$prof_psql_stats pstat
 join plg$prof_sessions pses
    on pses.profile_id = pstat.profile_id and
       pses.description = 'Profile Session 1'
 order by pstat.profile_id,
          pstat.request_id,
          pstat.line_num,
          pstat.column_num;

select pstat.*
  from plg$prof_record_source_stats pstat
 join plg$prof_sessions pses
    on pses.profile_id = pstat.profile_id and
       pses.description = 'Profile Session 2'
 order by pstat.profile_id,
          pstat.request_id,
          pstat.cursor_id,
          pstat.record_source_id;

```

Package routines

Function START_SESSION

RDB\$PROFILER.START_SESSION starts a new profiler session, makes it the current session (of the given ATTACHMENT_ID) and returns its identifier.

If FLUSH_INTERVAL is different from NULL, auto-flush is set up in the same way as manually calling RDB\$PROFILER.SET_FLUSH_INTERVAL.

If PLUGIN_NAME is NULL (the default), it uses the database configuration DefaultProfilerPlugin.

PLUGIN_OPTIONS are plugin specific options and currently should be NULL for the Default_Profiler plugin.

Input parameters

- DESCRIPTION type VARCHAR(255) CHARACTER SET UTF8 default NULL
- FLUSH_INTERVAL type INTEGER default NULL
- ATTACHMENT_ID type BIGINT NOT NULL default CURRENT_CONNECTION
- PLUGIN_NAME type VARCHAR(255) CHARACTER SET UTF8 default NULL
- PLUGIN_OPTIONS type VARCHAR(255) CHARACTER SET UTF8 default NULL

Return type

BIGINT NOT NULL

Procedure PAUSE_SESSION

RDB\$PROFILER.PAUSE_SESSION pauses the current profiler session (of the given ATTACHMENT_ID), so the next executed statements statistics are not collected.

If FLUSH is TRUE, the snapshot tables are updated with data up to the current moment, otherwise data remains only in memory for later update.

Calling RDB\$PROFILER.PAUSE_SESSION(TRUE) has the same semantics as calling RDB\$PROFILER.PAUSE_SESSION(FALSE) followed by RDB\$PROFILER.FLUSH (using the same ATTACHMENT_ID).

Input parameters

- FLUSH type BOOLEAN NOT NULL default FALSE
- ATTACHMENT_ID type BIGINT NOT NULL default CURRENT_CONNECTION

Procedure RESUME_SESSION

RDB\$PROFILER.RESUME_SESSION resumes the current profiler session (of the given ATTACHMENT_ID), if it was paused, so the next executed statements statistics are collected again.

Input parameter

- ATTACHMENT_ID type BIGINT NOT NULL default CURRENT_CONNECTION

Procedure FINISH_SESSION

RDB\$PROFILER.FINISH_SESSION finishes the current profiler session (of the given ATTACHMENT_ID).

If FLUSH is TRUE, the snapshot tables are updated with data of the finished session (and old finished sessions not yet present in the snapshot), otherwise data remains only in memory for later update.

Calling RDB\$PROFILER.FINISH_SESSION(TRUE) has the same semantics of calling

RDB\$PROFILER.FINISH_SESSION(FALSE) followed by RDB\$PROFILER.FLUSH (using the same ATTACHMENT_ID).

Input parameters

- FLUSH type BOOLEAN NOT NULL default TRUE
- ATTACHMENT_ID type BIGINT NOT NULL default CURRENT_CONNECTION

Procedure CANCEL_SESSION

RDB\$PROFILER.CANCEL_SESSION cancels the current profiler session (of the given ATTACHMENT_ID).

All session data present in the profiler plugin is discarded and will not be flushed.

Data already flushed is not deleted automatically.

Input parameter

- ATTACHMENT_ID type BIGINT NOT NULL default CURRENT_CONNECTION

Procedure DISCARD

RDB\$PROFILER.DISCARD removes all sessions (of the given ATTACHMENT_ID) from memory, without flushing them.

If there is an active session, it is cancelled.

Input parameter

- ATTACHMENT_ID type BIGINT NOT NULL default CURRENT_CONNECTION

Procedure FLUSH

RDB\$PROFILER.FLUSH updates the snapshot tables with data from the profile sessions (of the given ATTACHMENT_ID) in memory.

After flushing, the data is stored in tables PLG\$PROF_SESSIONS, PLG\$PROF_STATEMENTS, PLG\$PROF_RECORD_SOURCES, PLG\$PROF_REQUESTS, PLG\$PROF_PSQL_STATS and PLG\$PROF_RECORD_SOURCE_STATS and may be read and analyzed by the user.

Data is updated using an autonomous transaction, so if the procedure is called in a snapshot transaction, data will not be directly readable in the same transaction.

Once flush happens, finished sessions are removed from memory.

Input parameter

- ATTACHMENT_ID type BIGINT NOT NULL default CURRENT_CONNECTION

Procedure SET_FLUSH_INTERVAL

RDB\$PROFILER.SET_FLUSH_INTERVAL turns periodic auto-flush on (when FLUSH_INTERVAL is greater than 0) or off (when FLUSH_INTERVAL is equal to 0).

FLUSH_INTERVAL is interpreted as number of seconds.

Input parameters

- FLUSH_INTERVAL type INTEGER NOT NULL
- ATTACHMENT_ID type BIGINT NOT NULL default CURRENT_CONNECTION

Snapshot tables

Snapshot tables (as well views and sequence) are automatically created in the first usage of the profiler. They are owned by the database owner, with read/write permissions for PUBLIC.

When a session is deleted, the related data in other profiler snapshot tables are automatically deleted too through foreign keys with DELETE CASCADE option.

Below is the list of tables that stores profile data.

Table PLG\$PROF_SESSIONS

PROFILE_ID type BIGINT	Profile session ID
ATTACHMENT_ID type BIGINT	Attachment ID
USER_NAME type CHAR(63) CHARACTER SET UTF8	Username
DESCRIPTION type VARCHAR(255) CHARACTER SET UTF8	Description passed in RDB\$PROFILER.START_SESSION
START_TIMESTAMP type TIMESTAMP WITH TIME ZONE	Moment the profile session was started
FINISH_TIMESTAMP type TIMESTAMP WITH TIME ZONE	Moment the profile session was finished (NULL when not finished)
Primary key	PROFILE_ID

Table PLG\$PROF_STATEMENTS

PROFILE_ID type BIGINT	Profile session ID
STATEMENT_ID type BIGINT	Statement ID
PARENT_STATEMENT_ID type BIGINT	Parent statement ID — related to sub routines
STATEMENT_TYPE type VARCHAR(20) CHARACTER SET UTF8	BLOCK, FUNCTION, PROCEDURE or TRIGGER
PACKAGE_NAME type CHAR(63) CHARACTER SET UTF8	Package of FUNCTION or PROCEDURE
ROUTINE_NAME type CHAR(63) CHARACTER SET UTF8	Routine name of FUNCTION, PROCEDURE or TRIGGER
SQL_TEXT type BLOB SUB_TYPE TEXT CHARACTER SET UTF8	SQL text for BLOCK
Primary key	PROFILE_ID, STATEMENT_ID

Table PLG\$PROF_CURSORS

PROFILE_ID type BIGINT	Profile session ID
STATEMENT_ID type BIGINT	Statement ID
CURSOR_ID type INTEGER	Cursor ID
NAME type CHAR(63) CHARACTER SET UTF8	Name of explicit cursor
LINE_NUM type INTEGER	Line number of the cursor
COLUMN_NUM type INTEGER	Column number of the cursor
Primary key	PROFILE_ID, STATEMENT_ID, CURSOR_ID

Table PLG\$PROF_RECORD_SOURCES

PROFILE_ID type BIGINT	Profile session ID
STATEMENT_ID type BIGINT	Statement ID
CURSOR_ID type INTEGER	Cursor ID
RECORD_SOURCE_ID type INTEGER	Record source ID
PARENT_RECORD_SOURCE_ID type INTEGER	Parent record source ID
LEVEL	Indentation level for the record source
ACCESS_PATH type BLOB SUB_TYPE TEXT CHARACTER SET UTF8	Access path for the record source
Primary key	PROFILE_ID, STATEMENT_ID, CURSOR_ID, RECORD_SOURCE_ID

Table PLG\$PROF_REQUESTS

PROFILE_ID type BIGINT	Profile session ID
STATEMENT_ID type BIGINT	Statement ID
REQUEST_ID type BIGINT	Request ID
CALLER_STATEMENT_ID type BIGINT	Caller statement ID
CALLER_REQUEST_ID type BIGINT	Caller request ID
START_TIMESTAMP type TIMESTAMP WITH TIME ZONE	Moment this request was first gathered profile data
FINISH_TIMESTAMP type TIMESTAMP WITH TIME ZONE	Moment this request was finished

TOTAL_ELAPSED_TIME type BIGINT	Accumulated elapsed time (in nanoseconds) of the request
Primary key	PROFILE_ID, REQUEST_ID

Table PLG\$PROF_PSQL_STATS

PROFILE_ID type BIGINT	Profile session ID
STATEMENT_ID type BIGINT	Statement ID
REQUEST_ID type BIGINT	Request ID
LINE_NUM type INTEGER	Line number of the statement
COLUMN_NUM type INTEGER	Column number of the statement
COUNTER type BIGINT	Number of executed times of the line/column
MIN_ELAPSED_TIME type BIGINT	Minimal elapsed time (in nanoseconds) of a line/column execution
MAX_ELAPSED_TIME type BIGINT	Maximum elapsed time (in nanoseconds) of a line/column execution
TOTAL_ELAPSED_TIME type BIGINT	Accumulated elapsed time (in nanoseconds) of the line/column executions
Primary key	PROFILE_ID, REQUEST_ID, LINE_NUM, COLUMN_NUM

Table PLG\$PROF_RECORD_SOURCE_STATS

PROFILE_ID type BIGINT	Profile session ID
STATEMENT_ID type BIGINT	Statement ID
REQUEST_ID type BIGINT	Request ID
CURSOR_ID type INTEGER	Cursor ID
RECORD_SOURCE_ID type `INTEGER	Record source ID
OPEN_COUNTER type BIGINT	Number of open times of the record source
OPEN_MIN_ELAPSED_TIME type BIGINT	Minimal elapsed time (in nanoseconds) of a record source open
OPEN_MAX_ELAPSED_TIME type BIGINT	Maximum elapsed time (in nanoseconds) of a record source open
OPEN_TOTAL_ELAPSED_TIME type BIGINT	Accumulated elapsed time (in nanoseconds) of the record source openings

<code>FETCH_COUNTER</code> type <code>BIGINT</code>	Number of fetch times of the record source
<code>FETCH_MIN_ELAPSED_TIME</code> type <code>BIGINT</code>	Minimal elapsed time (in nanoseconds) of a record source fetch
<code>FETCH_MAX_ELAPSED_TIME</code> type <code>BIGINT</code>	Maximum elapsed time (in nanoseconds) of a record source fetch
<code>FETCH_TOTAL_ELAPSED_TIME</code> type <code>BIGINT</code>	Accumulated elapsed time (in nanoseconds) of the record source fetches
Primary key	<code>PROFILE_ID</code> , <code>REQUEST_ID</code> , <code>CURSOR_ID</code> , <code>RECORD_SOURCE_ID</code>

Auxiliary views

These views help profile data extraction aggregated at statement level.

They should be the preferred way to analyze the collected data. They can also be used together with the tables to get additional data not present on the views.

After hotspots are found, one can drill down in the data at the request level through the tables.

View `PLG$PROF_STATEMENT_STATS_VIEW`

```
select req.profile_id,
       req.statement_id,
       sta.statement_type,
       sta.package_name,
       sta.routine_name,
       sta.parent_statement_id,
       sta_parent.statement_type parent_statement_type,
       sta_parent.routine_name parent_routine_name,
       (select sql_text
        from plg$prof_statements
        where profile_id = req.profile_id and
              statement_id = coalesce(sta.parent_statement_id, req.statement_id)
       ) sql_text,
       count(*) counter,
       min(req.total_elapsed_time) min_elapsed_time,
       max(req.total_elapsed_time) max_elapsed_time,
       cast(sum(req.total_elapsed_time) as bigint) total_elapsed_time,
       cast(sum(req.total_elapsed_time) / count(*) as bigint) avg_elapsed_time
from plg$prof_requests req
join plg$prof_statements sta
  on sta.profile_id = req.profile_id and
     sta.statement_id = req.statement_id
left join plg$prof_statements sta_parent
  on sta_parent.profile_id = sta.profile_id and
     sta_parent.statement_id = sta.parent_statement_id
group by req.profile_id,
         req.statement_id,
```

```

        sta.statement_type,
        sta.package_name,
        sta.routine_name,
        sta.parent_statement_id,
        sta_parent.statement_type,
        sta_parent.routine_name
order by sum(req.total_elapsed_time) desc

```

View PLG\$PROF_PSQL_STATS_VIEW

```

select pstat.profile_id,
       pstat.statement_id,
       sta.statement_type,
       sta.package_name,
       sta.routine_name,
       sta.parent_statement_id,
       sta_parent.statement_type parent_statement_type,
       sta_parent.routine_name parent_routine_name,
       (select sql_text
        from plg$prof_statements
        where profile_id = pstat.profile_id and
              statement_id = coalesce(sta.parent_statement_id, pstat.statement_id)
       ) sql_text,
       pstat.line_num,
       pstat.column_num,
       cast(sum(pstat.counter) as bigint) counter,
       min(pstat.min_elapsed_time) min_elapsed_time,
       max(pstat.max_elapsed_time) max_elapsed_time,
       cast(sum(pstat.total_elapsed_time) as bigint) total_elapsed_time,
       cast(sum(pstat.total_elapsed_time) / nullif(sum(pstat.counter), 0) as bigint)
avg_elapsed_time
from plg$prof_psql_stats pstat
join plg$prof_statements sta
  on sta.profile_id = pstat.profile_id and
     sta.statement_id = pstat.statement_id
left join plg$prof_statements sta_parent
  on sta_parent.profile_id = sta.profile_id and
     sta_parent.statement_id = sta.parent_statement_id
group by pstat.profile_id,
         pstat.statement_id,
         sta.statement_type,
         sta.package_name,
         sta.routine_name,
         sta.parent_statement_id,
         sta_parent.statement_type,
         sta_parent.routine_name,
         pstat.line_num,
         pstat.column_num
order by sum(pstat.total_elapsed_time) desc

```

View PLG\$PROF_RECORD_SOURCE_STATS_VIEW

```

select rstat.profile_id,
       rstat.statement_id,
       sta.statement_type,
       sta.package_name,
       sta.routine_name,
       sta.parent_statement_id,
       sta_parent.statement_type parent_statement_type,
       sta_parent.routine_name parent_routine_name,
       (select sql_text
        from plg$prof_statements
        where profile_id = rstat.profile_id and
              statement_id = coalesce(sta.parent_statement_id, rstat.statement_id)
       ) sql_text,
       rstat.cursor_id,
       cur.name cursor_name,
       cur.line_num cursor_line_num,
       cur.column_num cursor_column_num,
       rstat.record_source_id,
       recsrc.parent_record_source_id,
       recsrc.level,
       recsrc.access_path,
       cast(sum(rstat.open_counter) as bigint) open_counter,
       min(rstat.open_min_elapsed_time) open_min_elapsed_time,
       max(rstat.open_max_elapsed_time) open_max_elapsed_time,
       cast(sum(rstat.open_total_elapsed_time) as bigint) open_total_elapsed_time,
       cast(sum(rstat.open_total_elapsed_time) / nullif(sum(rstat.open_counter), 0) as
bigint) open_avg_elapsed_time,
       cast(sum(rstat.fetch_counter) as bigint) fetch_counter,
       min(rstat.fetch_min_elapsed_time) fetch_min_elapsed_time,
       max(rstat.fetch_max_elapsed_time) fetch_max_elapsed_time,
       cast(sum(rstat.fetch_total_elapsed_time) as bigint) fetch_total_elapsed_time,
       cast(sum(rstat.fetch_total_elapsed_time) / nullif(sum(rstat.fetch_counter), 0)
as bigint) fetch_avg_elapsed_time,
       cast(coalesce(sum(rstat.open_total_elapsed_time), 0) +
coalesce(sum(rstat.fetch_total_elapsed_time), 0) as bigint)
open_fetch_total_elapsed_time
from plg$prof_record_source_stats rstat
join plg$prof_cursors cur
  on cur.profile_id = rstat.profile_id and
     cur.statement_id = rstat.statement_id and
     cur.cursor_id = rstat.cursor_id
join plg$prof_record_sources recsrc
  on recsrc.profile_id = rstat.profile_id and
     recsrc.statement_id = rstat.statement_id and
     recsrc.cursor_id = rstat.cursor_id and
     recsrc.record_source_id = rstat.record_source_id
join plg$prof_statements sta
  on sta.profile_id = rstat.profile_id and
     sta.statement_id = rstat.statement_id

```

```

left join plg$prof_statements sta_parent
  on sta_parent.profile_id = sta.profile_id and
     sta_parent.statement_id = sta.parent_statement_id
group by rstat.profile_id,
         rstat.statement_id,
         sta.statement_type,
         sta.package_name,
         sta.routine_name,
         sta.parent_statement_id,
         sta_parent.statement_type,
         sta_parent.routine_name,
         rstat.cursor_id,
         cur.name,
         cur.line_num,
         cur.column_num,
         rstat.record_source_id,
         recsrc.parent_record_source_id,
         recsrc.level,
         recsrc.access_path
order by coalesce(sum(rstat.open_total_elapsed_time), 0) +
         coalesce(sum(rstat.fetch_total_elapsed_time), 0) desc

```

RDB\$BLOB_UTIL package

Adriano dos Santos Fernandes

Tracker ticket: [#281](#)

This package provides procedures and functions to manipulate BLOBs in a way that standard Firebird functions, like BLOB_APPEND and SUBSTRING, cannot do or are very slow.

These routines operate on binary data directly, even for text BLOBs.

Package routines

Function NEW_BLOB

RDB\$BLOB_UTIL.NEW_BLOB creates a new BLOB SUB_TYPE BINARY. It returns a BLOB suitable for data appending, similar to BLOB_APPEND.

The advantage over BLOB_APPEND is that it's possible to set custom SEGMENTED and TEMP_STORAGE options.

BLOB_APPEND always creates BLOBs in temporary storage, which may not always be the best approach if the created BLOB is going to be stored in a permanent table, as this will require a copy operation.

The BLOB returned from this function, even when TEMP_STORAGE = FALSE, may be used with BLOB_APPEND for appending data.

Input parameters

- SEGMENTED type BOOLEAN NOT NULL
- TEMP_STORAGE type BOOLEAN NOT NULL

Return type

BLOB SUB_TYPE BINARY NOT NULL

Function OPEN_BLOB

RDB\$BLOB_UTIL.OPEN_BLOB opens an existing BLOB for reading. It returns a handle (an integer bound to the transaction) suitable for use with other functions of this package, like SEEK, READ_DATA and CLOSE_HANDLE.

Handles which are not explicitly closed are closed automatically when the transaction ends.

Input parameter

- BLOB type BLOB NOT NULL

Return type

INTEGER NOT NULL

Function IS_WRITABLE

RDB\$BLOB_UTIL.IS_WRITABLE returns TRUE when a BLOB is suitable for data appending using BLOB_APPEND without copying.

Input parameter

- BLOB type BLOB NOT NULL

Return type:

BOOLEAN NOT NULL

Function READ_DATA

RDB\$BLOB_UTIL.READ_DATA reads chunks of data of a BLOB handle opened with RDB\$BLOB_UTIL.OPEN_BLOB. When the BLOB is fully read and there is no more data, it returns NULL.

If LENGTH is passed with a positive number, it returns a VARBINARY with its maximum length.

If LENGTH is NULL it returns just a segment of the BLOB with a maximum length of 32765.

Input parameters

- HANDLE type INTEGER NOT NULL
- LENGTH type INTEGER

Return type

VARBINARY(32765)

Function SEEK

RDB\$BLOB_UTIL.SEEK sets the position for the next READ_DATA, it returns the new position.

MODE may be 0 (from the start), 1 (from current position) or 2 (from end).

When MODE is 2, OFFSET should be zero or negative.

Input parameters

- HANDLE type INTEGER NOT NULL
- MODE type INTEGER NOT NULL
- OFFSET type INTEGER NOT NULL

Return type

INTEGER NOT NULL



SEEK only works on stream blobs. Attempting to seek on a segmented blob results in error “invalid BLOB type for operation”.

Procedure CANCEL_BLOB

RDB\$BLOB_UTIL.CANCEL_BLOB immediately releases a temporary BLOB, like one created with BLOB_APPEND.

If the same BLOB is used after cancel, an “invalid blob id” error will be raised.

Input parameter

- BLOB type BLOB

Procedure CLOSE_HANDLE

RDB\$BLOB_UTIL.CLOSE_HANDLE closes a BLOB handle opened with RDB\$BLOB_UTIL.OPEN_BLOB.

Handles which are not explicitly closed are closed automatically when the transaction ends.

Input parameter

- HANDLE type INTEGER NOT NULL

Examples

Create a BLOB in temporary space and return it in EXECUTE BLOCK

```

execute block returns (b blob)
as
begin
    -- Create a BLOB handle in the temporary space.
    b = rdb$blob_util.new_blob(false, true);

    -- Add chunks of data.
    b = blob_append(b, '12345');

```

```

    b = blob_append(b, '67');

    suspend;
end

```

Open a BLOB and return chunks of it with EXECUTE BLOCK

```

execute block returns (s varchar(10))
as
    declare b blob = '1234567';
    declare bhandle integer;
begin
    -- Open the BLOB and get a BLOB handle.
    bhandle = rdb$blob_util.open_blob(b);

    -- Get chunks of data as string and return.

    s = rdb$blob_util.read_data(bhandle, 3);
    suspend;

    s = rdb$blob_util.read_data(bhandle, 3);
    suspend;

    s = rdb$blob_util.read_data(bhandle, 3);
    suspend;

    -- Here EOF is found, so it returns NULL.
    s = rdb$blob_util.read_data(bhandle, 3);
    suspend;

    -- Close the BLOB handle.
    execute procedure rdb$blob_util.close_handle(bhandle);
end

```

Seek in a blob

```

set term !;

execute block returns (s varchar(10))
as
    declare b blob;
    declare bhandle integer;
begin
    -- Create a stream BLOB handle.
    b = rdb$blob_util.new_blob(false, true);

    -- Add data.
    b = blob_append(b, '0123456789');

    -- Open the BLOB.

```

```

bhandle = rdb$blob_util.open_blob(b);

-- Seek to 5 since the start.
rdb$blob_util.seek(bhandle, 0, 5);
s = rdb$blob_util.read_data(bhandle, 3);
suspend;

-- Seek to 2 since the start.
rdb$blob_util.seek(bhandle, 0, 2);
s = rdb$blob_util.read_data(bhandle, 3);
suspend;

-- Advance 2.
rdb$blob_util.seek(bhandle, 1, 2);
s = rdb$blob_util.read_data(bhandle, 3);
suspend;

-- Seek to -1 since the end.
rdb$blob_util.seek(bhandle, 2, -1);
s = rdb$blob_util.read_data(bhandle, 3);
suspend;
end!

set term ;!

```

Check if blobs are writable

```

create table t(b blob);

set term !;

execute block returns (bool boolean)
as
    declare b blob;
begin
    b = blob_append(null, 'writable');
    bool = rdb$blob_util.is_writable(b);
    suspend;

    insert into t (b) values ('not writable') returning b into b;
    bool = rdb$blob_util.is_writable(b);
    suspend;
end!

set term ;!

```


Chapter 4. Changes to the Firebird API and ODS

since Firebird 4.0 release

ODS (On-Disk Structure) Changes

New Minor ODS Number

Firebird 5.0 creates databases with an ODS (On-Disk Structure) version of 13.1. It can also work with databases created in ODS 13.0 (by Firebird 4.0), but some new features will be unavailable.

See [Inline minor ODS upgrade](#) for a new method of upgrading an ODS 13.0 database to 13.1.

New System Tables

System tables added in ODS 13.1:

<code>RDB\$KEYWORDS</code>	Virtual table that enumerates keywords used by the SQL parser
<code>MON\$COMPILED_STATEMENTS</code>	Virtual table that reports compiled statements

New Columns in System Tables

- `RDB$CONDITION_SOURCE` and `RDB$CONDITION_BLR` were added to the system table `RDB$INDICES`, they belong to the partial indices feature
- `MON$SESSION_TIMEZONE` was added to virtual table `MON$ATTACHMENTS`
- `MON$COMPILED_STATEMENT_ID` was added to the virtual tables `MON$STATEMENTS` and `MON$CALL_STACK`
- `SEC$DESCRIPTION` was added to virtual table `SEC$GLOBAL_AUTH_MAPPING`

Application Programming Interfaces

The wire protocol version for the Firebird 5.0 API is 18 (version has been raised to 19 in v5.0.3, see [Inline blobs transmission in the wire protocol](#) below in this chapter). Additions and changes are described in the sections below.

Main API Extensions

A number of new methods have been added to the following interfaces.

ResultSet

```
void getInfo(Status status,
            uint itemsLength, const uchar* items,
            uint bufferLength, uchar* buffer);
```

Used to query cursor information. Currently, only one information item is supported, `INF_RECORD_COUNT`. `INF_RECORD_COUNT` returns the number of records cached by the scrollable cursor, or -1 for a uni-directional (forward-only) cursor.

Extensions to various `getInfo()` Methods

`Statement::getInfo()`

The following actions were added:

<code>isc_info_sql_exec_path_blr_bytes</code>	Execution path as BLR (binary format)
<code>isc_info_sql_exec_path_blr_text</code>	Execution path as BLR (textual format)

Services API Extensions

Support for parallel operations

Added support for parallel operations.

The following options were added:

<code>isc_spb_bkp_parallel_workers</code>	number of parallel workers for backup
<code>isc_spb_res_parallel_workers</code>	number of parallel workers for restore
<code>isc_spb_rpr_par_workers</code>	number of parallel workers for sweep

Examples of use of new parameters in *fbsvcmgr* utility (login and password were left out for brevity):

```
fbsvcmgr -action_backup -bkp_parallel_workers 4 <dbname> <backupname>
fbsvcmgr -action_restore -res_parallel_workers 4 <backupname> <dbname>
fbsvcmgr -action_repair -rpr_sweep_db -rpr_par_workers 4 <dbname>
```

Support for `gfix -upgrade`

Added support for minor ODS upgrade.

The following option was added:

<code>isc_spb_rpr_upgrade_db</code>	upgrade database
-------------------------------------	------------------

Example of use of new parameter in *fbsvcmgr* utility (login and password were left out for brevity):

```
fbsvcmgr -action_repair -rpr_upgrade_db <dbname>
```

Inline blobs transmission in the wire protocol

Firebird 5.0.3 introduces protocol version 19, which provides a significant performance improvement for small blobs. This is achieved by sending small blobs inline during the response to a cursor fetch, removing the need to explicitly open and request the blobs one by one. This reduces a number of roundtrips required to get the blob data and leads to a better performance, especially in high latency networks.

Blob metadata and data is sent using new type of packet `op_inline_blob` and new structure `P_INLINE_BLOB`. The `op_inline_blob` packet is sent before corresponding `op_sql_response` (in the case of response to `op_execute2` or `op_exec_immediate2` requests), or `op_fetch_response` (response to the `op_fetch` request). There could be as many `op_inline_blob` packets as the number of blob fields in output format.

Blobs with the total size less or equal to 64KB are sent inline (NULL blobs are not transmitted). Blobs are sent as a whole, i.e. the current implementation doesn't support transmitting just a part of blob.

Contents of `op_inline_blob` packets is cached by the remote provider in order to serve requests for data in that blobs without network access. Any cached blob is bound to the transaction object and will be released (what happens first):

- at transaction end, or
- when user opens the blob with non-empty BPB, or
- user opens the blob with empty BPB and then closes it

New API methods were added to support this improvement:

```
interface Attachment
...
    // Blob caching by client
    uint getMaxBlobCacheSize(Status status);
    void setMaxBlobCacheSize(Status status, uint size);

    // Inline blob transfer
    uint getMaxInlineBlobSize(Status status);
    void setMaxInlineBlobSize(Status status, uint size);

interface Statement
...
    // Inline blob transfer
    uint getMaxInlineBlobSize(Status status);
    void setMaxInlineBlobSize(Status status, uint size);
```

All methods above are implemented by both *Remote* and *Engine* providers.

Engine provider sets `isc_wish_list` error in status and returns zero, when appropriate. Remote provider checks the protocol version and gets/sets internal object data (no network roundtrip) or

sets `isc_wish_list` error in status and returns zero, when appropriate.

The maximum inline BLOB size defaults to 64KB. This value is set for each prepared statement (using `IStatement::setMaxInlineBlobSize` method) before its execution. If it's set to 0, inline BLOB transmission will be disabled. The default value for newly prepared statements can be overridden using the new DPB tag `isc_dpb_max_inline_blob_size` while connecting to a database, and/or changed at the connection level using `IAttachment::setMaxInlineBlobSize` method.

The size of the client blob cache is limited. The default cache size is 10MB and can be overridden using the new DPB tag `isc_dpb_max_blob_cache_size` while connecting to the database, and/or changed using `Attachment::setMaxBlobCacheSize()` method. There is no upper or lower limit for this value. The limit change is not applied immediately, i.e. if new limit is less than currently used size - nothing happens.

Chapter 5. Reserved Words and Changes

New Keywords in Firebird 5.0

Non-reserved

LOCKED

OPTIMIZE

QUARTER

TARGET

TIMEZONE_NAME

UNICODE_CHAR

UNICODE_VAL

Chapter 6. Configuration Additions and Changes

New configuration parameters:

Parameters for Parallel Operations

MaxParallelWorkers

Limits the total number of parallel workers that can be created within a single Firebird process for each attached database. Integer values in the range between 1 (no parallelism) and 64 are allowed. All other values are silently ignored and the default value of 1 is used.



Workers are accounted for each attached database independently.

ParallelWorkers

Specifies the default number of parallel workers for a single task. Integer values in the range between 1 (no parallelism) and *MaxParallelWorkers* (see above) are allowed. All other values are silently ignored and the default value of 1 is used.

Other Parameters

MaxStatementCacheSize

Defines the maximum amount of memory used to cache unused DSQL compiled statements. A value of zero ('0') means no statement caching is used. Default value is 2 megabytes.

OnDisconnectTriggerTimeout

Configures a timeout (in seconds) that is applied to the ON DISCONNECT trigger execution. The trigger will be automatically cancelled by the engine after the specified time has passed. A value of zero ('0') means no timeout is set. Default value is 180 seconds.

DefaultProfilerPlugin

Specifies the default profiler plugin used to profile connections using the RDB\$PROFILER package.

OptimizeForFirstRows

Defines whether queries should be optimized to retrieve the first records as soon as possible rather than returning the whole dataset as soon as possible. By default, retrieval of all rows is implied by the optimizer.

See also [OPTIMIZE FOR Clause](#) and [SET OPTIMIZE](#).

OuterJoinConversion

Defines whether OUTER joins can be converted into INNER joins by the optimizer, provided that such a transformation is possible from the query result perspective.

Enabled by default. Can be disabled to simplify the migration path if OUTER joins are used intentionally in SQL queries (e.g. as optimizer hints) even if they are known to be semantically equivalent to INNER joins.

SubQueryConversion

Defines whether IN/ANY/EXISTS sub-queries can be unnested and merged with the outer query using the semi-join algorithm, provided that such a transformation is possible.



Being an experimental feature, it's disabled by default.

It can be enabled to benefit from a possibly better performance due to sub-query being evaluated just once and then cached. However, it may also lead to a worse performance in some cases, e.g. when the sub-query returns many rows. Please test with real-world use cases before enabling this conversion in a production environment.

Changed configuration parameters

WireCryptPlugin

A new variant of the ChaCha#20 plugin was added. It uses a 64-bit internal counter rather than 32-bit. The new default value of this parameter is now ChaCha64, ChaCha, Arc4.

Replication Configuration Additions and Changes

cascade_replication

Specifies whether changes applied to the replica database will be also subject of further replication (if any configured). Default value is false (cascading is disabled).

Allow macros in replication.conf

Configuration file macros are now also supported in replication.conf.

Removed configuration parameters

RemotePipeName

This parameter was removed along with the removal of WNET (aka named pipes) protocol support for Windows.

TcpLoopbackFastPath

This parameter was removed because Microsoft discourages using the SIO_LOOPBACK_FAST_PATH socket option.

Chapter 7. Security

Security enhancements in Firebird 5 include:

System privilege PROFILE_ANY_ATTACHMENT

New system privilege PROFILE_ANY_ATTACHMENT has been added to the engine.

When remote SQL profiling is used and the attachment being profiled is from a different user, the calling user must have this system privilege.

See more details in the [SQL and PSQL profiler](#) chapter.

Trace events before a valid security context is established

Alex Peshkov

Tracker ticket: [#7165](#)

A privileged trace session (e.g. by an administrator or user with TRACE_ANY_ATTACHMENT) can now report events (i.e. errors) taking place before validation of an attachment's security context.

As an example:

Set up a conflicting mapping for a user:

```
# ./isql employee
Database: employee, User: SYSDBA
SQL> create user qq password 'qq';
SQL> create global mapping z1 using * from user qq to user z1;
SQL> create global mapping z2 using * from user qq to user z2;
SQL> ^D
```

Because of the conflicting mapping user QQ can not attach to a database even with valid login/password:

```
# ./isql localhost:employee -user qq -pas qq
Statement failed, SQLSTATE = 08004
Multiple maps found for QQ
Use CONNECT or CREATE DATABASE to specify a database
SQL> ^D
```

In the trace output one can see the following:

```
2023-03-17T13:38:41.5240 (25380:0x7f282c10c750) FAILED ATTACH_DATABASE
```

```
employee (ATT_0, QQ, NONE, TCPv4:127.0.0.1/39474)  
/opt/firebird/bin/isql:25396
```

Chapter 8. Management Statements

SET OPTIMIZE

Dmitry Yemanov

Configures whether the optimizer should optimize for fetching first or all rows.

Syntax

```
SET OPTIMIZE <optimize-mode>
```

```
<optimize-mode> ::=  
    FOR {FIRST | ALL} ROWS  
    | TO DEFAULT
```

This feature allows the optimizer to consider another (hopefully better) plan if only a subset of rows is fetched initially by the user application (with the remaining rows being fetched on demand), thus improving the response time.

It can also be specified at the statement level using the [OPTIMIZE FOR](#) clause.

The default behaviour can be specified globally using the *OptimizeForFirstRows* setting in *firebird.conf* or *databases.conf*.

Chapter 9. Data Definition Language (DDL)

Quick Links

- [Support for partial indices](#)
- [COMMENT ON MAPPING](#)

Support for partial indices

Dmitry Yemanov

Tracker ticket: [#7257](#)

This feature allows to index only a subset of table rows defined by the search condition specified during index creation.

Syntax

```
CREATE [UNIQUE] [{ASC[ENDING] | DESC[ENDING]}] INDEX <index_name> ON <table_name>
  { (<column_list>) | COMPUTED [BY] ( <value_expression> ) }
  WHERE <search_condition>
```

Examples

```
-- 1.
CREATE INDEX IT1_COL ON T1 (COL) WHERE COL < 100;
SELECT * FROM T1 WHERE COL < 100;
-- PLAN (T1 INDEX (IT1_COL))

-- 2.
CREATE INDEX IT1_COL2 ON T1 (COL) WHERE COL IS NOT NULL;
SELECT * FROM T1 WHERE COL > 100;
-- PLAN (T1 INDEX IT1_COL2)

-- 3.
CREATE INDEX IT1_COL3 ON T1 (COL) WHERE COL = 1 OR COL = 2;
SELECT * FROM T1 WHERE COL = 2;
-- PLAN (T1 INDEX IT1_COL3)
```

Notes:

1. A partial index definition may include the UNIQUE specification. In this case, every key in the index is required to be unique. This allows to enforce uniqueness across some subset of table rows.
2. A partial index is usable only in the following cases:
 - The WHERE condition includes exactly the same boolean expression as the one defined for

the index;

- The search condition defined for the index contains ORed boolean expressions and one of them is explicitly included in the WHERE condition;
- The search condition defined for the index specifies IS NOT NULL and the WHERE condition includes an expression on the same field that is known to ignore NULLs.

COMMENT ON MAPPING

Alex Peshkov

Tracker ticket: [#7046](#)

The COMMENT ON statement was extended to be able to add a comment to a MAPPING.

```
COMMENT ON [GLOBAL] MAPPING <mapping name> IS {<comment> | NULL};
```

Chapter 10. Data Manipulation Language (DML)

Quick Links

- `SKIP LOCKED` clause
- Support for `WHEN NOT MATCHED BY SOURCE` in the `MERGE` statement
- Support multiple rows for `DML RETURNING`
- Allow parenthesized query expressions
- Support `PLAN` and `ORDER BY` on `MERGE`
- Support `PLAN`, `ORDER BY` and `ROWS` on `UPDATE OR INSERT`
- `OPTIMIZE FOR` Clause
- Changes to literals
- Improvements to `IN`
- New Expressions and Built-in Functions

SKIP LOCKED clause

Adriano dos Santos Fernandes

Tracker ticket: [#7350](#)

`SKIP LOCKED` can be used with `SELECT ... WITH LOCK`, `UPDATE` and `DELETE` statements. It makes the engine skip records locked by other transactions instead of waiting on them or raise conflict errors.

This is very useful to implement work queues where one or more processes post work to a table and issue an event, while workers listen for events and read/delete items from the table. Using `SKIP LOCKED` multiple workers can get exclusive work items from the table without conflicts.

Syntax

```
SELECT
  [FIRST ...]
  [SKIP ...]
FROM <sometable>
[WHERE ...]
[PLAN ...]
[ORDER BY ...]
[{ ROWS ... } | {OFFSET ...} | {FETCH ...}]
[FOR UPDATE [OF ...]]
[WITH LOCK [SKIP LOCKED]]
```

```
UPDATE <sometable>
SET ...
```

```
[WHERE ...]
[PLAN ...]
[ORDER BY ...]
[ROWS ...]
[SKIP LOCKED]
[RETURNING ...]
```

```
DELETE FROM <sometable>
  [WHERE ...]
  [PLAN ...]
  [ORDER BY ...]
  [ROWS ...]
  [SKIP LOCKED]
  [RETURNING ...]
```



If a statement has both `SKIP LOCKED` and `OFFSET/SKIP/ROWS` subclauses, locked rows may be skipped before `OFFSET/SKIP/ROWS` subclause can account for them, thus skipping more rows than specified in `OFFSET/SKIP/ROWS`.

Examples:

- Prepare metadata

```
create table emails_queue (
  subject varchar(60) not null,
  text blob sub_type text not null
);

set term !;

create trigger emails_queue_ins after insert on emails_queue
as
begin
  post_event('EMAILS_QUEUE');
end!

set term ;!
```

- Sender application or routine

```
insert into emails_queue (subject, text)
  values ('E-mail subject', 'E-mail text...');
commit;
```

- Client application

```
-- Client application can listen to event 'EMAILS_QUEUE' to actually send e-mails
```

using this query:

```
delete from emails_queue
rows 10
skip locked
returning subject, text;
```

More than one instance of the application may be running, for example to load balance work.

Support for WHEN NOT MATCHED BY SOURCE in the MERGE statement

Adriano dos Santos Fernandes

Tracker ticket: [#6681](#)

Syntax

```
<merge when> ::=
    <merge when matched> |
    <merge when not matched by target> |
    <merge when not matched by source>

<merge when not matched by target> ::=
    WHEN NOT MATCHED [ BY TARGET ] [ AND <condition> ] THEN
        INSERT [ <left paren> <column list> <right paren> ]
            VALUES <left paren> <value list> <right paren>

<merge when not matched by source> ::=
    WHEN NOT MATCHED BY SOURCE [ AND <condition> ] THEN
        { UPDATE SET <assignment list> | DELETE }
```

<merge when not matched by target> is called when a source record matches no record in target. INSERT will change the target table.

<merge when not matched by source> is called when a target record matches no record in source. UPDATE or DELETE will change the target table.

Example

```
MERGE
  INTO customers c
  USING new_customers nc
  ON (c.id = nc.id)
  WHEN MATCHED THEN
    UPDATE SET name = nc.name
  WHEN NOT MATCHED BY SOURCE THEN
    DELETE
```


Support multiple rows for DML RETURNING

Adriano dos Santos Fernandes

Tracker ticket: [#6815](#)

In DSQL, the RETURNING clause is now able to return multiple rows for DML statements than can affect multiple rows.

See [compatibility notes on RETURNING](#) for more information.

Allow parenthesized query expressions

Adriano dos Santos Fernandes

Tracker ticket: [#6740](#)

The DML syntax was extended to allow a parenthesized *query expression* (select including order by, offset and fetch clauses, but without with clause) to occur where previously only a *query specification* (select without with, order by, offset and fetch clauses) was allowed.

This allows more expressive queries, especially in UNION statements, and offers more compatibility with statements generated by certain ORMs.



Using parenthesized *query expressions* comes at a cost, as they consume an additional query context compared to a plain *query specification*. The maximum number of query contexts in a statement is 255.

Example

```
(
  select emp_no, salary, 'lowest' as type
  from employee
  order by salary asc
  fetch first row only
)
union all
(
  select emp_no, salary, 'highest' as type
  from employee
  order by salary desc
  fetch first row only
);
```

Support PLAN and ORDER BY on MERGE

The MERGE statement now supports the PLAN and ORDER BY clauses.

Syntax

```

MERGE INTO target [[AS] target_alias]
  USING <source> [[AS] source_alias]
  ON <join_condition>
  <merge_when> [<merge_when> ...]
  [PLAN <plan-expr>]
  [ORDER BY <ordering-list>]
  [RETURNING <returning_list> [INTO <variables>]]

```

Support PLAN, ORDER BY and ROWS on UPDATE OR INSERT

The UPDATE OR INSERT statement now supports the PLAN, ORDER BY and ROWS clauses.

Syntax

```

UPDATE OR INSERT INTO
  target [(<column_list>)]
  [<override_opt>]
  VALUES (<value_list>)
  [MATCHING (<column_list>)]
  [PLAN <plan-expr>]
  [ORDER BY <ordering-list>]
  [ROWS <m> [TO <n>]]
  [RETURNING <returning_list> [INTO <variables>]]

```

OPTIMIZE FOR Clause

Dmitry Yemanov

SELECT statements now support the OPTIMIZE FOR clause.

Syntax

```

SELECT
  ...
  [WITH LOCK [SKIP LOCKED]]
  [OPTIMIZE FOR {FIRST | ALL} ROWS]

```

The OPTIMIZE FOR clause can only occur on a top-level SELECT.

This feature allows the optimizer to consider another (hopefully better) plan if only a subset of rows is fetched initially by the user application (with the remaining rows being fetched on demand), thus improving the response time.

It can also be specified at the session level using the [SET OPTIMIZE](#) management statement.

The default behaviour can be specified globally using the *OptimizeForFirstRows* setting in

firebird.conf or *databases.conf*.

Changes to literals

Full SQL standard character string literal syntax

Adriano dos Santos Fernandes

Tracker ticket: [#5589](#)

The syntax of character string literals was changed to support the full SQL standard syntax. This means a literal can be “interrupted” by whitespace or a comment. This can be used, for example, to break up a long literal over several lines, or provide inline comments.

```
<character string literal> ::=
  [ <introducer> <character set specification> ]
  <quote> [ <character representation>... ] <quote>
  [ { <separator> <quote> [ <character representation>... ] <quote> }... ]

<separator> ::=
  { <comment> | <white space> }...
```

— ISO/IEC 9075-2:2016 SQL - Part 2: Foundation

Examples

```
-- whitespace between literal
select 'ab'
       'cd'
from RDB$DATABASE;
-- output: 'abcd'

-- comment and whitespace between literal
select 'ab' /* comment */ 'cd'
from RDB$DATABASE;
-- output: 'abcd'
```

Full SQL standard binary string literal syntax

Adriano dos Santos Fernandes

Tracker ticket: [#5588](#)

The syntax of binary string literals was changed to support the full SQL standard syntax. This means a literal can contain spaces to separate hexadecimal characters, and it can be “interrupted” by whitespace or a comment. This can be used, for example, to make the hex string more readable by grouping characters, or to break up a long literal over several lines, or provide inline comments.

```

<binary string literal> ::=
  X <quote> [ <space>... ] [ { <hexit> [ <space>... ] <hexit> [ <space>... ] }...
] <quote>
  [ { <separator> <quote> [ <space>... ] [ { <hexit> [ <space>... ]
<hexit> [ <space>... ] }... ] <quote> }... ]

```

— ISO/IEC 9075-2:2016 SQL - Part 2: Foundation

Examples

```

-- Group per byte (whitespace inside literal)
select _win1252 x'42 49 4e 41 52 59'
from RDB$DATABASE;
-- output: BINARY

-- whitespace between literal
select _win1252 x'42494e'
                '415259'
from RDB$DATABASE;
-- output: BINARY

```



The usage of the `_win1252` introducer in above example is a non-standard extension and equivalent to an explicit cast to a `CHAR` of appropriate length with character set `WIN1252`.

Improvements to IN

Dmitry Yemanov

Processing of `IN <list>` predicates is now linear rather than recursive, thus no runtime stack limitations. The limit of 1500 items has been raised to 65535 items.

Lists that are known to be constant are pre-evaluated as invariants and cached as a binary search tree, making comparisons faster if the condition needs to be tested for many rows or if the value list is long.

If the list is very long or if the `IN` predicate is not selective, the index scan supports searching groups using the sibling pointer (i.e. horizontally) rather than searching every group from the root (i.e. vertically), thus utilizing a single index scan for the whole `IN` list.

New Expressions and Built-in Functions

UNICODE_CHAR and UNICODE_VAL

Adriano dos Santos Fernandes

UNICODE_CHAR

Returns the Unicode character with the specified code point.

Syntax

```
UNICODE_CHAR( <number> )
```



The argument to UNICODE_CHAR must be a valid Unicode code point and not in the range of high/low surrogates (0xD800 to 0xDFFF), otherwise it throws an error.

Example

```
select unicode_char(x) from y;
```

UNICODE_VAL

Returns the Unicode code point of the first character of the specified string, or zero if the string is empty.

Syntax

```
UNICODE_VAL( <string> )
```

Example

```
select unicode_val(x) from y;
```

QUARTER added to EXTRACT, FIRST_DAY and LAST_DAY

The scalar functions EXTRACT, FIRST_DAY and LAST_DAY now support the date part QUARTER.

Example

```
select
  extract(quarter from date '2023-09-21') as Q,
  first_day(of quarter from date '2023-09-21') as Q_START,
  last_day(of quarter from date '2023-09-21') as Q_END
from rdb$database;
```

Q	Q_START	Q_END
3	2023-07-01	2023-09-30

Chapter 11. Procedural SQL (PSQL)

Allow subroutines to access variables/parameters defined at the outer/parent level

Adriano dos Santos Fernandes

Tracker ticket: [#4769](#)

Subroutines can now read from and write to variables and parameters of the outer/parent routine. This does not apply to cursors: subroutines cannot access cursors of their parent.

Variables and parameters that are accessed by subroutines may have a small performance penalty (even in the main routine) when being read.



The restriction on cursors may be lifted in a future version.

See tracker ticket [#7120](#).

Chapter 12. Monitoring & Command-line Utilities

Improvements and additions to the Firebird utilities continue.

Monitoring

New virtual tables:

RDB\$KEYWORDS:

RDB\$KEYWORD_NAME	Keyword name
RDB\$KEYWORD_RESERVED	Whether keyword is a reserved word

MON\$COMPILED_STATEMENTS:

MON\$COMPILED_STATEMENT_ID	Compiled statement ID
MON\$SQL_TEXT	Text of the SQL query
MON\$EXPLAINED_PLAN	Plan (in the explained form) of the SQL query
MON\$OBJECT_NAME	PSQL object name
MON\$OBJECT_TYPE	PSQL object type
MON\$PACKAGE_NAME	Package name of the PSQL object
MON\$STAT_ID	Runtime statistics ID (references MON\$*_STATS tables)

New columns in the tables:

In MON\$ATTACHMENTS:

MON\$SESSION_TIMEZONE	Actual timezone of the session
-----------------------	--------------------------------

In MON\$STATEMENTS:

MON\$COMPILED_STATEMENT_ID	Compiled statement ID (references MON\$COMPILED_STATEMENTS)
----------------------------	---

In MON\$CALL_STACK:

MON\$COMPILED_STATEMENT_ID	Compiled statement ID (references MON\$COMPILED_STATEMENTS)
----------------------------	---

In SEC\$GLOBAL_AUTH_MAPPING:

SEC\$DESCRIPTION	Textual description
------------------	---------------------

nbackup

Clean Backup History

Vlad Khorsun

Tracker ticket [#7186](#)

Two options were added to clean the backup history of *nbackup*:

1. The `-clean_history` and `-keep n {rows | days}` option that can be specified with ``-backup`.

This retains the last *n* rows, or last *n* days before today.

2. Manual deletion from `RDB$BACKUP_HISTORY`; this is available for administrators and for users with the `USE_NBACKUP_UTILITY` system privilege



Be careful with deleting backup history using `-CLEAN_HISTORY` or manual deletion. If you delete too soon, or the wrong records, you can break your level or GUID chain. That can result in *nbackup* not knowing where to start a next backup, requiring you, for example, to start backing up from level 0 again.

isql

Unify display of system procedures and functions & packages with other system objects

Alex Peshkov

Tracker tickets: [#7411](#), [#7475](#)

The `SHOW SYSTEM` command of *isql* now lists system packages and their procedures and functions.

The `SHOW SYSTEM <object-type>` now supports the additional object types `PROCEDURES`, `PACKAGES`, and `PUBLICATIONS` (see also [below](#)).

The equivalent “normal” `SHOW <object-type>` commands no longer list system functions, procedures or packages.

Display statement BLR

Adriano dos Santos Fernandes

Tracker ticket: [#6910](#)

The `SET EXEC_PATH_DISPLAY` command is a debug command to show the BLR (compiled form) of the statement. When a statement is executed, it retrieves the compiled execution path of a DML statement formatted as BLR text.


```
SET EXEC_PATH_DISPLAY {BLR | OFF};
```



This feature is intimately tied to engine internals. Its usage is discouraged if you do not understand very well how these internals are subject to change between versions.

Replication information added to SHOW output

Dmitry Yemanov

Tracker tickets: [#7001](#), [#7425](#)

- The `SHOW DATABASE` command now reports the “Replica mode” and “Publication” state
- The `SHOW TABLE name` command now reports the publications that include the table
- The `SHOW PUBLICATIONS [name]` command will show the named publication, or — without *name* — all user-defined publications in the current database



Currently, Firebird doesn't support user-defined publication, so `SHOW PUBLICATIONS` currently will never list any publications.

- The command `SHOW SYSTEM` now also lists system publications, and `SHOW SYSTEM PUBLICATIONS` was added to only show system publications

Display per-table statistics

Vlad Khorsun

Tracker tickets (pull requests): [#7218](#)

isql can now show per-table statistics for an executed statement.

```
SET PER_TABLE_stats [ON | OFF]
```

This command turns the display of per-table statistics on or off as desired. If no parameter is supplied to the command, it toggles the current state of the per-table statistics display.

When set to ON, *isql* shows per-table run-time statistics after query execution. It is set to OFF by default. This command is independent of the `SET STATS` command. The name `PER_TABLE_STATS` can be abbreviated up to `PER_TAB`. Tables in the output are sorted by their relation id's.

Example (width reduced from original output):

```
-- check current value
SQL> SET;
...
Print per-table stats:  OFF
```

...

```
-- turn per-table stats on
SQL> SET PER_TABLE_STATS ON;
SQL> SELECT COUNT(*) FROM RDB$RELATIONS JOIN RDB$RELATION_FIELDS USING
(RDB$RELATION_NAME);
```

```

COUNT
=====
534
```

Per table statistics:

Table name	Natural	Index	Insert	Update	Delete	Backout	Purge	Expunge
RDB\$INDICES		3						
RDB\$RELATION_FIELDS		534						
RDB\$RELATIONS	59							
RDB\$SECURITY_CLASSES		3						

Note, some system tables are shown that were not listed in the query; the engine reads some additional metadata when preparing the query.

```
-- turn per-table stats off, using shortened name
SQL> SET PER_TAB OFF;
```

gbak

Parallel backup/restore

Vlad Khorsun

Tracker tickets: [#1783](#), [#3374](#)

A new command-line switch has been added to *gbak*: `-PAR[ALLEL] <N>`.

It defines how many parallel workers will be used for the requested task.

Usage examples:

```
gbak -b -par 4 -user <username> -pass <password> <dbname> <backupname>
gbak -r -par 4 -user <username> -pass <password> <backupname> <dbname>
```

gfix

Parallel sweep and ICU dependencies rebuild

Vlad Khorsun

Tracker tickets: [#7447](#), [#7550](#)

A new command-line switch has been added to *gfix*: `-PAR[ALLEL] <N>`.

It defines how many parallel workers will be used for the requested task.

Usage example:

```
gfix -sweep -par 4 -user <username> -pass <password> <dbname>
gfix -icu -par 4 -user <username> -pass <password> <dbname>
```

The `-parallel` option is only valid in combination with the `-sweep` and `-icu` tasks.

ODS upgrade

Dmitry Yemanov

Tracker tickets: [#7397](#)

A new command-line switch has been added to *gfix*: `-UP[GRADE]`.

It allows to upgrade ODS of the database to the latest supported minor version (within the supported major version).

Usage example(s):

```
gfix -upgrade <dbname> -user <username> -pass <password>
```

Chapter 13. Compatibility Issues

This section lists features and modifications that might affect the way you have installed and used Firebird in earlier releases.

Migrating the security database from Firebird 4.0 to 5.0

To migrate the security database from Firebird 4.0 to 5.0, backup `security4.fdb` with *gbak* of Firebird 4.0 and restore it as `security5.fdb` with *gbak* of Firebird 5.0. Use *gbak* locally (using an embedded connection) while Firebird Server is not running.



Copying `security4.fdb` and renaming it to `security5.fdb` and upgrading the ODS with *gfix* option `-UPGRADE` will work as well, but we recommend backing up and restoring.

SQL

Changes that may affect existing SQL code:

Multi-row RETURNING behaviour

Client-side `INSERT ... SELECT`, `UPDATE`, `DELETE`, `MERGE` and `UPDATE OR INSERT` queries containing the `RETURNING` clause may now return multiple records instead of raising error “multiple rows in singleton select” as it happened before.

These queries are now described as *isc_info_sql_stmt_select* during preparation, while in previous versions they were described as *isc_info_sql_stmt_exec_procedure*.

Singleton `INSERT ... VALUES` statements, as well as positioned `UPDATE` and `DELETE` statements (i.e. the ones containing the `WHERE CURRENT OF` clause) preserve the existing behaviour, being described as *isc_info_sql_stmt_exec_procedure*. They also preserve the ability of being executed within a single protocol roundtrip to the server.

However, all these queries, if used in PSQL and the `RETURNING` clause is applied, are still treated as singleton.

Removal of WNET protocol

Network protocol *WNET* (a.k.a. Named Pipes, a.k.a. NetBEUI) previously supported on Windows platform is removed in Firebird 5.0. Those Windows users who operated with any *WNET* connection string (`\\server\dbname` or `wnet://server/dbname`) should switch to *INET* (TCP) protocol instead (connection string `server:dbname`, `server/port:dbname`, `inet://server/dbname`, or `inet://server:port/dbname`).

Removal of QLI

Command-line utility *QLI* is removed in Firebird 5.0, in accordance with its deprecation announcement published in the Firebird 4.0 release notes.

Removal of *fbclient_bor.lib*

API import library for Embarcadero (former Borland) compilers *fbclient_bor.lib* is no longer distributed in Firebird 5.0 packages for Windows x86 platform. Anyone affected by this change may generate it using the *implib.exe* utility supplied with Embarcadero (former Borland) development tools as documented [here](#).

Chapter 14. Bugs Fixed

Firebird 5.0.3 Release: Bug Fixes

Core Engine

#8592 — Presence of 'ROWS <n_limit>' causes garbage in string conversion error message

Implemented by Adriano dos Santos Fernandes

#8590 — Line “BLR to Source mapping:” may look broken when `procedures.rdb$debug_info` is queried using remote protocol

Implemented by Vlad Khorsun

#8589 — `PERCENT_RANK` may return `NaN` instead of 0

Implemented by Adriano dos Santos Fernandes

#8554 — Vulnerability GHSA-7qp6-hqxj-pjjp / ZDI-CAN-26486

Implemented by Alexander Peshkov

#8520 — Error in `iTransaction.getInfo()` on embedded connection

Implemented by Alexander Peshkov

#8509 — “Error creating private namespace” message in *firebird.log*

Implemented by Vlad Khorsun

#8508 — Conversion error with `old.field` inside `UPDATE OR INSERT`

Implemented by Adriano dos Santos Fernandes

#8488 — `MIN/MAX` aggregates may badly affect the join order in queries with mixed `INNER/LEFT` joins

Implemented by Dmitry Yemanov

[#8487](#) — Unexpected *SetFilePointer* error when remapping the shared memory

Implemented by Vlad Khorsun

[#8477](#) — Inheritance of WINDOW does not work

Implemented by Adriano dos Santos Fernandes

[#8449](#) — Races when server is closed during forced database shutdown

Implemented by Alexander Peshkov

[#8440](#) — Wrong result for MINVALUE/MAXVALUE with string arguments

Implemented by Dmitry Yemanov

[#8431](#) — Make sure only one error will be sent to not-started Service

Implemented by Alexander Peshkov

[#8430](#) — Unstable error messages in services with trace enabled

Implemented by Artyom Abakumov

[#8320](#) — Do not allow concurrent run of sweep instances

Implemented by Alexander Peshkov

[#8194](#) — Internal consistency check (page in use during flush) with small number of *DefaultDbCachePages*

Implemented by Vlad Khorsun

[#8182](#) — IN predicate incorrectly handles single parenthesized subquery as IN-list, instead of table subquery

Implemented by Mark Rotteveel

#8139 — Conflict resolution code uses constraint name instead of index name

Implemented by Andrey Kravchenko, Vlad Khorsun

Server Crashes/Hang-ups

#8649 — AV when ON CONNECT triggers use EXECUTE STATEMENT ON EXTERNAL

Implemented by Vlad Khorsun

#8437 — Segmentation fault when running query with PARTITION BY and subquery

Implemented by Adriano dos Santos Fernandes

#8485 — Segfault/AV on incorrect *databases.conf* starting with subconfig (without line alias=database_path)

Implemented by Alexander Peshkov

#8432 — Fix server hang when trying to stop a server with active connections with event notifications

Implemented by Artyom Ivanov

#8296 — Crash in TipCache::findStates

Implemented by Vlad Khorsun

Utilities

#8588 — Error doing *nbackup* when database file name has extended ASCII chars

Implemented by Alexander Peshkov

#8524 — *ISQL* truncates lines longer than 255 characters when pasting

Implemented by Vlad Khorsun

Firebird 5.0.2 Release: Bug Fixes

Core Engine

[#8429](#) — Segfault when already destroyed callback interface is used

Implemented by Alexander Peshkov

[#8417](#) — Negative statement ID in trace output

Implemented by Dmitry Yemanov

[#8413](#) — Fix incorrect maximum size when reading database parameter values in SHOW DATABASE

Implemented by Artyom Ivanov

[#8408](#) — Add option to disable install of *MSVCRT* runtime libraries via a scripted install

Implemented by Paul Reeves

[#8407](#) — *InnoSetup* based installer deletes *msiexec* log of runtime libraries install

Implemented by Paul Reeves

[#8403](#) — Fix potential deadlock when starting the encryption thread

Implemented by Alexander Peshkov

[#8390](#) — Deadlock might happen when database is shutting down with internal worker attachments

Implemented by Vlad Khorsun

[#8389](#) — Unnecessary reload of the encryption plugin in the *SuperServer* architecture

Implemented by Alexey Mochalov

[#8386](#) — Crash when creating index on table that uses UDR and `ParallelWorkers > 1`

Implemented by Vlad Khorsun

[#8380](#) — Fix race condition in shutdownThread start that can cause server to be unable to stop

Implemented by Artyom Ivanov

[#8379](#) — Incorrect cardinality estimation for retrievals with multiple compound indices having common set of fields

Implemented by Dmitry Yemanov

[#8350](#) — Missing records in the replicated database

Implemented by Vlad Khorsun

[#8341](#) — Cleanup batches inside the engine if they were not released explicitly before disconnection

Implemented by Dmitry Yemanov

[#8336](#) — Error: “Invalid clumplet buffer structure: buffer end before end of clumplet - clumplet too long” when using trusted authentication

Implemented by Vlad Khorsun

[#8334](#) — MacOS ARM version requires Rosetta

Implemented by Adriano dos Santos Fernandes

[#8327](#) — Fix use :@ characters and add sub-section to configure username and password for sync_replica

Implemented by Andrey Kravchenko

[#8324](#) — Make asynchronous replica re-initialization reliable

Implemented by Dmitry Yemanov

[#8319](#) — Use a 64-bit counter for records written during backup

Implemented by Dmitry Starodubov

[#8315](#) — Crash at database restore due to failed system call

Implemented by Vlad Khorsun, Ilya Eremin

[#8304](#) — Wrong results using MINVALUE/MAXVALUE in join condition

Implemented by Adriano dos Santos Fernandes

[#8292](#) — run_all PDB fails with "Error calling COPY_XTRA"

Implemented by Vlad Khorsun

[#8290](#) — "Unique scan" is incorrectly reported in the explained plan for unique index and IS NULL predicate

Implemented by Dmitry Yemanov

[#8288](#) — GPRE generated code is incompatible with GCC 14.2

Implemented by Adriano dos Santos Fernandes

[#8283](#) — Assertion in ~thread_db() due to not released page buffer

Implemented by Vlad Khorsun

[#8268](#) — Fix refetch header data from delta when database is in the backup lock state

Implemented by Andrey Kravchenko

[#8265](#) — Nested IN/EXISTS subqueries should not be converted into semi-joins if the outer context is a sub-query which wasn't unnested

Implemented by Dmitry Yemanov

[#8263](#) — `gbak` on `Classic` with `ParallelWorkers > 1` doesn't restore indices, giving a cryptic error message

Implemented by Vlad Khorsun

[#8262](#) — Make server correctly handle case when `accept()` returns both success and data for client

Implemented by Vlad Khorsun

[#8255](#) — Catch possible stack overflow when preparing and compiling user statements

Implemented by Vlad Khorsun

[#8253](#) — Incorrect handling of non-ASCII object names in `CREATE MAPPING` statement

Implemented by Vlad Khorsun

[#8252](#) — Incorrect subquery unnesting with complex dependencies (`SubQueryConversion = true`)

Implemented by Dmitry Yemanov

[#8250](#) — Bad performance on simple two joins query on tables with a composite index

Implemented by Dmitry Yemanov

[#8243](#) — Fix a bug where the CS shutdown handler could be called again

Implemented by Alexander Zhdanov

[#8241](#) — `gbak` may lose NULLs in restore

Implemented by Adriano dos Santos Fernandes

[#8238](#) — Fix using macro with regex in path parameter in `fbtrace.conf`

Implemented by Artyom Ivanov

[#8237](#) — Database access error when `nbackup` is starting

Implemented by Alexander Peshkov

[#8236](#) — Avoid "hangs" in `clock_gettime()` in *tomcrypt*'s PRNG

Implemented by Alexander Peshkov

[#8233](#) — `SubQueryConversion = true` — error "multiple rows in singleton select"

Implemented by Dmitry Yemanov

[#8224](#) — `SubQueryConversion = true` — wrong resultset with FIRST/SKIP clauses inside the outer query

Implemented by Dmitry Yemanov

[#8223](#) — `SubQueryConversion = true` — error "no current record for fetch operation" with complex joins

Implemented by Dmitry Yemanov

[#8222](#) — Fix a case of deleted memory modification

Implemented by Ilya Eremin

[#8221](#) — Crash when function `MAKE_DBKEY` is called with 0 or 1 arguments

Implemented by Vlad Khorsun

[#8219](#) — Database creation in 3.0.12, 4.0.5 and 5.0.1 slower than in previous releases

Implemented by Adriano dos Santos Fernandes

[#8215](#) — Rare sporadic segfaults in test for *core-6142* on Windows

Implemented by Alexander Peshkov

[#8214](#) — Incorrect result of index list scan for a composite index, the second segment of which is a text field with `COLLATE UNICODE_CI`

Implemented by Dmitry Yemanov

#8213 — WHEN NOT MATCHED BY SOURCE - does not work with a direct table as source

Implemented by Adriano dos Santos Fernandes

#8211 — DATEADD truncates milliseconds for month and year deltas

Implemented by Adriano dos Santos Fernandes

#8203 — Function MAKE_DBKEY may produce random errors if used with a relation name

Implemented by Vlad Khorsun

#8109 — Plan/performance regression when using special construct for IN

Implemented by Vlad Khorsun, Dmitry Yemanov

#8069 — Add missing synchronization to cached vectors of known pages

Implemented by Dmitry Yemanov, Vlad Khorsun

Firebird 5.0.1 Release: Bug Fixes

Core Engine

#8189 — Slow connection times with a lot of simultaneous connections and active trace session present

Implemented by Alexander Peshkov, Vlad Khorsun

#8186 — Fixed a few issues with IPC used by remote profiler

Implemented by Vlad Khorsun

#8180 — Sometimes a trace session is terminated spontaneously

Implemented by Artyom Abakumov

#8178 — Fix boolean conversion to string inside `DataTypeUtil::makeFromList()`

Implemented by Dmitry Yemanov

[#8171](#) — Trace plugin unloaded if called method is not implemented

Implemented by Vlad Khorsun

[#8168](#) — MAKE_DBKEY bug after backup/restore

Implemented by Vlad Khorsun

[#8156](#) — Can not specify concrete IPv6 address in ES/EDS connection string

Implemented by Vlad Khorsun

[#8150](#) — Process could attach to the deleted instance of shared memory

Implemented by Alexander Peshkov, Vlad Khorsun

[#8138](#) — Bugcheck when replicator state is changed concurrently

Implemented by Vlad Khorsun

[#8123](#) — Procedure manipulation can lead to wrong dependencies removal

Implemented by Adriano dos Santos Fernandes

[#8120](#) — CAST dies with numeric value is out of range error

Implemented by Vlad Khorsun

[#8115](#) — FB 5.0.0.1306 - unexpected results using LEFT JOIN with WHEN

Implemented by Dmitry Yemanov

[#8112](#) — Error `isc_read_only_trans (335544361)` should report `SQLSTATE 25006`

Implemented by Adriano dos Santos Fernandes

[#8108](#) — ICU 63.1 suppresses conversion errors

Implemented by Dmitry Kovalenko

[#8100](#) — The `isc_array_lookup_bounds` function returns invalid values for low and high array bounds

Implemented by Adriano dos Santos Fernandes

[#8094](#) — Index creation error when restoring with parallels workers

Implemented by Vlad Khorsun

[#8086](#) — IN predicate with string-type elements is evaluated wrongly against a numeric field

Implemented by Dmitry Yemanov

[#8085](#) — Memory leak when executing a lot of different queries and `StatementTimeout > 0`

Implemented by Vlad Khorsun

[#8084](#) — Partial index uniqueness violation

Implemented by Vlad Khorsun

[#8078](#) — `SIMILAR TO` with constant pattern using `'|'`, `'*'`, `'?'` or `'{0,N}'` doesn't work as expected

Implemented by Adriano dos Santos Fernandes

[#8077](#) — Error "Too many recursion levels" does not stop execution of code that uses `ON DISCONNECT` trigger (FB 4.x+)

Implemented by Alexander Peshkov, Vlad Khorsun

[#8063](#) — (var)char variables/parameters assignments fail in stored procedures with subroutines

Implemented by Adriano dos Santos Fernandes

[#8058](#) — Replicated DDL changes do not set the correct grantor

Implemented by Dmitry Yemanov

[#8056](#) — Error "Too many temporary blobs" with BLOB_APPEND when selecting a stored procedure with rows-clause

Implemented by Vlad Khorsun

[#8040](#) — Bugcheck 183 (wrong record length) could happen on replica database after UK violation on insert

Implemented by Vlad Khorsun

[#8033](#) — Invalid result when string compared with indexed NUMERIC(x,y) field where x > 18 and y != 0

Implemented by Alexander Peshkov

[#8027](#) — Broken *gbak* statistics

Implemented by Alexander Peshkov

[#8011](#) — DECFLOAT error working with INT128 in UDR

Implemented by Alexander Peshkov

[#8006](#) — INT128 datatype not supported in FB_MESSAGE macro

Implemented by Alexander Peshkov

[#7997](#) — Unexpected results when comparing integer with string containing value out of range of that integer datatype

Implemented by Alexander Peshkov

[#7995](#) — Unexpected results after creating partial index

Implemented by Dmitry Yemanov

[#7993](#) — Unexpected results when using CASE WHEN with RIGHT JOIN

Implemented by Dmitry Yemanov

[#7976](#) — False validation error for short unpacked records

Implemented by Dmitry Yemanov

[#7969](#) — Characters are garbled when replicating fields with type BLOB SUB_TYPE TEXT if the character set of the connection and the field are different

Implemented by Dmitry Yemanov

[#7942](#) — Database file appears corrupted after restore from backup

Implemented by Vlad Khorsun

[#7937](#) — Inner join raises error "no current record for fetch operation" if a stored procedure depends on some table via input parameter and also has an indexed relationship with another table

Implemented by Dmitry Yemanov

[#7927](#) — Some default values are set incorrectly for SC/CS architectures

Implemented by Vlad Khorsun

[#7921](#) — Firebird 5 uses PK for ordered plan even if matching index with fewer fields exists

Implemented by Dmitry Yemanov

[#7899](#) — Inconsistent state of master-detail occurs after reconnect + 'SET AUTODDL OFF' + 'drop <FK>' which is rolled back

Implemented by Vlad Khorsun

[#7896](#) — *replication.log* remains empty (and without any error in *firebird.log*) until concurrent FB instance is running under different account and generates segments on its master. Significant delay

required after stopping concurrent FB to allow first one to write in its replication log.

Implemented by Vlad Khorsun

#7873 — Wrong memory buffer alignment and I/O buffer size when working in direct I/O mode

Implemented by Vlad Khorsun

#7863 — Non-correlated sub-query is evaluated multiple times if it is based on a view rather than on an equivalent derived table

Implemented by Dmitry Yemanov

Server Crashes/Hangups

#8185 — SIGSEGV in Firebird 5.0.0.1306 embedded during update on cursor

Implemented by Adriano dos Santos Fernandes, Dmitry Yemanov

#8176 — Firebird 5 hangs after starting remote profiling session

Implemented by Vlad Khorsun

#8151 — Deadlock happens when running 'List Trace Sessions' service and there are many active trace sessions

Implemented by Vlad Khorsun

#8149 — A hang or crash could happen when connection fires *TRACE_EVENT_DETACH* event and a new trace session is created concurrently

Implemented by Vlad Khorsun

#8136 — Server crashes with IN (dbkey1, dbkey2, ...) condition

Implemented by Dmitry Yemanov

#8114 — Segfault in connection pool during server shutdown

Implemented by Vlad Khorsun

[#8110](#) — Firebird 5 crash on Android API level 34

Implemented by Vlad Khorsun

[#8101](#) — Firebird crashes if a plugin factory returns *nullptr* and no error in the status

Implemented by Vlad Khorsun, Dmitry Sibiryakov

[#8089](#) — AV when attaching database while low on free memory

Implemented by Vlad Khorsun

[#8087](#) — AV when preparing a query with IN <list> that contains both literals and sub-query

Implemented by Vlad Khorsun

[#8083](#) — AV when writing to internal trace log

Implemented by Vlad Khorsun

[#8079](#) — Engine could crash when executing some trigger(s) while another attachment modifies them

Implemented by Vlad Khorsun

[#8039](#) — Segfault when opening damaged (last TIP is missing in *RDB\$PAGES*, user's FW was OFF) database

Implemented by Alexander Peshkov

[#8026](#) — Crash LI-V5.0.0.1306 in *libEngine13.so*

Implemented by Alexander Peshkov

[#7998](#) — Crash during partial index checking if the condition raises a conversion error

Implemented by Dmitry Yemanov

[#7985](#) — Hang in case of error when sweep thread is attaching to database (*Classic Server*)

Implemented by Alexander Peshkov

[#7979](#) — Hang when database with disconnect trigger using MON\$ tables is shutting down

Implemented by Alexander Peshkov

Utilities

isql

[#8016](#) — Free memory issued for *isql* command list but has never been freed on output file write

Implemented by Alexey Mochalov

[#7962](#) — System procedure/function inconsistency between *isql* SHOW FUNCTIONS and SHOW PROCEDURES

Implemented by Artyom Ivanov

gbak

[#8003](#) — *gbak* v4 can't backup database in ODS < 13

Implemented by Vlad Khorsun

[#7996](#) — *gbak* terminates/crashes when a read error occurs during restore

Implemented by Vlad Khorsun

[#7992](#) — Assertion (space > 0) failure during restore

Implemented by Vlad Khorsun

[#7974](#) — Restore of wide table can fail with "adjusting an invalid decompression length from <N> to <M>"

Implemented by Vlad Khorsun

#7950 — Unable to restore database when .fbk was created on host with other ICU

Implemented by Alexander Peshkov

#7869 — GBAK can write uninitialized data into RDB\$RETURN_ARGUMENT and RDB\$ARGUMENT_POSITION fields

Implemented by Dmitry Kovalenko

Builds/Packaging

#8172 — File include/firebird/impl/iberror_c.h is missing in the Linux x64 tar archive

Implemented by Adriano dos Santos Fernandes

#8037 — Remove directory entries from debug symbols tarbal

Implemented by Alexander Peshkov

#8034 — (Re)set owner/group in tarbal of non-root builds

Implemented by Alexander Peshkov

Firebird 5.0 Release Candidate 2: Bug Fixes

Core Engine

#7904 — FB5 bad plan for query

Implemented by Dmitry Yemanov

#7903 — Unexpected results when using CASE-WHEN with LEFT JOIN

Implemented by Dmitry Yemanov

#7885 — Unstable error messages in services due to races related with service status vector

Implemented by Alexander Peshkov

[#7879](#) — Unexpected results when using natural right join

Implemented by Dmitry Yemanov

[#7867](#) — Error "wrong page type" during garbage collection on v4.0.4

Implemented by Ilya Eremin

[#7853](#) — Do not consider non-deterministic expressions as invariants in pre-filters

Implemented by Dmitry Yemanov

[#7839](#) — Potential bug in BETWEEN operator

Implemented by Vlad Khorsun

[#7831](#) — Incorrect type of UDF-argument with array

Implemented by Dmitry Kovalenko

[#7827](#) — Problem using Python firebird-driver with either Intel or M1 Mac builds with version 4.0.3 or 5.0+

Implemented by Adriano dos Santos Fernandes

[#7817](#) — Memory leak is possible for UDF array arguments

Implemented by Dmitry Yemanov

[#7795](#) — NOT IN <list> returns incorrect result if NULLs are present inside the value list

Implemented by Dmitry Yemanov

[#7772](#) — Blob corruption in FB4.0.3 (embedded)

Implemented by Vlad Khorsun

[#7767](#) — Slow drop trigger command execution under FB5.0

Implemented by Dmitry Yemanov

[#7760](#) — Parameters inside IN list may cause a string truncation error

Implemented by Dmitry Yemanov

[#7759](#) — Routine calling overhead increased by factor 6 vs Firebird 4.0.0

Implemented by Adriano dos Santos Fernandes

[#7461](#) — Differences in field metadata descriptions between Firebird 2.5 and Firebird 4

Implemented by Dmitry Yemanov

Server Crashes/Hangups

[#7917](#) — Hang in case of error when the sweep thread is attaching the database

Implemented by Alexander Peshkov

[#7905](#) — Segfault during TIP cache initialization

Implemented by Alexander Peshkov

[#7860](#) — Crash potentially caused by BETWEEN operator

Implemented by Vlad Khorsun

[#7832](#) — Firebird 5 and 6 crash on "... RETURNING * " without INTO in PSQL

Implemented by Adriano dos Santos Fernandes

[#7779](#) — Firebird 4.0.3 is constantly crashing with the same symptoms (fbclient.dll) (incl. DMP File Analysis)

Implemented by Vlad Khorsun

[#7762](#) — Crash on "Operating system call pthread_mutex_destroy failed. Error code 16" in log

Implemented by Alexander Peshkov

Utilities

isql

[#7844](#) — Removing first column with SET WIDTH crashes ISQL

Implemented by Adriano dos Santos Fernandes

[#7761](#) — Regression when displaying line number of errors in ISQL scripts

Implemented by Adriano dos Santos Fernandes

gbak

[#7851](#) — [FB1+, GBAK, Restore] The skip of att_functionarg_field_precision does not check RESTORE_format

Implemented by Dmitry Kovalenko

[#7846](#) — FB4 can't backup/restore int128-array

Implemented by Dmitry Kovalenko

[#7812](#) — Service backup does not work in multiple engines configuration

Implemented by Alexander Peshkov

[#7800](#) — Default publication status is not preserved after backup/restore

Implemented by Dmitry Yemanov

[#7770](#) — Restore takes 25% more time vs 4.0.0

Implemented by Vlad Khorsun

Firebird 5.0 Release Candidate 1: Bug Fixes

Core Engine

[#7747](#) — Fix an issue where the garbage collection in indexes and blobs is not performed in VIO_backout

Fixed by Ilya Eremin

[#7737](#) — Fix cases where the precedence relationship between a record page and a blob page is not set

Fixed by Ilya Eremin

[#7731](#) — Display length of `TIMESTAMP WITH TIMEZONE` is wrong in Dialect 1

Fixed by Alexander Peshkov

[#7730](#) — Server ignores the size of `VARCHAR` when performing `SET BIND ... TO VARCHAR(N)`

Fixed by Alexander Peshkov

[#7729](#) — `SET BIND OF TIMESTAMP WITH TIMEZONE TO VARCHAR(128)` uses the date format of Dialect 1

Fixed by Alexander Peshkov

[#7727](#) — Index for integer column cannot be used when `INT128/DECFLOAT` value is being searched

Fixed by Dmitry Yemanov

[#7723](#) — Wrong error message on login if the user doesn't exist and *WireCrypt* is disabled

Fixed by Alexander Peshkov

[#7713](#) — `FOR SELECT` statement can not see any changes made in D0 block

Fixed by Vlad Khorsun

[#7710](#) — Expression index — more than one null value cause attempt to store duplicate value error

Fixed by Vlad Khorsun

[#7703](#) — Requests leak in *AutoCacheRequest*

Fixed by Alexander Peshkov

[#7696](#) — SELECT from external procedure validates output parameters even when fetch method returns false

Fixed by Adriano dos Santos Fernandes

[#7694](#) — Fix false positives of “missing entries for record X” error during index validation when a deleted record version is committed and has a backversion

Fixed by Ilya Eremin

[#7691](#) — WITH CALLER PRIVILEGE has no effect in triggers

Fixed by Alexander Peshkov

[#7683](#) — RDB\$TIME_ZONE_UTIL.TRANSITIONS returns an infinite result set

Fixed by Adriano dos Santos Fernandes

[#7676](#) — Error “attempt to evaluate index expression recursively”

Fixed by Dmitry Yemanov

[#7670](#) — Cursor name can duplicate parameter and variable names in procedures and functions

Fixed by Adriano dos Santos Fernandes

[#7665](#) — Wrong result ordering in LEFT JOIN query

Fixed by Dmitry Yemanov

[#7664](#) — DROP TABLE executed for a table with big records may lead to “wrong page type” or “end of file” error

Fixed by Vlad Khorsun, Ilya Eremin

[#7662](#) — Fix performance issues in `prepare_update()`

Fixed by Ilya Eremin

[#7661](#) — Classic Server rejects new connections

Fixed by Vlad Khorsun

[#7649](#) — Switch Linux performance counter timer to `CLOCK_MONOTONIC_RAW`

Fixed by Adriano dos Santos Fernandes

[#7641](#) — Fix wrong profiler measurements due to overflow

Fixed by Adriano dos Santos Fernandes

[#7638](#) — `OVERRIDING USER VALUE` should be allowed for `GENERATED ALWAYS AS IDENTITY`

Fixed by Adriano dos Santos Fernandes

[#7627](#) — The size of a database with big records becomes bigger after backup/restore

Fixed by Ilya Eremin

[#7626](#) — Segfault when new attachment is done to shutting down database

Fixed by Alexander Peshkov

[#7611](#) — Can't backup/restore database from v3 to v4 with `SEC$USER_NAME` field longer than 10 characters

Fixed by Adriano dos Santos Fernandes

[#7610](#) — Uninitialized/random value assigned to `RDB$ROLES` → `RDB$SYSTEM PRIVILEGES` when restoring from FB3 backup

Fixed by Adriano dos Santos Fernandes

[#7604](#) — PSQL functions do not convert the output BLOB to the connection character set

Fixed by Adriano dos Santos Fernandes

[#7603](#) — BIN_SHR on INT128 does not apply sign extension

Fixed by Alexander Peshkov

[#7599](#) — Conversion of text with '\0' to DECFLOAT without errors

Fixed by Alexander Peshkov

[#7598](#) — DDL statements hang when the compiled statements cache is enabled

Fixed by Vlad Khorsun

[#7582](#) — Missing isc_info_end in *Firebird.pas*

Fixed by Alexander Peshkov

[#7574](#) — Derived table syntax allows dangling AS

Fixed by Adriano dos Santos Fernandes

[#7569](#) — Multi-level order by and offset/fetch ignored on parenthesized query expressions

Fixed by Adriano dos Santos Fernandes

[#7562](#) — Profiler elapsed times are incorrect in Windows

Fixed by Adriano dos Santos Fernandes

[#7556](#) — FB Classic can hang when attempts to attach DB while it is starting to encrypt/decrypt

Fixed by Alexander Peshkov

[#7555](#) — Invalid configuration for random fresh created database may be used after drop of another one with alias in *databases.conf*

Fixed by Alexander Peshkov

[#7553](#) — Firebird 5 profiler error with subselects

Fixed by Adriano dos Santos Fernandes

[#7548](#) — SET BIND OF TIMESTAMP WITH TIME ZONE TO CHAR is not working with UTF8 connection charset

Fixed by Adriano dos Santos Fernandes

[#7537](#) — Wrong name in error message when unknown namespace is passed into RDB\$SET_CONTEXT()

Fixed by Vlad Khorsun

[#7535](#) — High CPU usage connect to Firebird 3 database using Firebird 4 Classic and SuperClassic service

Fixed by Vlad Khorsun

[#7499](#) — Error during restore: “Index cannot be used in the specified plan”

Fixed by Vlad Khorsun

[#7488](#) — Invalid real to string cast

Fixed by Alexander Peshkov, Artyom Abakumov

[#7486](#) — No initialization of rpb’s runtime flags causes problems with SKIP LOCKED when config ReadConsistency = 0 and SuperServer

Fixed by Adriano dos Santos Fernandes

[#7484](#) — External engine SYSTEM not found

Fixed by Adriano dos Santos Fernandes

[#7480](#) — Firebird server stops accepting new connections after some time

Fixed by Alexander Peshkov

[#7456](#) — Impossible to drop function in package with name of PSQL-function

Fixed by Adriano dos Santos Fernandes

[#7387](#) — Unreliable replication behaviour in Linux Classic

Fixed by Dmitry Yemanov

[#7233](#) — Postfix for #5385 (CORE-5101): Fix slow database restore when Classic server mode is used

Fixed by Ilya Eremin

Server Crashes/Hangups

[#7738](#) — Crash on multiple connections/disconnections

Fixed by Alexander Peshkov

[#7658](#) — Segfault when closing database in valgrind-enabled build

Fixed by Alexander Peshkov

[#7554](#) — Firebird 5 partial index creation causes server hang up

Fixed by Vlad Khorsun

[#7514](#) — Segfault when detaching after deleting shadow on Classic

Fixed by Alexander Peshkov

[#7504](#) — Segfault when closing SQL statement in remote provider during shutdown

Fixed by Alexander Peshkov

[#7472](#) — Window functions may lead to crash interacting with others exceptions

Fixed by Adriano dos Santos Fernandes

[#7464](#) — Crash on repeating update in 5.0

Fixed by Adriano dos Santos Fernandes

Utilities

gbak

nbackup

[#7579](#) — Cannot *nbackup* a Firebird 3.0 database in Firebird 4.0 service with *engine12* setup in *Providers*

Fixed by Alexander Peshkov

Firebird 5.0 Beta 1 Release: Bug Fixes



This sections enumerates only bugfixes not already fixed in maintenance releases of earlier Firebird versions.

Core Engine

[#7422](#) — Seek in temporary blob level 0 makes read return wrong data

Fixed by Adriano dos Santos Fernandes

[#7388](#) — Different invariants optimization between views and CTEs

Fixed by Dmitry Yemanov

[#7304](#) — Events in system attachments (like garbage collector) are not traced

Fixed by Alex Peshkov

[#7227](#) — Dependencies of subroutines are not preserved after backup restore

Fixed by Adriano dos Santos Fernandes

#7220 — TYPE OF COLUMN dependency not tracked in package header and external routines

Fixed by Adriano dos Santos Fernandes

#7183 — Regression when derived table has column evaluated as result of subquery with IN, ANY or ALL predicate: “invalid BLR at offset ... / context already in use”

Fixed by Adriano dos Santos Fernandes

#7164 — Multi-way hash/merge joins are impossible for expression-based keys

Fixed by Dmitry Yemanov

#7133 — ORDER BY for big (>34 digits) *int128* values is broken when index on that field is used

Fixed by Alex Peshkov

#7077 — EXECUTE BLOCK (without RETURNS) do not work with batches

Fixed by Adriano dos Santos Fernandes

#7009 — IReplicatedTransaction receives wrong savepoint event

Fixed by Dmitry Sibiryakov, Dmitry Yemanov

#6942 — Incorrect singleton error with MERGE and RETURNING

Fixed by Adriano dos Santos Fernandes

#6869 — Domain CHECK-expression can be ignored when we DROP objects that are involved in it

Fixed by Adriano dos Santos Fernandes

#6807 — Regression: error “Unexpected end of command” with incorrect line/column info

Fixed by Adriano dos Santos Fernandes

[#5749](#) — “Token unknown” error on formfeed in query

Fixed by Adriano dos Santos Fernandes

[#3812](#) — Query with a stored procedure doesn’t accept explicit plan

Fixed by Dmitry Yemanov

[#3218](#) — Optimizer fails applying stream-local predicates before merging

Fixed by Dmitry Yemanov

Server Crashes/Hangups

[#7195](#) — Crash when accessing already cleared memory in the sorting module

Fixed by Andrey Kravchenko

Utilities

gbak

[#7436](#) — Backup error for wide table

Fixed by Alex Peshkov

Chapter 15. Firebird 5.0 Project Teams

Table 1. Firebird Development Teams

Developer	Country	Major Tasks
Dmitry Yemanov	Russian Federation	Full-time database engineer/implementor; core team leader
Alexander Peshkov	Russian Federation	Full-time security features coordinator; buildmaster; porting authority
Vladyslav Khorsun	Ukraine	Full-time DB engineer; SQL feature designer/implementor
Adriano dos Santos Fernandes	Brazil	International character-set handling; text and text BLOB enhancements; new DSQL features; code scrutineering
Roman Simakov	Russian Federation	Engine contributions
Dimitry Sibiryakov	Czech Republic	Engine and replication contributions
Ilya Eremin	Russian Federation	Engine contributions
Paul Beach	France	Release Manager; MacOS Builds
Pavel Cisar	Czech Republic	QA tools designer/coordinator; Firebird Butler coordinator; Python driver developer
Pavel Zotov	Russian Federation	QA tester and tools developer
Paul Reeves	France	Windows installers and builds
Mark Rotteveel	The Netherlands	Jaybird implementer and co-coordinator; Documentation writer
Jiri Cincura	Czech Republic	Developer and coordinator of .NET providers
Martin Koeditz	Germany	Developer and coordinator of PHP driver Documentation translator
Alexey Kovyazin	Russian Federation	Website coordinator
Helen Borrie	Australia	Release notes editor; Chief of Thought Police

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 Licence. Copies of the Licence are available at <https://www.firebirdsql.org/pdfmanual/pdl.pdf> (PDF) and <https://www.firebirdsql.org/manual/pdl.html> (HTML).

The Original Documentation is entitled *Firebird 5.0 Release Notes*.

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

Copyright © 2023-2024. All Rights Reserved. Initial Writer contact: dimitr at users dot sourceforge dot net.

Contributor(s): Mark Rotteveel.

Portions created by Mark Rotteveel are Copyright © 2023-2024. All Rights Reserved. (Contributor contact(s): mrotteveel at users dot sourceforge dot net).