



# Firebird 5

Firebird v5 features and  
improvements

# Fabio Codebue

P-soft



P-Soft

[www.p-soft.biz](http://www.p-soft.biz)



[www.firebirdsql.it](http://www.firebirdsql.it)



@fcodebue



[linkedin.com/in/fcodebue](https://linkedin.com/in/fcodebue)



@delphiforce



[f.codebue@p-soft.biz](mailto:f.codebue@p-soft.biz)



# AGENDA

- Roadmap
- Bugfixes
- New features



# Roadmap

# *Firebird 5*

- A various type of bug fixed...
- See tracker on github to details  
<https://github.com/FirebirdSQL/firebird/issues>
- See details in next slides



# *Firebird 6*

Postponed, to be finally implemented

- **TRUNCATE TABLE** command
- Stored optimizer statistics, histograms

Currently in development

- **Shared metadata cache** in SuperServer  
(implemented in HQbird)



# Firebird 6

To be ported from forks

- Support for JSON functions and JSON\_TABLE operator as per SQL
- Tablespaces
- fb\_repl\_print — textual printer for replication journals



HQbird: Advanced FirebirdSQL

# ***Firebird 6*** - Top-10 votes in the tracker

1. Job / Task Scheduler
2. Local temporary tables
3. Database Links
4. Add support for INTERSECT and EXCEPT data set operators
5. Create Table as Select ....
6. GIS implementation (opengis)
7. Full-Text indexing
8. Support native JSON datatype for columns as MySQL / PostgreSQL
9. Add support for SQL Schemas





# Bug fixes

# Bug fixes

Firebird version	Close	Open
5.0 beta 1	240	1
5.0 beta 2	37	3
<b>total</b>	<b>277</b>	<b>4</b>



**New features**



## ODS 13.1

ODS will be **13.1**

### Remember

backup with old gbak version

Restore with new gbak version

# *Parallel (multi-threaded) operations*

- Parallel execution is supported for both auto- and manual sweep
- decreasing the total operation time
- By default, parallel execution is not enabled.

# *Parallel (multi-threaded) operations*

There are two ways to enable parallelism in user attachment:

- set number of parallel workers in DPB using new tag *isc\_dpb\_parallel\_workers*,
- set default number of parallel workers using new setting *ParallelWorkers* in *firebird.conf*.

# ***Parallel (multi-threaded) operations***

- New firebird.conf setting (*Default value is 1* )  
**ParallelWorkers** - set default number of parallel workers that used by user attachments.
  - Could be overridden by attachment using tag ***isc\_dpb\_parallel\_workers*** in DPB
- **MaxParallelWorkers** - limit number of simultaneously used workers for the given database and Firebird process.

# ***Parallel (multi-threaded) operations***

## **SWEEP**

- For gfix utility there is **new command-line switch** **-parallel** that allows to set number of parallel workers for the sweep task.

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

will run sweep on given database and ask engine to use 4 workers. gfix uses DPB tag *isc\_dpb\_parallel\_workers* when attaches to <database>

# *Parallel (multi-threaded) operations*

## **GBAK**

- 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.

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

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

# *Support for partial indices*

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

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

# *Support for partial indices*

## Examples:

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

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

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

# *Support for partial indices*

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

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

# *PSQL and SQL profiler*

## **RDB\$PROFILER package**

- allows users to **measure performance cost** of SQL and PSQL
- a **system package** in the engine passing data to a profiler plugin

collecting statistics of

- how many times each line was executed along with its minimum, maximum and accumulated execution times (with ns precision)
- open and fetch statistics of implicit and explicit SQL

# *PSQL and SQL profiler*

## COLLECT DATA

- a user must first start a profile session with  
`RDB$PROFILER.START_SESSION`
- 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).

# *SKIP LOCKED* clause

- It makes the engine **skip records locked by other transactions** instead of wait on them or raise conflict errors

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

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

```
UPDATE <sometable>  
SET ...  
[WHERE ...]  
[PLAN ...]  
[ORDER BY ...]  
[ROWS ...]  
[SKIP LOCKED]  
[RETURNING ...]
```

# *Optimize the record-level RLE algorithm*

- A **denser compression** of shorter-than-declared strings and sets of subsequent NULL
- Starting with ODS 13.1
- **reducing the storage overhead**
- This **improves compression for long VARCHAR fields** (especially UTF8 encoded) that are filled only partially

# ***Inline minor ODS upgrade***

- allows to **upgrade the existing database to the newest ODS version without backup/restore**
- **belongs to the same major ODS version**
- **must be done manually**
- **It requires exclusive access** to the database or error is thrown
- **USE\_GFIX\_UTILITY system privilege** is required
- **Upgrade is transactional**, all changes are reverted if any error

# *Inline minor ODS upgrade*

## **gfix command**

a user must first start a profile session with

```
gfix -up[grade] <database>  
      -user <username> -pass <password>
```

# *Support multiple rows for DML RETURNING*

- Currently, DML RETURNING only supports singleton inserts/updates/deletes
- This restriction be relaxed so that the following statements can return multiple rows:

INSERT ... SELECT ...

Searched UPDATE

Searched DELETE

MERGE

UPDATE OR INSERT

# ***Avoid data retrieval if the WHERE ...***

**Avoid data retrieval if the WHERE clause always evaluates to FALSE**

In query like this

```
SELECT * FROM SOME_TABLE WHERE 1 = 0
```

It is obviously that condition in WHERE clause is always FALSE, but FB will read all records from SOME\_TABLE without any benefits

Imagine into a recursive CTE (Common Table Expression)

# *BLOB\_APPEND*

- Regular operator || with BLOB arguments creates temporary BLOB per every pair of args with BLOB.
- This could lead to the excessive memory consumption and growth of database file.
- The **BLOB\_APPEND** function is designed to concatenate BLOBs without creating intermediate BLOBs.
- Available in: DSQL, PSQL.

```
BLOB_APPEND(<blob> [, <value1>, ... <valueN>]
```

# *BLOB.NEW*

## **Function NEW**

RDB\$BLOB\_UTIL.NEW is used to create a new BLOB. It returns a handle (an integer bound to the transaction) that should be used with the others functions of the package.

Input

SEGMENTED type BOOLEAN NOT NULL

TEMP\_STORAGE type BOOLEAN NOT NULL

Return type: INTEGER NOT NULL

# *OPEN\_BLOB*

**RDB\$BLOB\_UTIL.OPEN\_BLOB** is used to open an existing BLOB for read. It returns a handle that should be used with the others functions of the package.

Input parameter:

BLOB type BLOB NOT NULL

Return type: INTEGER NOT NULL.

# *BLOB\_APPEND*

**RDB\$BLOB\_UTIL.APPEND** is used to append chunks of data to a BLOB handle created with **RDB\$BLOB\_UTIL.NEW**.

Input parameters:

HANDLE type INTEGER NOT NULL

DATA type VARBINARY(32767) NOT NULL

# *BLOB\_READ*

**RDB\$BLOB\_UTIL.READ**, When is fully read 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(32767) .

# *BLOB\_SEEK*

**RDB\$BLOB\_UTIL.SEEK** is used to set the position for the next READ. 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 negative.

Input parameter:

HANDLE type INTEGER NOT NULL

MODE type INTEGER NOT NULL

OFFSET type INTEGER NOT NULL

Return type: INTEGER NOT NULL.

# *BLOB CANCEL*

**RDB\$BLOB\_UTIL.CANCEL** is used to release a BLOB handle opened with RDB\$BLOB\_UTIL.OPEN\_BLOB or discard one created with RDB\$BLOB\_UTIL.NEW.

Input parameter:

HANDLE type INTEGER NOT NULL

# *MAKE\_BLOB*

**RDB\$BLOB\_UTIL.MAKE\_BLOB** is used to create a BLOB from a BLOB handle created with NEW followed by its content added with APPEND. After MAKE\_BLOB is called the handle is destroyed and should not be used with the others functions.

Input parameter:

HANDLE type INTEGER NOT NULL

Return type: BLOB NOT NULL.

# Compiled statement cache

- The engine 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).

# UNICODE\_CHAR

Returns the UNICODE character with the specified code point.

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

Notes: 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.

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

Notes: Returns 0 if the string is empty.

Example:

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

# *Replication Configuration*

## **cascade\_replication**

New parameter that 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*

# *OTHERS....*

- Improve performance of STARTING WITH with insensitive collations
- SIMILAR TO should use index when pattern starts with non-wildcard character (as LIKE does)
- Cost-based choice between hash/merge joins

# *OTHERS....*

- Support for WHEN NOT MATCHED BY SOURCE for MERGE statement
- Network support for bi-directional cursors
- Allow sub-routines to access variables/parameters defined at the outer/parent level
- WireCryptPlugin, RemotePipeName, TcpLoopbackFastPath



**THANK YOU**

