FBOpt - The Database Optimiser

Version 3.0

IBPhoenix
THE POWER WITHIN

- Introduction
- Requirements and Installation
- Running FBOpt
- Modes of Work
- Scanning database content for later analysis
- Creating optimized copy of the database
- Creating a simple copy of the database
- Data pump
- Database structure upgrade and downgrade
- Converting database to new character set
- Parallel operations

Introduction

IBPhoenix FbOpt is a command-line tool for optimising Firebird databases. The tool addresses some aspects of the internal structure of databases that the Firebird engine itself cannot rectify, and some issues that survive even a backup and restore with gbak.

Beside optimization, FBOpt can be also used to **upgrade** or **downgrade** databases between Firebird versions (ODS structures), convert databases to new default character set, or to create regular `gbak` database backups with **simultaneous** restore.

FBOpt can run on 64-bit versions of Windows and Linux, and requires at least the client part of 64-bit version of Firebird 3 or newer to be installed on machine where you want to run it.

Internal Database Structure

Efficiency of the internal database structure plays an important role in the performance of the Firebird engine. The layout of internal structures has a direct impact on the complexity and speed of operations needed for processing them, which translates to utilisation of CPU and memory. Data distribution has a primary impact on the number of I/O operations and on cache utilisation. While the layout of data structures is fixed in the code, data distribution is variable, depending on the actual operations performed on the database. In current Firebird implementations, the engine pays only minimal attention to the storage and distribution of both user and internal data. The engine's primary focus is on the speed of the current operation, rather than the overall balanced performance. Over time this can lead to degradation of performance as the internal database layout becomes progressively less efficient.

Eventually the drop in performance requires intervention which, for Firebird, means a rebuild of the database from a logical backup (gbak). However, restore from backup fixes **only the following problems**, namely:

- **Row fragmentation**, so each row is stored on single page (if a single row is not bigger than database page)
- **Table and index fragmentation**, data and index pages are compacted to hold as much data as possible, and all pages that belong to single database table or index are stored consecutively.

It does **not** address the following issues:

- **Index clustering:** jumps to different data pages while walking through an index
- **Data density on data pages:** all data pages could be either highly packed or have space reserved for updates
- **Physical location of user data and index structures:** all user data is stored first, in creation order, then all the indices are added
- **Prevention of data fragmentation**

FBOpt was designed and built to solve all these problems.

Physical Order of Rows

Specifying the order of rows can significantly improve performance for queries that read data in a range of key values or in a given key order.

If there is an index with the same key, it has the highest possible “clustering factor” and there are no unnecessary jumps to different data pages while walking through the index. Any potential loss of performance from using such an index for returning data in key order, rather than internal sort order, is eliminated.

With FBOpt, you can specify one such index for each table.

Space Reservation

Specifying the space reservation level for individual tables allows the option to mark tables that do not change for higher storage density while keeping necessary space reserved for tables that are frequently updated. High data density saves disk space and increases query performance by reducing the number of I/O operations.

Physical Order of Tables and Clustering

Specifying the physical order of tables in a database allows storage of tables with static data before tables that are updated or extended, so the volatile part of the database is at the end of the file. Combining this with table clustering makes it possible to keep static data more or less separated from volatile data which, over time, helps reduce the build-up of fragmentation.

Database analysis

Since version 2.0, FBOpt can determine the best clustering and physical ordering of tables, and which tables could use dense storage without reserving space.

To do so, FBOpt analyzes the database and collects information about transactions that saved individual table rows. By comparing two such information sets taken at different time, FBOpt can determine data change patterns for each table, and assign it to appropriate data cluster.

Database optimization

FBOpt is basically a database copy utility that perform specific optimizations while creating the new database:

- row defragmentation so that each row is stored on one page (if one row is not larger than a database page)
- defragmentation of tables and indexes, so that data and index pages are compressed to contain as much data as possible and all pages that belong to one database table or index are stored one by one.
- in selected cases, further compression of data pages is performed by not reserving space for additional versions of rows on each page.
- optional physical sorting of rows in individual tables (for a better clustering factor for the most important index in a given table).
- setting the physical order of the tables in the database in a way that reduces the possibility and extent of data fragmentation due to subsequent changes in the database. This physical layout is divided into multiple blocks stored sequentially.

The optimized database structure is as follows:

- The **first block** contains tables whose data does not change. Table data is placed first, followed by any indexes defined for those tables. Table data is written with the “no reserve” flag set, for higher data density.
- The **second block** contains tables that have been shortened (only deleting rows from the end). Data and indexes are stored in the same way as in the first block.
- The **third block** contains only the data of the tables, the contents of which are only expanded. Table data is written with the “no reserve” flag set.
- The **fourth block** contains only the data of the tables whose contents are changing (via update / delete). Data is written with the “reserve space” flag set.
- The **fifth block** contains only the data of the tables that are new (without the information in the base data file used for the analysis). Data is written with the “reserve space” flag set.
- Finally, the remaining indexes (not created within block 1 and 2 storage) are created.

Requirements

FBOpt can optimize only databases created by Firebird 3 or newver. It could be used on 64-bit versions of Windows and Linux, and requires at least the client part of 64-bit version of Firebird to be installed on machine where you want to run it.

Installation

FBOpt is distributed as a single executable file, that you can place it into any directory that suits your needs. A license file **`fbopt.license`** must be placed in the same directory.

The license file should be automatically sent to the e-mail address you entered on purchase (for FREE version, a 14-day trial license is sent). If you did not received your license, please contact us at ***`support@ibphoenix.com`***.

Running FBOpt

The CLI has next basic schema:

```
fbopt [options] source-db [new-db]
```

Where:

- `source-db` is a database from which an optimized copy or snapshot should be created.
- `new-db` is the newly created optimized copy.

Both `source-db` and `new-db` could be either local or remote databases. However, for best performance it's recommended to use local database specifications (FBOpt uses embedded Firebird server to work with local databases).

Command line options

-A(NALYZE) <file>
Read saved database statistics from the given file

-B(ACKUP) <file>
Store backup file locally as well

-BU(FFERS) <number>
Override page buffers default

-CRYPT <name>
Crypt plugin name for new database

-DATA(_ONLY)
Copy only data into already existing target database

-FE(TCH_PASSWORD) <file>
Fetch user password from text file

-H(ELP), -?
Display help text

-I(NACTIVE)
Do not activate indexes in created database

-KEYHOLDER <name>
Key holder plugin name

-KEY(NAME) <name>
Name of encryption key

-M(ETADATA)

Use database comments to adjust optimization

-N(O_VALIDITY)
Do not activate data validation constraints in created database

-NOD(BTRIGGERS)
Do not run database triggers

-NOOP(TIMIZE)
Perform backup-restore only

-P(AGE_SIZE) <number>
Set page size for new database

-PAR(ALLEL) <N>
Perform operations using N threads

-PAS(SWORD) <password>
User password

-Q
Be quiet. More ‘Q’s - more quiet

-REPLICA <mode>
“none”, “read_only” or “read_write” replica mode for created database

-RO(LE) <name>
User SQL role

-S(NAPSHOT) <file>
Save database statistics to the given file

-U(SER) <name>
Firebird user name

-V

Be verbose. More 'V's - more verbose

-Z

Print version number and licensing information

Modes of Work

The development of FBOpt went through several phases. In the first version, FBOpt used manually created specification files to drive the optimization process. The second version added functionality to analyze the database content at different points in time, to detect data change patterns, which were then used to generate parameter sets for subsequent optimization. The current third version further simplifies the optimization process by bypassing the need to use parameter sets and also adds brand new features that go beyond database optimization.

The FBOpt functionality could then be divided into several areas:

- Creating optimized copy of the database
- Scanning database content for later comparative analysis
- Creating a simple copy of the database

The "simple copy" mode could be used for several tasks:

- Upgrading or downgrading the database internal structure (ODS)
- Converting database content to different character set
- Creating logical database backup with simultaneous restore
- Executing direct logical backup and restore in single step without intermediate backup file

The following sections describe the individual work modes.

Scanning database content for later comparative analysis

Activation:

```
fbopt [options] -SNAPSHOT file source-db
```

Enabled options:

```
-CRYPT <name>
-KEYHOLDER <name>
-KEY(NAME) <name>
-NOD(BTRIGGERS)
-PAR(ALLEL) <N>
-U(SER) <name>
-PAS(SWORD) <password>
-FE(TCH_PASSWORD) <file>
-Q
-V
```

In this mode, FBOpt simply connects to the specified source database, reads information about transactions that created individual data rows, and saves it to the `file` specified by the `-SNAPSHOT` parameter. You should keep it at safe place for future use with FBOpt.

The speed of snapshot creation depends on many factors, but it's typically much faster than backup with `gbak`. By default, FBOpt use `6` worker threads to gather required information in parallel. You can adjust number of parallel workers with `-PAR(ALLEL)` option.

Creating optimized copy of the database

Activation:

```
fbopt [options] [-METADATA] [-ANALYZE file] source-db new-db
```

Enabled options:

```
-REP(LACE_DATABASE)
-NOD(BTRIGGERS)
-BU(FFERS) <number>
-I(NACTIVE)
-N(O_VALIDITY)
-P(AGE_SIZE) <number>
-REPLICA <mode>
-CRYPT <name>
-KEYHOLDER <name>
-KEY(NAME) <name>
-U(SER) <name>
-PAS(SWORD) <password>
-FE(TCH_PASSWORD) <file>
-RO(LE) <name>
-PAR(ALLEL) <N>
-Q and -V
-SNAPSHOT <file>
-DATA(_ONLY)
```

The parameters for optimization process could be provided in several ways:

1. If classification of tables and indices is stored in database metadata, use the `METADATA` option.

2. If classification of tables should be determined by comparative analysis of old and current data creation snapshots, use the `-ANALYZE` option to specify the base snapshot.

3. In all other cases, classification of tables will be determined by quick heuristic algorithm.

When `-METADATA` option is specified, FBOpt looks for specific `regex` patterns in COMMENT content of each table:

- contains regex ".*(FBOPT STATIC).*" - table is stored in segment 1

- contains regex ".*(FBOPT EXTENDED).*" - table is stored in segment 3

- contains regex ".*(FBOPT VOLATILE).*" - table is stored in segment 4

If COMMENT for any INDEX contains regex ".*(FBOPT).*", the table data are stored as sorted in the index key order. If more than one index that belongs to the same table is marked like this, the first one found is used.

It's possible to optimize data into already existing database using `-DATA(_ONLY)` option. However, this database must be able to accept ALL data from source database.

If the `-SNAPSHOT` switch is specified, then analytic information snapshot is created for newly created database copy.

Creating a simple copy of the database

Activation:

```
fbopt [options] [-B(ACKUP) <file>] -NOOPTIMIZE source-db new-db
```

Enabled options:

```
-REP(LACE_DATABASE)
-NOD(BTRIGGERS)
-BU(FFERS) <number>
-I(NACTIVE)
-N(O_VALIDITY)
-P(AGE_SIZE) <number>
-REPLICA <mode>
-CRYPT <name>
-KEYHOLDER <name>
-KEY(NAME) <name>
-U(SER) <name>
-PAS(SWORD) <password>
-FE(TCH_PASSWORD) <file>
-RO(LE) <name>
-PAR(ALLEL) <N>
-Q and -V
-SNAPSHOT <file>
-DATA(_ONLY)
```

In this mode, FBOpt basically pipelines `gbak` backup to restore. The backup file is not created, unless `-BACKUP` switch is used. This FBOpt mode could be used either as:

1. more convenient and effective replacement for usual `gbak` backup+restore cycle which does not require disk space for intermediate backup file.

2. regular `gbak` backup with simultaneous restore test (the created database could be deleted after successful completion, while backup file is kept).

Data pump

With `-DATA_ONLY` switch, FBOpt could be used as simple and fast data pump that moves ALL data from source database to existing (empty) database. The target database does not need to have exactly the same structure as the source database, but must be able to accept all data from source without error. This for example means that all constraint indices must be deactivated.

The `-DATA_ONLY` switch could be used with both, optimized or simple database copy operations. However, due to technical constraints, the simple database copy (`-NOOPTIMIZE`) in this case does not use `gbak` to transfer data, but method normally used to create the optimized copy (it just places all tables and indices in segment 5). Hence switches `-BACKUP` and `-DATA_ONLY` are mutually exclusive.

Database structure upgrade and downgrade

There are several simple methods that could be used to easily upgrade databases to new ODS structure with FBOpt.

1. You can use remote server for either source or target database (or both), where server for the target database has the desired higher version.

2. You can install older engine library (for example `Engine12.so/dll` from Firebird v3) into `plugins` directory of newer Firebird installation, and add its name before newer engine in `providers` option in `firebird.conf`. Then you can use the local server connections, and the newly created database will have upgraded ODS structure.

3. In some cases, when newer Firebird version also supports access to older databases (like Firebird v5 can access databases created by Firebird v4), the newly created database will have new ODS automatically.

In all cases, you can use both, simple or optimized database copy methods.

You can use the same methods also to downgrade the database to older ODS versions, if you'll first create an empty database with older ODS, and use the `-DATA_ONLY` switch.

Converting database to new default character set

Using empty database and `-DATA_ONLY` switch could be used not only to downgrade database ODS, but also to convert data to new default character set. All you have to do is changethe default character set in the new database and the data will be automatically converted during the transfer.

Parallel operations

By default, FBOpt uses `6` parallel worker threads to speed up database analysis, copying of data and activation of indices. You can adjust the number of worker threads with `–PARALLEL` option. These parallel workers differ from parallel workers introduced in Firebird v5.

When FBOpt is used with Firebird v5 or newer, parallel workers provided by Firebird are used automatically for simple database copy operations without `–DATA_ONLY` switch, and if paraller workers are properly configured in `firebird.conf`.

IBPhoenix

THE POWER WITHIN

Support & tools for Firebird

IBPhoenix is the leading provider of information, tools and services for Firebird users and developers

IBPhoenix website & e-shop