Practical Migration Guide to Firebird 5.0

Denis Simonov

Version 1.0 from 23.02.2024

This material was created with the support and sponsorship of the company IBSurgeon, developer of HQbird, advanced distribution of Firebird for enterprises, and Mass Migration Framework for Firebird, toolset to perform migration of many (100+) servers during several days.

Contents

Introduction

This article is intended primarily for those who plan to upgrade the Firebird DBMS to version 5.0 in the near future. Many administrators still use Firebird 2.5, but plan to upgrade to version 5.0. That is why the process of migration from Firebird versions 2.5, 3.0 and 4.0 is described here.

Chapter 1. Manual installation of Firebird 5.0 in Windows

The process of installation from a zip archive is described below. Even if you install Firebird from a special installation package, you may still need to change some settings after installation. In addition, manual installation allows you to install multiple versions of Firebird on one machine.

Download the archive of the corresponding bitness and unpack it to the directory where the Firebird server will be located.

Next, you need to create the SYSDBA user. For administrators who are migrating from Firebird 3.0 or 4.0, this operation is not new. In Firebird 2.5 and earlier, after installation, the SYSDBA user always existed and had the default password *masterke*, which had to be changed immediately.

1.1. Initialization of SYSDBA

Starting from Firebird 3.0, the SYSDBA user is not initialized by default (for the SRP user management plugin), so you need to explicitly create a user and specify a password for him. This can be done in two ways: using the console tool for executing interactive queries isql.exe and the console tool for managing the security database gsec.exe.

Note

Depending on the location of Firebird, these utilities may require running with administrator privileges.

1.1.1. Initialization of SYSDBA using ISQL

Run the tool for executing interactive queries isql.exe. Connect to the security database in embedded server mode, specifying the user SYSDBA without a password. The user SYSDBA does not exist in the security database yet, but in embedded mode the user and his password are not checked, and Firebird trusts any specified user name. Execute the SQL query to create the user SYSDBA:

```
CREATE USER SYSDBA PASSWORD '<password>';
```

The user SYSDBA is initialized, you can exit the interactive mode.

Example 1. Initialization of SYSDBA via ISQL

```
c:\Firebird\5.0>isql security.db -user SYSDBA

Database: security.db, User: SYSDBA

SQL> CREATE USER SYSDBA PASSWORD 'm8ku234pp';
```

```
SQL> exit;
```

1.1.2. Initialization of SYSDBA using GSEC

Run `gsec.exe`, specifying the user `SYSDBA` and the database `security.db`. Execute the command to add the user SYSDBA:

```
add SYSDBA -pw <password>
```

Example 2. Initialization of SYSDBA via GSEC

```
c:\Firebird\5.0>gsec -user SYSDBA -database security.db

gsec is deprecated, will be removed soon

GSEC> add SYSDBA -pw m8ku234pp
GSEC> quit
```

Warning

The tool `gsec.exe` is obsolete, many features available through SQL are not available in it.

1.2. Configuration

Before installing and running Firebird as a service, you need to choose the server mode.

1.2.1. Server mode

By default, Firebird will start in `SuperServer` mode. If you want the server to run in another architecture, you need to change the value of the `ServerMode` parameter in `firebird.conf`. Uncomment it (remove the hash) and set the desired mode: `Super`, `SuperClassic` or `Classic`. For example, to set Classic:

```
ServerMode = Classic
```

1.2.2. Authorization from Firebird 2.5 client libraries

In Firebird 5.0, secure password authentication (SRP) is used by default. Clients of Firebird 2.5 and earlier versions used traditional authentication (`Legacy_Auth`), which is disabled in Firebird 5.0 by default, as it is not secure.

To support traditional authentication, you need to change the following parameters AuthServer,

UserManager and WireCrypt.

Example 3. Enabling authorization with previous versions of Firebird client

```
AuthServer = Srp256, Srp, Legacy_Auth
UserManager = Srp, Legacy_UserManager
WireCrypt = Enabled
```

After the above manipulations, we will have two active user managers, by default the first one in the UserManager list is active.

Important

Users with the same name in different user managers are different users and they may have different passwords. This applies to both SYSDBA and the database owner.

If you do not need support for secure password authentication (SRP), then remove the Srp256 and Srp plugins from AuthServer; Srp from UserManager, and you can change WireCrypt to Disabled.

We have already created SYSDBA in the SRP user manager. In Legacy_UserManager, SYSDBA already exists, with the standard password *masterkey*, which needs to be changed. Let's do this using the isql tool. In the ALTER USER statement, you must specify the Legacy_UserManager user manager.

Example 4. Changing the password of SYSDBA in Legacy_UserManager

```
c:\Firebird\5.0>isql security.db -user SYSDBA

Database: security.db, User: SYSDBA

SQL> ALTER USER SYSDBA SET PASSWORD 'er34gfde' USING PLUGIN Legacy_UserManager;
SQL> exit;
```

1.2.3. Setting the default connection time zone

Starting from Firebird 4.0, new date and time types with time zone support are available.

Even if you are not going to use types with time zones in the near future, you need to consider that the expressions CURRENT_TIMESTAMP and CURRENT_TIME now return data types with time zones. There is a compatibility mode that allows you to convert types with time zones to types without time zones. However, such a conversion may work incorrectly if the connection time zone is set incorrectly.

Usually the session time zone is set on the client side. If the time zone on the client side is not set, then the time zone of the operating system is used by default. You can also set the default session time zone using the configuration parameter DefaultTimeZone.

```
DefaultTimeZone = Europe/Moscow
```

1.2.4. Running multiple instances of Firebird simultaneously

Here it is assumed that you want to run instances of different versions of Firebird, each of which is installed in its own directory.

To run multiple instances of Firebird simultaneously, you need to separate them by different tcp ports (if, of course, the listener is running in TCP/IP listening mode). To do this, you need to change the RemoteServicePort parameter in firebird.conf.

For example, if you already have one server that listens on port 3050, then you need to set any other free port, for example 3051. In this case, you will need to specify the new port in the connection string (except when the application and Firebird client have access to firebird.conf with the changed default port number).

```
RemoteServicePort = 3051
```

You also need to set unique values for the IpcName parameter for each instance of the DBMS server. This will avoid error messages in `firebird.log`. These errors are not critical if you do not use the XNET protocol. However, if it is used, then you should keep in mind that this parameter will have to be changed on the client side through DPB.

1.2.5. How to create firebird.conf for better performance?

To optimize the performance of your VM/hardware, consider generating firebird.conf and databases.conf for version 5 with the Configuration Calculator for Firebird, a free tool provided by IBSurgeon. These configurations will outperform the default settings. Just remember, you'll need to manually implement the suggested changes from sections above into the configuration produced by the Calculator.

1.3. Installing and running Firebird as a service

The `instsvc.exe` utility writes, deletes or changes information about the server startup in the service base of the operating system. In addition, it allows you to manage the startup and shutdown of the service.

If you run it without parameters, it will display help on commands and parameters.

```
instsvc
Usage:
instsvc i[nstall]
[ -a[uto]* | -d[emand] ]
[ -g[uardian] ]
[ -l[ogin] username [password] ]
[ -n[ame] instance ]
```

```
[ -i[nteractive] ]

sta[rt]   [ -b[oostpriority] ]
[ -n[ame] instance ]
sto[p]    [ -n[ame] instance ]
q[uery]
r[emove]  [ -n[ame] instance ]

'*' denotes the default values
'-z' can be used with any other option, prints version
'username' refers by default to a local account on this machine.
Use the format 'domain\username' or 'server\username' if appropriate.
```

Important

The `instsvc` utility must be run in a console with administrative privileges (run the console as administrator).

To install the service, you need to enter the command

```
instsvc install
```

In this case, Firebird will be installed as a service with the name "Firebird Server – DefaultInstance". This service will start automatically when the OS starts, under the LocalSystem account, intended for services.

If you need to install multiple instances of Firebird running as services, then you need to assign them different names using the `-n` option

```
instsvc install -n fb50
```

To start the service, use the command

```
instsvc start
```

If the service was installed with a name different from the default, then you need to use the `-n` switch

```
instsvc start -n fb50
```

To stop the service, use the command

```
instsvc stop
```

If the service was installed with a name different from the default, then you need to use the `-n` switch

```
instsvc stop -n fb50
```

To remove the service, you need to enter the command

```
instsvc remove
```

If the service was installed with a name different from the default, then you need to use the `-n` switch

```
instsvc remove -n fb50
```

To view all Firebird services installed in the system, use the command

```
instsvc query
```

```
Firebird Server - fb30 IS installed.
Status  : running
Path    : C:\Firebird\3.0\firebird.exe -s fb30
Startup : automatic
Run as  : LocalSystem

Firebird Server - fb40 IS installed.
Status  : running
Path    : C:\Firebird\4.0\firebird.exe -s fb40
Startup : automatic
Run as  : LocalSystem

Firebird Server - fb50 IS installed.
Status  : running
Path    : C:\Firebird\5.0\firebird.exe -s fb50
Startup : automatic
Run as  : LocalSystem
```

1.3.1. Using install_service.bat and uninstall_service.bat

To simplify the procedure of installing and uninstalling services in the ZIP archive, two BAT files are supplied with Firebird: `install_service.bat` and `uninstall_service.bat`.

In this case, the procedure of installing Firebird as a service looks like this

```
install_service.bat
```

In this case, the procedure of removing the Firebird service looks like this

```
uninstall_service.bat
```

If you need to assign a different name to the service, then specify this name as an argument

```
install_service.bat fb50
```

If the service was installed with a name different from the default, then specify this name as an argument

```
uninstall_service.bat fb50
```

1.4. Installing the client

If you are talking about installing only the client part, then the file fbclient.dll is required. The Firebird 5.0 client requires the installed Microsoft Runtime C++ 2015-2022 of the corresponding bitness. If this library is not installed, you can copy additional libraries that are supplied in the ZIP archive under Windows `msvcp140.dll` and `vcruntime140.dll` (and `vcruntime140_1.dll` for 64-bit installation).

It is desirable that the file firebird.msg be located next to `fbclient.dll`. Most error messages are already contained in `fbclient.dll`, but if you are going to use console utilities, the file firebird.msg must be present.

Unlike Firebird 2.5 and Firebird 3.0, the client library also requires ICU files (`icudt63.dll`, `icuin63.dll`, `icuuc63.dll` and `icudt63l.dat`). Previously, the ICU library was required only by the server. Now it may be required by the client part, if you are going to work with data types `TIMESTAMP WITH TIME ZONE` and `TIME WITH TIME ZONE`. The ICU library is also required when calling the functions `UtilInterface::decodeTimeTz()` and `UtilInterface::decodeTimestampTz()`.

Note

In Windows 10, the ICU library supplied with the operating system can be used.

If you need traffic compression when working over TCP/IP, then you will need the library `zlib1.dll`.

You may need the library `plugins/chacha.dll` if you are going to use the ChaCha traffic encryption plugin. This plugin is used by default starting from Firebird 4.0, as it is the first in the list of values in the configuration parameter `WireCryptPlugin = ChaCha, Arc4`.

Note on loading plugins

`fbclient.dll` version 3.0 did not load plugins from dynamic libraries from the

plugins directory by default. `fbclient.dll` version 4.0 and higher uses `plugins/chacha.dll` by default, if this plugin is present. Missing plugins are ignored.

However, there is an important feature. `fbclient.dll` looks for the file `firebird.conf` in its directory, and if it is missing, it tries to find it in the directory above. The directory where `firebird.conf` is found is the root directory - from which all other known relative paths (`plugins`, `intl`) are counted.

This behavior can play a cruel joke on you. The thing is that the 64-bit installer places the 32-bit library fbclient.dll in the folder `$(fbroot)/WOW64`. If you want to use the library from this directory, you may get the following error message

```
Error loading plugin ChaCha.
Module C:\Firebird\5.0\plugins/ChaCha exists but can not be loaded.
unknown Win32 error 193.
```

In this case, the 32-bit `fbclient.dll` tried to load the 64-bit ChaCha plugin.

To fix this error, just put the file firebird.conf in the folder $(fbroot)/WOW64.

The library `fbclient.dll`, as well as other files of the client library, must be located either next to the application, or in one of the directories where the search is performed, for example added to `PATH` or the system directory for placing public libraries (`system32` or `SysWOW64`).

Important

Placing the client library in `PATH` may interfere with other applications that require a client library of a different version or a different server. Therefore, if it is assumed that the application should work independently of other applications with a specific version of the client, then the client files should be placed in the application folder, and not add this path to `PATH`.

1.4.1. Using instclient

To deploy the Firebird client library in the Windows system directory, use the command

```
instclient install fbclient
```

Important

The `instclient` utility does not copy any files to the system directory except `fbclient.dll`.

1.5. Installing the embedded version

Starting from version Firebird 3.0, the embedded version is not distributed separately. You can use the same set of files as both a network server and an embedded server. But, if you need an

embedded set of minimal size, then the structure of files and directories for Firebird 5.0 embedded is as follows:

- intl
  - fbintl.conf
  - fbintl.dll
- plugins
  - engine13.dll
- firebird.conf
- icudt63l.dat
- fbclient.dll
- ib_util.dll
- icudt63.dll
- icuin63.dll
- icuuc63.dll
- msvcp140.dll
- vcruntime140.dll
- vcruntime140_1.dll
- firebird.msg

If necessary, you can also copy the executable files of the utilities `fbsvcmgr.exe`, `fbtracemgr.exe`, `gbak.exe`, `gfix.exe`, `gstat.exe`, `isql.exe`, `nbackup.exe`. If you are going to use `gbak` with the `-zip` switch, then you will also need the library `zlib1.dll`.

Note

For those who are migrating from Firebird 2.5, you should consider 2 points:

- Instead of a single library `fbembed.dll`, several files are required, and the file `fbclient.dll` cannot be renamed. Access components must use the library `fbclient.dll` as the entry point.
- In the configuration file `firebird.conf`, you should change the value of the parameter `ServerMode` to `SuperClassic` or `Classic` so that on one computer you can connect to the same database from different applications using embedded (behavior of Firebird 2.5 embedded by default).

Chapter 2. Converting the database to the new format

Firebird 5.0 databases have ODS (On-Disk Structure) 13.1. To make Firebird 5.0 work with your database, you need to bring it to the native ODS. Usually this is done using the `gbak` tool. However, do not rush to make a backup of your database and restore it with the new ODS - first you need to eliminate possible compatibility issues.

2.1. List of incompatibilities at the SQL language level

SQL language compatibility issues are possible both for the objects of the database itself (PSQL procedures and functions), and for the DSQL queries used in your application.

To detect SQL language compatibility issues for database objects, the following method is recommended. Extract the metadata of the database into a script on the old version of Firebird.

```
isql <database> -x -o create_script.sql
```

Uncomment the CREATE DATABASE statement inside the script, make the necessary changes, and try to create a new database from the script in Firebird 5.0:

```
isql -i create_script.sql -o error.log -m
```

where, the `-i` key is the input file of the script; the `-o` key is the output file of messages; the `-m` key makes isql output error messages to the output message file.

Next, look at the file `error.log` for errors, and if they are found, change the metadata in the original database. Repeat the algorithm described above until all errors are eliminated. After that, you can safely do backup/restore.

Next, we will list some of the most common compatibility issues at the SQL level, which you can fix before moving to the new ODS. You can read the full list of incompatibilities in Release Notes 5.0 in the chapter "Compatibility Issues". When migrating from 3.0, you should also familiarize yourself with the chapter of the same name in Release Notes 4.0, and when migrating from 2.5 - Release Notes 3.0.

2.1.1. New reserved words

Check your database for new reserved words in identifiers, columns, and variables. In the first SQL dialect, such words cannot be used at all (you will have to rename them), in the third - they can be used, but must be enclosed in double quotes.

You can find the list of new keywords and reserved words in the Release Notes 3.0 and 4.0 in the chapter "Reserved Words and Changes". Keywords can be used as identifiers, although this is not recommended.

Starting from Firebird 5.0, you can view the full list of keywords and reserved words using the query:

```
SELECT
RDB$KEYWORD_NAME,
RDB$KEYWORD_RESERVED
FROM RDB$KEYWORDS
```

This query can be executed on any database with ODS 13.1, for example on employee.db, included in the Firebird 5.0 distribution.

The column `RDB$KEYWORD_NAME` contains the keyword itself, and `RDB$KEYWORD_RESERVED` - the flag whether the keyword is reserved.

2.1.2. Column names in PSQL cursors

Relevant: when migrating from Firebird 2.5.

All output columns in PSQL cursors declared as `DECLARE CURSOR` must have an explicit name or alias. The same applies to PSQL cursors used as `FOR SELECT ⋯ AS CURSOR <cursor name> DO ⋯`.

Example 5. Problem with unnamed columns in cursors

```
create procedure sp_test
returns (n int)
as
  declare c cursor for (select 1 /* as a */ from rdb$database);
begin
  open c;
  fetch c into n;
  close c;
  suspend;
end

Statement failed, SQLSTATE = 42000
unsuccessful metadata update
-ALTER PROCEDURE SP_TEST failed
-Dynamic SQL Error
-SQL error code = -104
-Invalid command
-no column name specified for column number 1 in derived table C
```

2.1.3. New data types

Relevant: when migrating from Firebird versions 2.5, 3.0.

Firebird 4.0 introduces new data types:

- TIMESTAMP WITH TIME ZONE
- TIME WITH TIME ZONE
- INT128
- NUMERIC(38, x) and DECIMAL(38, x)
- DECFLOAT(16) and DECFLOAT(34)

The last two types do not cause much trouble, since you did not use them before, and usually expressions do not return them.

Some expressions can now return types NUMERIC(38, x), DECIMAL(38, x) and INT128. We will talk about solving this problem later, since at the stage of changing the ODS they usually do not manifest themselves.

Expressions CURRENT_TIMESTAMP and CURRENT_TIME now return types TIMESTAMP WITH TIME ZONE and TIME WITH TIME ZONE.

For old client libraries and applications, you can set the data type compatibility mode, but this will not help inside stored procedures, functions, and triggers. You need to use expressions LOCALTIMESTAMP and LOCALTIME instead of CURRENT_TIMESTAMP and CURRENT_TIME where you do not want to get data types with time zones. These expressions were specially introduced in the corrective releases Firebird 2.5.9 and Firebird 3.0.4, so that you could prepare your databases for migration to Firebird 4.0 and higher in advance.

When assigning a variable (column) of type TIMESTAMP the value of the expression CURRENT_TIMESTAMP, a type conversion will be performed, that is, an implicit CAST(CURRENT_TIMESTAMP AS TIMESTAMP), so even without replacing CURRENT_TIMESTAMP and CURRENT_TIME with LOCALTIMESTAMP and LOCALTIME everything will continue to work, but performance in some cases may drop. For example:

```
create global temporary table gtt_test (
id  integer not null,
t   timestamp default current_timestamp
) on commit preserve rows;

alter table gtt_test add constraint pk_gtt_test primary key (id);
```

Here the field t has the type TIMESTAMP, and CURRENT_TIMESTAMP returns TIMESTAMP WITH TIME ZONE, which reduces the performance of INSERT into such a table.

This case is described in detail in the bug tracker, ticket 7854.

Initially, the performance drop was 30%, which is quite significant, but after a series of optimizations, the overhead was reduced to 3-5%. If you do not want extra costs, it is better to use LOCALTIMESTAMP where it is not supposed to operate with time with a time zone.

2.1.4. Date and time literals

Relevant: when migrating from Firebird versions 2.5, 3.0.

In Firebird 4.0, the syntax of date and time literals is tightened.

Literals 'NOW', 'TODAY', 'TOMORROW', 'YESTERDAY' with prefixes `TIMESTAMP`, `DATE`, `TIME` are now prohibited. The fact is that the value of such literals was calculated at the time of preparing the DSQL query or compiling the PSQL modules, which led to unexpected results.

If something like `TIMESTAMP 'NOW'` was used in DSQL queries in the application code or in the transferred PSQL, there will be a compatibility problem with Firebird 4 and higher.

Example 6. The following code will not compile

```
..
DECLARE VARIABLE moment TIMESTAMP;
..
SELECT TIMESTAMP 'NOW' FROM RDB$DATABASE INTO :moment;

/* here the variable: moment will be "frozen" as a timestamp
at the time of the last compilation of the procedure or function  */
..
```

You need to clean up such literals, for example, replace them with an explicit conversion `CAST('NOW' AS TIMESTAMP)`, in the code of your procedures and functions before converting your database to a new ODS.

In addition, you need to check other date and time literals with an explicit assignment of a known date time. Previously, separators of parts of the date and time that did not correspond to the standard were allowed in such literals. Now such separators are prohibited. You can read more about the allowed formats of date and time literals in the "Firebird 5.0 SQL Language Reference" in the chapter "Date and Time Literals".

2.1.5. INSERT ... RETURNING requires SELECT privilege

Relevant: when migrating from Firebird versions 2.5, 3.0.

Starting from Firebird 4.0, if any `INSERT` statement contains a `RETURNING` clause that references columns of the base table, the calling party must be granted the corresponding `SELECT` privilege.

2.1.6. Cursor stability effect

Relevant: when migrating from Firebird 2.5.

In Firebird 3.0, an important improvement was made, which is called "cursor stability". As a result of this improvement, some queries may work differently. This primarily concerns queries that modify the table and read it in the same cursor. Cursor stability allows you to eliminate many

errors that were present in previous versions of Firebird, the most famous of which is the infinite loop in the query:

```
INSERT INTO some_table
SELECT * FROM some_table
```

It is unlikely that your applications contain exactly such queries, however, cursor stability can manifest itself in not quite obvious cases:

- some DML trigger modifies the table, and then in the same trigger there is a read of this table through the SELECT operator. If the data was modified not in the current context of the trigger execution, then you may not see the changes in the SELECT query;
- a selective stored procedure SP_SOME modifies records in some table SOME_TABLE, and then you perform a JOIN with the same table:

  ```
  FOR
  SELECT ...
  FROM SP_SOME(...) S
  JOIN SOME_TABLE ...
  ```

If your code contains such cases, we recommend rewriting these parts taking into account the effect of "cursor stability".

2.2. Support for external functions (UDF) is declared obsolete

Support for external functions (UDF) starting from Firebird 4 is declared obsolete.

The effect of this is that UDF cannot be used with the default configuration, since for the parameter `UdfAccess` in `firebird.conf` the default value is now None. UDF libraries `ib_udf` and `fbudf` are removed from the distribution.

Most of the functions in these libraries are already obsolete in previous versions of Firebird and have been replaced by built-in analogues. Now safe replacements for some of the remaining functions are available either in the new library of user-defined subprograms (UDR) with the name `[lib]udf_compat.[dll/so/dylib]` (this is done after changing the ODS), or in the form of script conversions to stored PSQL functions.

We recommend replacing UDF functions with their built-in analogues in advance (before switching to a new ODS). If you are migrating from Firebird 3.0, you can also rewrite some functions in PSQL.

If after these steps you still have UDF functions, you need to change the configuration parameter

```
UdfAccess = Restrict UDF
```

2.3. Converting the database to a new ODS

After preliminary preparation, you can try to convert the database to a new ODS using the gbak tool.

It is recommended to always start with backup/resore of metadata:

```
old_version\gbak -b -g -m old_db stdout | new_version\gbak -c -m stdin
new_db
```

Otherwise, you can get a metadata error after the entire terabyte of data has been written, which will be very disappointing. In addition, on restored in the new version of metadata, it is convenient to check the work of scripts for recompiling database objects. In this example, it is assumed that Firebird 3.0 and Firebird 5.0 are installed on the same machine. Firebird 3.0 works using TCP port 3053, and Firebird 5.0 - 3055.

First of all, you need to create a backup of your database on the current version of Firebird using the following command.

```
gbak -b -g -V -user <username> -pas <password> -se <service> <database> <backup_file>
-Y <log_file>
```

Example 7. Creating a backup on the current version of Firebird

```
gbak -b -g -V -user SYSDBA -pas 8kej712 -se server/3053:service_mgr my_db
d:\fb30_backup\my_db.fbk -Y d:\fb30_backup\backup.log
```

Next, you need to restore your copy on Firebird 5.0.

```
gbak -c -v -user <username> -pas <password> -se <service> <backup_file>
<database_file> -Y <log_file>
```

Starting from Firebird 5.0, the gbak utility can create a backup and restore a database using parallelism. The number of parallel threads used for backup or restore is specified using the `-parallel` or abbreviated `-par` option. Using parallel threads can speed up the restore process by 2-3 times, depending on your hardware and database.

By default, parallelism is disabled in Firebird 5.0. To be able to use it, you need to set the `MaxParallelWorkers` parameter in `firebird.conf`. This parameter limits the maximum number of parallel threads that can be used by the Firebird core or its utilities. By default, it is equal to 1. It is recommended to set `MaxParallelWorkers` to a value equal to the maximum number of physical or logical cores of your processor (or processors).

Now for the restore you can use the following command.

```
gbak -c -par <N>  -v -user <username> -pas <password> -se <service> <backup_file>
<database_file> -Y <log_file>
```

Here `N` is the number of parallel threads that gbak will use, it must be less than or equal to the value set in MaxParallelWorkers.

Example 8. Restoring a backup on Firebird 5.0 using 8 parallel threads

```
gbak -c -par 8 -v -user SYSDBA -pas 8kej712 -se server/3055:service_mgr
d:\fb30_backup\my_db.fbk d:\fb50_data\my_db.fdb -Y d:\fb50_data\restore.log
```

Important

Pay attention to the switches `-V` and `-Y`, they must be used so that you can see in the log file what went wrong during the restore.

After the restore, carefully examine the `restore.log` for errors. However, there will be no SQL level compatibility errors in this log, since the objects DB at restore are not recompiled. If some procedure or trigger contains incompatible constructions, then later when `ALTER` such an object, an error will be issued.

You can completely clear the DB of such errors only if you extract the script from the DB operation

```
isql -x <database> > script.sql
```

in the previous version of Firebird, and create an empty DB in Firebird 5.0 from this script, correcting the errors of creating metadata in turn.

2.3.1. Warnings about missing UDF

After the restore, you may see the following warnings in the `restore.log` file

```
gbak: WARNING:function UDF_FRAC is not defined
gbak: WARNING: module name or entrypoint could not be found
```

This means that you have UDFs that are declared in the database, but their library is missing. It has already been described above what to do in this case. But this mainly concerned your UDF libraries. However, if you used UDFs from the package supplied with Firebird, namely `ib_udf` and `fbudf`, then you can replace them with built-in functions or with safe UDR analogues located in the `udf_compat.dll` library. To do this, you need to run the SQL migration script supplied with Firebird 5.0, which is located in `misc/upgrade/v4.0/udf_replace.sql`. This is done by the following command

```
isql -user sysdba -pas masterkey -i udf_replace.sql {your-database}
```

Example 9. Warning

This script will not affect the declarations of UDFs from third-party libraries!

2.3.2. Fast ODS upgrade when migrating from Firebird 4.0

If you are migrating from Firebird 4.0, there is a faster way to upgrade ODS than backup/restore.

The traditional way of upgrading ODS (On-Disk Structure) is to perform a backup on the old version of Firebird and a restore on the new one. This is a rather lengthy process, especially on large databases.

However, in the case of upgrading a minor version of ODS (the number after the dot) backup/restore is redundant (it is only necessary to add the missing system tables and fields, as well as some packages). An example of such an upgrade is upgrading ODS 13.0 (Firebird 4.0) to ODS 13.1 (Firebird 5.0), since the major version of ODS 13 remained the same.

Starting from Firebird 5.0, there is a possibility to upgrade the minor version of ODS without the lengthy backup and restore operations. For this, the `gfix` utility is used with the `-upgrade` switch.

Key points:

- The upgrade must be done manually using the command `gfix -upgrade`
- Exclusive access to the database is required, otherwise an error is issued.
- The system privilege `USE_GFIX_UTILITY` is required.
- The upgrade is transactional, all changes are rolled back in case of an error.
- After the upgrade, Firebird 4.0 can no longer open the database.

- This is a one-way modification, return is impossible. Therefore, before upgrading, make a copy of the database (using `nbackup -b 0`), to have a restore point, if something goes wrong during the process.
- Upgrading ODS using `gfix -upgrade` does not change the data pages of user tables, thus records will not be repacked using the new RLE compression algorithm. But newly inserted records will be compressed using the improved RLE.

Thus, for a fast ODS upgrade, you need to do the following steps:

- Make a backup of the database, for example using `nbackup -b 0`, to have a restore point, if something goes wrong.
- Execute the command:

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

This way of upgrading ODS, unlike backup/restore, takes seconds (we are talking about `gfix`

`-upgrade`), not minutes or hours.

Chapter 3. Transfer of database aliases

This section is relevant for those who are migrating from Firebird 2.5.

The file `aliases.conf` in which the database aliases were configured has been renamed to `databases.conf`. It is fully backward compatible in syntax, but its purpose has been significantly expanded. Now it is possible to set some individual parameters for each database. We strongly recommend using this feature if your server serves more than one database.

The parameters that can be set at the database level are marked in the file firebird.conf with the inscription 'Per-database configurable'.

Chapter 4. Transfer of user list

The transfer of the user list from Firebird versions 2.5, 3.0 and 4.0 is done differently.

4.1. Transfer of user list from Firebird 4.0

The easiest will be to transfer the user list from Firebird 4.0.

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

Copying the file `security4.fdb` and renaming it to `security5.fdb` and upgrading ODS using the `gfix -UPGRADE` option will also work, but we recommend performing a backup and restore.

4.2. Transfer of user list from Firebird 3.0

To transfer users from the Firebird 3.0 security database to the Firebird 4.0 security database, you need to perform a backup of `security3.fdb` using gbak and restore it as `security5.fdb` using `gbak` Firebird 5.0.

However, keep in mind that in this case you will lose some new features. We will go the more difficult way:

1. Make a backup of the security database on Firebird 3.0

```
c:\Firebird\3.0>gbak -b -g -user SYSDBA security.db d:\fb30_backup\security.fbk
```

2. Restore the backup on Firebird 5.0 under a new name

```
c:\Firebird\5.0>gbak -c -user SYSDBA -pas 8kej712 -se localhost/3054:service_mgr
d:\fb30_backup\security.fbk d:\fb50_data\security_30.fdb
```

3. Save the following script to transfer users to the file `copy_user.sql`

```
set term ^;

EXECUTE BLOCK
AS
  -- change path to the copy of your security database
  DECLARE SRC_SEC_DB     VARCHAR(255) = 'c:\fb50_data\security_30.fdb';
  DECLARE SRC_SEC_USER   VARCHAR(63) = 'SYSDBA';
  ------------------------------------------------
  DECLARE PLG$USER_NAME  SEC$USER_NAME;
```

```
  DECLARE PLG$VERIFIER   VARCHAR(128) CHARACTER SET OCTETS;
  DECLARE PLG$SALT       VARCHAR(32) CHARACTER SET OCTETS;
  DECLARE PLG$COMMENT    BLOB SUB_TYPE TEXT CHARACTER SET UTF8;
  DECLARE PLG$FIRST      SEC$NAME_PART;
  DECLARE PLG$MIDDLE     SEC$NAME_PART;
  DECLARE PLG$LAST       SEC$NAME_PART;
  DECLARE PLG$ATTRIBUTES BLOB SUB_TYPE TEXT CHARACTER SET UTF8;
  DECLARE PLG$ACTIVE     BOOLEAN;
  DECLARE PLG$GROUP_NAME SEC$USER_NAME;
  DECLARE PLG$UID        PLG$ID;
  DECLARE PLG$GID        PLG$ID;
  DECLARE PLG$PASSWD     PLG$PASSWD;
BEGIN
  -- move users of SRP plugin
  FOR EXECUTE STATEMENT Q'!
      SELECT
          PLG$USER_NAME,
          PLG$VERIFIER,
          PLG$SALT,
          PLG$COMMENT,
          PLG$FIRST,
          PLG$MIDDLE,
          PLG$LAST,
          PLG$ATTRIBUTES,
          PLG$ACTIVE
      FROM PLG$SRP
      WHERE PLG$USER_NAME <> 'SYSDBA'
!'
          ON EXTERNAL :SRC_SEC_DB
          AS USER :SRC_SEC_USER
          INTO :PLG$USER_NAME,
               :PLG$VERIFIER,
               :PLG$SALT,
               :PLG$COMMENT,
               :PLG$FIRST,
               :PLG$MIDDLE,
               :PLG$LAST,
               :PLG$ATTRIBUTES,
               :PLG$ACTIVE
  DO
  BEGIN
    INSERT INTO PLG$SRP (
        PLG$USER_NAME,
        PLG$VERIFIER,
        PLG$SALT,
        PLG$COMMENT,
        PLG$FIRST,
        PLG$MIDDLE,
        PLG$LAST,
        PLG$ATTRIBUTES,
        PLG$ACTIVE)
```

```
    VALUES (
        :PLG$USER_NAME,
        :PLG$VERIFIER,
        :PLG$SALT,
        :PLG$COMMENT,
        :PLG$FIRST,
        :PLG$MIDDLE,
        :PLG$LAST,
        :PLG$ATTRIBUTES,
        :PLG$ACTIVE);
  END
  -- move users of plugin Legacy_UserManager
  FOR EXECUTE STATEMENT Q'!
      SELECT
          PLG$USER_NAME,
          PLG$GROUP_NAME,
          PLG$UID,
          PLG$GID,
          PLG$PASSWD,
          PLG$COMMENT,
          PLG$FIRST_NAME,
          PLG$MIDDLE_NAME,
          PLG$LAST_NAME
      FROM PLG$USERS
      WHERE PLG$USER_NAME <> 'SYSDBA'
!'
          ON EXTERNAL :SRC_SEC_DB
          AS USER :SRC_SEC_USER
          INTO :PLG$USER_NAME,
               :PLG$GROUP_NAME,
               :PLG$UID,
               :PLG$GID,
               :PLG$PASSWD,
               :PLG$COMMENT,
               :PLG$FIRST,
               :PLG$MIDDLE,
               :PLG$LAST
  DO
  BEGIN
    INSERT INTO PLG$USERS (
        PLG$USER_NAME,
        PLG$GROUP_NAME,
        PLG$UID,
        PLG$GID,
        PLG$PASSWD,
        PLG$COMMENT,
        PLG$FIRST_NAME,
        PLG$MIDDLE_NAME,
        PLG$LAST_NAME)
    VALUES (
        :PLG$USER_NAME,
```

```
        :PLG$GROUP_NAME,
        :PLG$UID,
        :PLG$GID,
        :PLG$PASSWD,
        :PLG$COMMENT,
        :PLG$FIRST,
        :PLG$MIDDLE,
        :PLG$LAST);
  END
END^

set term ;^

commit;

exit;
```

Important

Do not forget to replace the value of the SRC_SEC_DB variable with the path to the copy of your security database.

Note

We excluded the copy of the SYSDBA user, since we initialized it during installation.

4. Run the script on Firebird 5.0 by connecting to the security database in embedded mode

```
c:\Firebird\5.0>isql -i "d:\fb50_data\copy_users.sql" -u SYSDBA -ch UTF8
security.db
```

Congratulations! Your users have been transferred with all attributes and passwords.

4.3. Transfer of user list from Firebird 2.5

Transferring users from Firebird 2.5 is more difficult. In Firebird 3.0, a new authentication method was introduced - SRP - Secure Remote Password Protocol. The old authentication method is also available, but disabled by default as it is considered insufficiently secure. The Release Notes 3.0 describe how to transfer users from `Legacy_UserManager` to SRP, but in this case you will not be able to connect via `fbclient` version 2.5. In addition, it is impossible to transfer passwords from `Legacy_UserManager` to SRP. The proposed script will transfer the user list, but random passwords will be generated. If you want to restore your previous passwords, you will have to do it manually. We wrote an alternative script that allows you to transfer users from `security2.fdb` to `security5.fdb` in the `Legacy_UserManager` plugin. Here we will describe both options.

4.3.1. Copying user list to SRP plugin

Due to the new authentication model in Firebird 3, updating the security database version 2.5 (`security2.fdb`) directly for use in Firebird 5 is impossible. However, there is an update procedure that allows you to save the user account data - username, name and other attributes, but not passwords - from the `security2.fdb` database, which was used on servers version 2.x.

The procedure requires running the script `security_database.sql`, which is located in the `misc/upgrade` directory of your Firebird 3 installation. These instructions assume that you have a temporary copy of this script in the same directory as the isql executable.

Note

- In Firebird 5.0, the security database update script file `security_database.sql` is missing from the `misc/upgrade` directory, so you need to download the zip archive with the Firebird 3.0 distribution.
- In the commands below, replace *masterkey* with the actual `SYSDBA` password for your server, if necessary.

1. Make a backup of the security database `security2.fdb `on Firebird 2.5

```
c:\Firebird\2.5>bin\gbak -b -g -user SYSDBA -password masterkey -se service_mgr
c:\Firebird\2.5\security2.fdb d:\fb25_backup\security2.fbk
```

2. Restore the backup on Firebird 5.0

```
c:\Firebird\5.0>gbak -c -user SYSDBA -password masterkey -se
localhost/3054:service_mgr d:\fbdata\5.0\security2.fbk
d:\fbdata\5.0\security2db.fdb -v
```

3. On the Firebird 5.0 server, go to the directory where the `isql` utility is located, and run the update script:

```
isql -user sysdba -pas masterkey -i security_database.sql
{host/path}security2db.fdb
```

   `security2db.fdb` is just an example database name: it can be any preferred name.

4. The procedure generates new random passwords and then displays them on the screen. Copy the output and notify the users of their new passwords.

4.3.2. Copying user list to Legacy_UserManager plugin

Unlike the previous option, this script will save your original passwords. However, we advise you to switch to the `Srp` plugin in the future anyway.

1. Make a backup of the security database `security2.fdb` on Firebird 2.5

```
c:\Firebird\2.5>bin\gbak -b -g -user SYSDBA -password masterkey -se service_mgr
c:\Firebird\2.5\security2.fdb d:\fb25_backup\security2.fbk
```

2. Deploy the backup on Firebird 5.0

```
c:\Firebird\5.0>gbak -c -user SYSDBA -password masterkey -se
localhost/3054:service_mgr d:\fbdata\5.0\security2.fbk
d:\fbdata\5.0\security2db.fdb -v
```

3. Save the following script to transfer users to the file `copy_security2.sql`

```
set term ^;

EXECUTE BLOCK
AS
  -- change path to the copy of your security database
  DECLARE SRC_SEC_DB     VARCHAR(255) = 'd:\fbdata\5.0\security2.fdb';
  DECLARE SRC_SEC_USER   VARCHAR(63) = 'SYSDBA';
  ------------------------------------------------
  DECLARE PLG$USER_NAME  SEC$USER_NAME;
  DECLARE PLG$COMMENT    BLOB SUB_TYPE TEXT CHARACTER SET UTF8;
  DECLARE PLG$FIRST      SEC$NAME_PART;
  DECLARE PLG$MIDDLE     SEC$NAME_PART;
  DECLARE PLG$LAST       SEC$NAME_PART;
  DECLARE PLG$GROUP_NAME SEC$USER_NAME;
  DECLARE PLG$UID        INT;
  DECLARE PLG$GID        INT;
  DECLARE PLG$PASSWD     VARBINARY(64);
BEGIN
  FOR EXECUTE STATEMENT q'!
      SELECT
          RDB$USER_NAME,
          RDB$GROUP_NAME,
          RDB$UID,
          RDB$GID,
          RDB$PASSWD,
          RDB$COMMENT,
          RDB$FIRST_NAME,
          RDB$MIDDLE_NAME,
          RDB$LAST_NAME
      FROM RDB$USERS
      WHERE RDB$USER_NAME <> 'SYSDBA'
!'
      ON EXTERNAL :SRC_SEC_DB
      AS USER :SRC_SEC_USER
      INTO
          :PLG$USER_NAME,
          :PLG$GROUP_NAME,
```

```
          :PLG$UID,
          :PLG$GID,
          :PLG$PASSWD,
          :PLG$COMMENT,
          :PLG$FIRST,
          :PLG$MIDDLE,
          :PLG$LAST
  DO
  BEGIN
    INSERT INTO PLG$USERS (
        PLG$USER_NAME,
        PLG$GROUP_NAME,
        PLG$UID,
        PLG$GID,
        PLG$PASSWD,
        PLG$COMMENT,
        PLG$FIRST_NAME,
        PLG$MIDDLE_NAME,
        PLG$LAST_NAME)
    VALUES (
        :PLG$USER_NAME,
        :PLG$GROUP_NAME,
        :PLG$UID,
        :PLG$GID,
        :PLG$PASSWD,
        :PLG$COMMENT,
        :PLG$FIRST,
        :PLG$MIDDLE,
        :PLG$LAST);
  END
END^

set term ;^

commit;

exit;
```

Important

Do not forget to replace the value of the `SRC_SEC_DB` variable with the path to the copy of your security database.

Note

We excluded the copy of the SYSDBA user, since we initialized it during installation.

4. Run the script on Firebird 5.0 by connecting to the security database in embedded mode

```
c:\Firebird\5.0>isql -i "d:\fb40_data\copy_security2.sql" -u SYSDBA -ch UTF8
security.db
```

Congratulations! Your users have been transferred with all attributes and passwords.

Chapter 5. Setting up trusted authentication

Setting up trusted authentication in Firebird 5.0 is done exactly the same way as it was done in Firebird 3.0 or 4.0. For those migrating from Firebird 2.5, we will describe this process in more detail.

1. First of all, you need to setup trusted authentication plugin in the configuration file `firebird.conf` or `databases.conf` in the AuthServer parameter (by default it is disabled). To do this, you need to add a plugin with the name `Win_Sspi`, and we will use it together with `Srp256`.

   ```
   AuthServer = Srp256, Win_Sspi
   ```

2. The next step is to enable the mapping of users from `Win_Sspi` to `CURRENT_USER`. To do this, you need to create a mapping in the target database using the following query

   ```
   CREATE MAPPING TRUSTED_AUTH
   USING PLUGIN WIN_SSPI
   FROM ANY USER
   TO USER;
   ```

   This SQL query creates a mapping only at the level of the current database. The mapping will not apply to other databases located on the same server. If you want to create a common mapping for all databases, then add the `GLOBAL` keyword.

   ```
   CREATE GLOBAL MAPPING TRUSTED_AUTH
   USING PLUGIN WIN_SSPI
   FROM ANY USER
   TO USER;
   ```

3. Enabling SYSDBA-like access for Windows administrators (if needed).

   To enable such access, you need to create the following mapping

   ```
   CREATE MAPPING WIN_ADMINS
   USING PLUGIN WIN_SSPI
   FROM Predefined_Group
   DOMAIN_ANY_RID_ADMINS
   TO ROLE RDB$ADMIN;
   ```

   Instead of enabling SYSDBA-like access for all Windows administrators, you can give administrative privileges to a specific user with the following mapping

   ```
   create global mapping cto_sysdba
   using plugin win_sspi
   ```

```
from user "STATION9\DEVELOPER"
to user SYSDBA;
```

Chapter 6. Application-level incompatibilities

At the API level, the fbclient 5.0 library is compatible with previous versions. However, there may be compatibility issues at the level of some SQL queries. Most of them we have already described earlier in the section List of incompatibilities at the SQL language level. Next, we will describe some other problems that may arise in the application.

6.1. Removed network protocol WNET

The network protocol WNET (also known as Named Pipes, also known as NetBEUI), previously supported on the Windows platform, has been removed in Firebird 5.0.

Windows users who worked with any WNET connection string (`\\server\dbname` or `wnet://server/dbname`) should instead switch to the INET (TCP) protocol (connection string `server:dbname`, `server/port:dbname`, `inet://server/dbname` or `inet://server:port/dbname`).

6.2. New data types

Relevant: when migrating from Firebird versions 2.5, 3.0.

As mentioned earlier, some expressions may return new data types that cannot be interpreted by your application without modification. Such modification may take a significant amount of time or be beyond your capabilities. To simplify migration to new versions, you can set the `DataTypeCompatibility` parameter to the compatibility mode with the required version in `firebird.conf` or `databases.conf`.

```
DataTypeCompatibility = 3.0
```

or

```
DataTypeCompatibility = 2.5
```

This is the fastest way to achieve compatibility with new data types. However, over time you may start to implement support for new types in your application. Naturally, this will happen gradually - first one type, then another, and so on. In this case, you need to configure the mapping of those types that you have not yet completed, to other data types. For this, the operator `SET BIND OF` is used.

Syntax

```
SET BIND OF { <type-from> | TIME ZONE } TO { <type-to> | LEGACY | NATIVE | EXTENDED }
```

The keyword `LEGACY` in the `TO` part is used when a data type that is absent in the previous version of Firebird should be represented in a way that is understandable to the old client software (some data loss is possible). There are the following conversions to `LEGACY` types:

Table 1. Conversions to legacy types

DataTypeCompatibility	Native type	Legacy type
2.5	BOOLEAN	CHAR(5)
2.5 or 3.0	DECFLOAT	DOUBLE PRECISION
2.5 or 3.0	INT128	BIGINT
2.5 or 3.0	TIME WITH TIME ZONE	TIME WITHOUT TIME ZONE
2.5 or 3.0	TIMESTAMP WITH TIME ZONE	TIMESTAMP WITHOUT TIME ZONE

When setting the DataTypeCompatibility parameter, new data types are converted to legacy types according to the table described above.

A detailed description of this operator is available in the "Firebird 4.0 Release Notes" and "Firebird 5.0 SQL Language Reference". With it, you can control the mapping of new types in your application by executing the corresponding query immediately after connecting, and even write an `AFTER CONNECT` trigger in which to use several such operators.

For example, suppose you have added support for date and time with time zones to your application, but you still do not support the types `INT128` and `DECFLOAT`. In this case, you can write the following trigger.

```
create or alter trigger tr_ac_set_bind
on connect
as
begin
  set bind of int128 to legacy;
  set bind of decfloat to legacy;
end
```

6.3. Consistent reading in READ COMMITTED transactions

Relevant: when migrating from Firebird versions 2.5, 3.0.

Firebird 4 not only introduces consistent reading (`READ CONSISTENCY`) for queries in `READ COMMITTED` transactions, but also makes it the default mode for all `READ COMMITTED` transactions, regardless of their `RECORD VERSION` or `NO RECORD VERSION` properties.

This is done to provide users with better behavior - both conforming to the SQL specification and less prone to conflicts. However, this new behavior may also have unexpected side effects.

Perhaps the most important of them is the so-called restarts when processing update conflicts. This can lead to some code that is not subject to transactional control being executed multiple times within PSQL. Examples of such code may be:

- using external tables, sequences, or context variables;
- sending emails using UDF;
- using autonomous transactions or external queries.

In the isolation mode READ COMMITTED READ CONSISTENCY, the update conflict is handled differently. If an `UPDATE` or `DELETE` statement detects a record that has already been modified or deleted by another transaction (the transaction is committed), then all changes made in the current query are rolled back and it is executed again. This is called a query restart.

More about consistent reading in `READ COMMITTED` transactions can be read in the "Firebird 4.0 Release Notes".

Another important effect is that unfetched cursors in `READ COMMITTED READ CONSISTENCY` transactions in Read Only mode now hold garbage collection. We recommend that you stop using a single long `READ COMMITTED READ ONLY` transaction in your application, and replace it with several such transactions, each of which is active for as long as necessary.

If the features of the `READ CONSISTENCY` mode are undesirable for some reason, then to restore the obsolete behavior, you need to set the configuration parameter `ReadConsistency` to 0.

6.4. Changes in the optimizer

The optimizer changes in each version of Firebird. Mostly these changes are positive, that is, your queries should run faster, but some queries may slow down, so you need to test the performance of your application, and if there is a slowdown somewhere, you need intervention from the programmer.

For most optimizer changes, you cannot influence the query plan by changing the server configuration. In this case, you can do the following:

- rewrite the SQL query so that it runs faster on the new version of the server;
- create or delete indexes;
- if none of the above helped, then create a regression ticket at https://github.com/FirebirdSQL/firebird/issues.

There are a couple of points in the optimizer's work that can be influenced by changing the configuration:

6.4.1. Using Refetch for sorting wide data sets

Relevant: when migrating from Firebird versions 2.5, 3.0.

Starting from Firebird 4.0, a new access method Refetch was introduced, which allows to optimize the sorting of wide data sets. A wide data set is a data set in which the total length of the record fields is large.

Historically, when performing external sorting, Firebird writes both key fields (i.e., those specified

in the ORDER BY or GROUP BY clause) and non-key fields (all other fields that have references within the query) to sorting blocks, which are either stored in memory or in temporary files. After the sorting is completed, these fields are read back from the sorting blocks. Usually this approach is considered faster, since records are read from temporary files in the order corresponding to the sorted records, rather than selected randomly from the data page. However, if the non-key fields are large (for example, long VARCHARs are used), this increases the size of the sorting blocks and, thus, leads to more I/O operations for temporary files. Firebird 4 offers an alternative approach (the Refetch access method), when only key fields and DBKEY records are stored inside the sorting blocks, and non-key fields are extracted from the data pages after sorting. This improves the performance of sorting in the case of long non-key fields.

Thus, the plans of your queries using sorting may change. To control this access method, a new configuration parameter InlineSortThreshold was introduced. The value specified for InlineSortThreshold determines the maximum size of the sorting record (in bytes) that can be stored inline, i.e. inside the sorting block. Zero means that records are always refetched. The optimal value of this parameter should be determined experimentally. The default value is 1000 bytes.

Consider the following example:

```
SELECT
  field_1, field_2, field_3, field_4
FROM SomeTable
ORDER BY field_1
```

Before Firebird 4.0, all 4 fields were always included in the sorting blocks. Starting from Firebird 4.0, if the total length of the fields field_1 .. field_4 exceeds the value of InlineSortThreshold, then only field_1 will be included in the sorting blocks, and then Refetch will be performed.

6.4.2. Converting OUTER JOINs to INNER JOINs

There are a number of problems with optimizing OUTER JOINs in Firebird.

First, currently OUTER JOIN can only be performed by one join algorithm NESTED LOOP JOIN, which may be changed in future versions.

Second, when joining streams with outer joins, the join order is strictly fixed, i.e., the optimizer cannot change it to keep the result correct.

However, if there is a predicate in the WHERE condition for the field of the "right" (joined) table, which explicitly does not handle the NULL value, then there is no point in the outer join. In this case, starting from Firebird 5.0, such a join will be converted to an inner one, which allows the optimizer to apply the full range of available join algorithms.

Suppose you have the following query:

```
SELECT
  COUNT(*)
```

```
FROM
  HORSE
  LEFT JOIN FARM ON FARM.CODE_FARM = HORSE.CODE_FARM
WHERE FARM.CODE_COUNTRY = 1
```

In Firebird 5.0, such a query will be implicitly converted to an equivalent form:

```
SELECT
  COUNT(*)
FROM
  HORSE
  JOIN FARM ON FARM.CODE_FARM = HORSE.CODE_FARM
WHERE FARM.CODE_COUNTRY = 1
```

If `LEFT JOIN` was used as a hint to indicate the join order very actively, then rewriting a lot of queries in a new way may be problematic. For such developers, there is a configuration parameter `OuterJoinConversion` in `firebird.conf` or `database.conf`. Setting the parameter OuterJoinConversion to false disables the transformation of Outer Join to inner join. Note that this parameter is a temporary solution to facilitate migration and may be removed in future versions of Firebird.

6.5. RETURNING, returning multiple records

Starting from Firebird 5.0, client modifying operators `INSERT .. SELECT`, `UPDATE`, `DELETE`, `UPDATE OR INSERT` and `MERGE`, containing the `RETURNING` clause, return a cursor, i.e. they are able to return multiple records instead of issuing an error "multiple rows in singleton select", as it happened before.

Now these queries during preparation are described as `isc_info_sql_stmt_select`, whereas in previous versions they were described as isc_info_sql_stmt_exec_procedure.

Singleton operators `INSERT .. VALUES`, as well as positioned operators UPDATE and DELETE (i.e., those containing the `WHERE CURRENT OF` clause) retain the existing behavior and are described as `isc_info_sql_stmt_exec_procedure`.

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

If your application uses modifying operators `INSERT .. SELECT`, `UPDATE`, `DELETE`, `UPDATE OR INSERT` and `MERGE`, containing the `RETURNING` clause, then this may be the cause of errors. Make sure that your driver or access component correctly handles such queries, and if not, then either modify the code (application or component), or wait until an update of the corresponding driver/component that correctly handles these queries is released.

Examples of modifying operators containing `RETURNING`, and returning a data set:

```
INSERT INTO dest(name, val)
SELECT desc, num + 1 FROM src WHERE id_parent = 5
RETURNING id, name, val;
```

```
UPDATE dest
SET a = a + 1
WHERE id = ?
RETURNING id, a;

DELETE FROM dest
WHERE price < 0.52
RETURNING id;

MERGE INTO PRODUCT_INVENTORY AS TARGET
USING (
  SELECT
    SL.ID_PRODUCT,
    SUM(SL.QUANTITY)
  FROM
    SALES_ORDER_LINE SL
    JOIN SALES_ORDER S ON S.ID = SL.ID_SALES_ORDER
  WHERE S.BYDATE = CURRENT_DATE
    AND SL.ID_PRODUCT = :ID_PRODUCT
  GROUP BY 1
) AS SRC(ID_PRODUCT, QUANTITY)
ON TARGET.ID_PRODUCT = SRC.ID_PRODUCT
WHEN MATCHED AND TARGET.QUANTITY - SRC.QUANTITY <= 0 THEN
  DELETE
WHEN MATCHED THEN
  UPDATE SET
    TARGET.QUANTITY = TARGET.QUANTITY - SRC.QUANTITY,
    TARGET.BYDATE = CURRENT_DATE
RETURNING OLD.QUANTITY, NEW.QUANTITY, SRC.QUANTITY;
```

Chapter 7. Mass Migration Framework for Firebird

If you're looking to migrate a bunch of servers—like over a hundred—and you want it done smoothly over just a few days, you might want to check out the Mass Migration Framework for Firebird. It's got all the pro tools and support you'll need to make the migration a breeze.

Chapter 8. Conclusion

In this article, we tried to describe the most common problems and their solutions when migrating to Firebird 5.0 from Firebird 2.5, 3.0 and 4.0. We hope that this article will help you to migrate your databases and applications to Firebird 5.0 and take advantage of the new version.