Getting started with Firebird: part II

Michaël Van Canneyt

January 1, 2008

Abstract

While firebird can run 24/7 without any administration, some tasks must be performed from time to time, or even periodically: taking a backup, adding a new user to the system. In this article, the needed tools for these tasks are presented. Firebird also comes with support for aliases for databases: the configuration of aliases will also be discussed.

1 Introduction

Once the firebird server is up and running, it can be left unattended: it doesn't need operator intervention to keep running. Obviously, there are some tasks that need to be done on every RDBMS: create backups, and do some basic user administration. In case of a calamity, it may even be necessary to try and repair a broken firebird database.

Firebird comes with some command-line tools that perform these tasks: gbak to create backups, gfix to try and fix broken databases, and finally gsec to manage users.

As pointed out in the previous article, Firebird has a feature called aliases: this means that on the server, symbolic names can be given to databases. Doing so for all databases has 2 advantages:

1. When connecting to a database, the user does not need to remember the full filename of the database.
2. The database administrator has the ability to change the location of the database as he sees fit: to a faster drive, or simply to a new database.

For instance, for a database that contains log data, this mechanism can be used to rotate the database on a daily basis without fear that users may lose data.

This feature will be discussed first.

2 Aliases

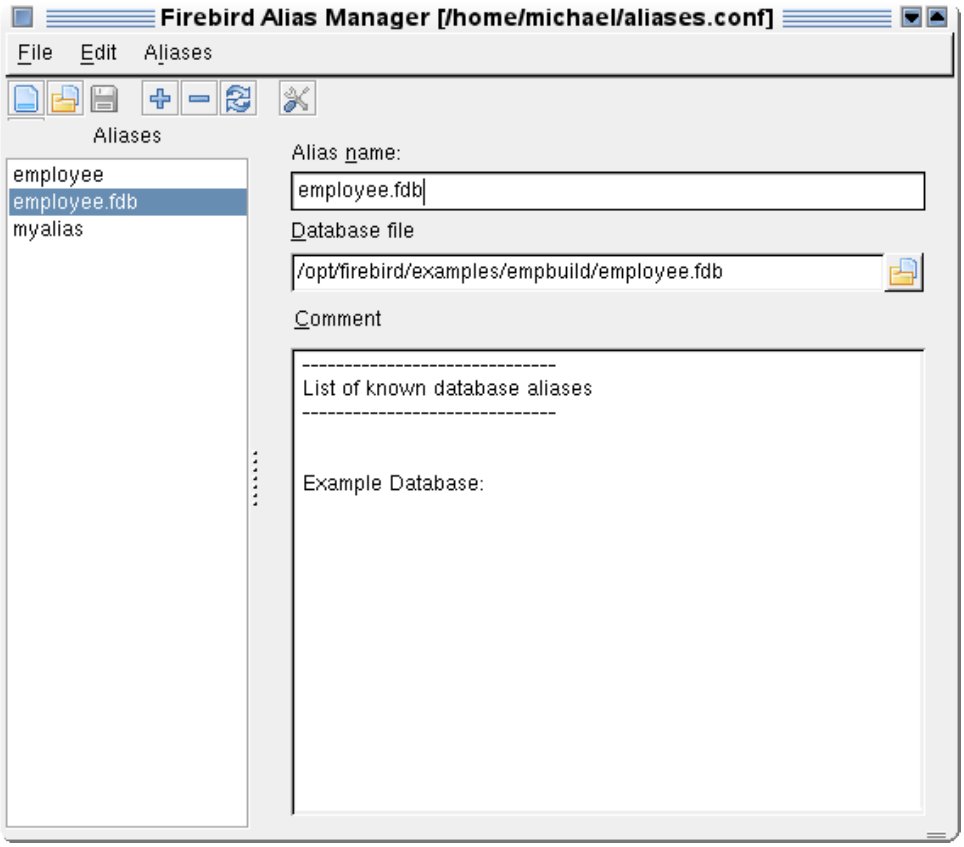
The aliases are in a file in the firebird installation directory, and is called aliases.conf. It is a very simple configuration file, which contains statements of the form:

```
alias = databasefile
```

lines starting with # or empty lines are considered comments. By default it contains an alias for the sample employee database (if it was installed), called 'employee'.

If firebird was installed using the tar.gz installer, it also installs a small script which adds an alias to this file: createAliasDB.sh which can be used very simply:

Figure 1: Firebird alias manager application




```
createAliasDB.sh myalias /path/to/my/database.fdb
```

this will add `myalias` to the aliases files, if no such alias exists yet. The aliases are effective immediatly, no server restart is needed for the aliases to take effect.

On the firebirdsql website there are some small GUI programs that can be used to manage aliases on Windows servers. On the CD accompagnying this issue is a small program (written in Lazarus) which does the same for almost all firebird supported platforms. It is shown in figure 1 on page 2. Note that to edit the aliases file, root or firebird user priviliges are required, depending on the installation. The alias manager can edit an alternate file, which can be moved to it's proper location when finished. It also makes backups of the aliases.conf file if desired.

3 User management

Firebird offers full user and role management for its databases. The users and roles are defined in a central database, the security database. This database is called security2.fdb, and is located in the firebird installation directory. Despite the fact that this is a regular firebird database, it is not possible to connect to this database directly - as a security measure. Adding users or changing their passwords must be done with the firebird API.

To do this, the `gsec` security tool is provided. It can be used to manage users. In general, the tool works as follows:

```
gsec [options] [command] [options]
```

The `Options` are for instance `-USER` and `-PASSWORD` to supply the (sysdba) user name and password. The `Commands` are one of `add delete` or `modify`, which have their obvious meaning.

To add a user, the following command-line can be used:

```
gsec -USER SYSDBA -PASSWORD masterkey -add NewUser -pw PWD
```

This will add a new user `NewUser` with password `PWD`.

To change the password of a user, the following command can be used:

```
gsec -US SYSDBA -PA masterkey -modify NewUser -pw NewPWD
```

As can be seen from this command, the options can be abbreviated.

Deleting a user can be done with the following command:

```
gsec -US SYSDBA -PA masterkey -delete NewUser
```

The gsec tool can also be run interactively: in that case, a command-prompt will appear, asking for commands. To add the user in interactive mode, the following command could be used:

```
~: > gsec -US SYSDBA -PA masterkey
GSEC> add NewUser -pw PWD
GSEC> quit
```

The commands and their options are given as if they were specified on the command-line, except that the command should not be prefixed with a dash (-) character.

To see if a particular user is defined, the `display` command can be used:

```
~: >gsec -US SYSDBA -PA masterkey -display SYSDBA
user name                 uid   gid    full name
------------------------------------------------------
SYSDBA                    0     0      Sql Server Administrator
```

To simplify user management, both IBWebAdmin and Flamerobin have user-management modules, which are more easy to use than the rather terse command-line tool `gsec`. Their interface is very intuitive, and therefore will not be discussed here.

Note that since the user database is a normal database, it can be backed up like any other firebird database, despite the fact that it cannot be accessed directly.

4 Backing up databases

One of the most - if not the most - important tasks of the server manager is to make backups of databases so the database can be restored in case of an accident. Backing up databases can be done while the database is still on-line: the backup is performed in the context of a transaction, so the backup is consistent. It can also be profitable to periodically restore a database from a backup: doing so will clean up database pages (resulting in a smaller database), and will recreate all indexes, making them more balanced (resulting in faster queries).

Sometimes it is also necessary to back up and restore a database when upgrading to a newer version of Firebird: if the ODS (On Disk Structure, the way the engine writes data in the database) changes, then the database must be backed up with the old Firebird version, and restored with the new version.

Firebird delivers a commandline backup tool `gbak` which can be used for this. It's syntax is basically very simple:

```
gbak -USER SYSDBA -PASSWORD secret database.fdb backupfile.fbk
```

The username/password must either be the SYSDBA user or the user that owns the database: For obvious security reasons, only those two users can backup the database. The gbak tool always backs up the complete database. Note that the database can also be on a remote machine, physical access to the database is not required.

There are some options to control the backup format:

-T(ransportable) creates a backup in a format that can be also restored on machines with a different architecture.

-M(ETA_DATA) only backs up metadata, no actual data.

-CO(NVERT) convert tables in external files to actual tables.

The part of the option between parentheses () is optional.

Restoring a database can be done in a similar manner:

```
gbak -C -USER SYSDBA -PASSWORD secret backup.fbk database.fdb
```

The `-C(REATE_DATABASE)` option tells Firebird that a new database database.fdb should be created. If the database with the given name already exists, an error will be generated.

To replace an existing database, the `-REP(LACE_DATABASE)` option can be used instead:

```
gbak -REP -USER SYSDBA -PASSWORD secret backup.fbk database.fdb
```

This will give an error if the database does not yet exist. The `OVERWRITE` option will replace the database or create it if it does not yet exist.

When restoring a databases, there are some options to influence the structure of the created database:

-I(NACTIVE) To speed up the restore process, this option deactivates the indexes during restore. The indexes can be activated using SQL statements after the restore process is completed. This option can also be used when there is invalid data (2 identical entries for a unique index, for example).

-N(O_VALIDITY) When used, the database validity conditions (check constraints) are not restored. This is needed in case the backup contains inconsistent data: enabling this option may ensure that the backup can be restored. By default all check constraints are again enforced when restoring the data, this option disables that check. A case where this can occur is when a check constraint on a field is changed, but the table already contains data that do not match the new constraint.

-O(NE_AT_A_TIME) When this option is used, the database is restore one table at a time, committing the data after each table. This allows a partial restore of a database in case a table has been corrupted beyond repair.

-P(AGE_SIZE) Each database has a page size associated with it. With this option the default page size can be changed to something else. This must be a multiple of 1k (1024), up to 32k.

-USE_(ALL_SPACE) by default, firebird does not fill all database pages completely, so inserts or updates can be peformed without immediatly allocating new pages. With this option, no space for record versions is left in the database pages, creating a smaller database at the expense of insert/update speed.

As can be seen from the options discussed here, it is possible to create a backup which cannot be restored. Therefor it is important to check whether the backups are correct. The best way to do so is by immediatly restoring them in a temporary database.

There exist various - Microsoft Windows based - tools that perform scheduled database backups (fibs, ibstudio and others). For unix (or Darwin) no such tools exist. However, on unix a simple shell script can do a lot of things.

The CD accompagnying this issue contains a script (backupfb) that can be used to create backups of firebird databases on a daily basis using a cron job. It is a regular shell script, which can be configured by editing the value of some variables in the beginning of the script:

BackupDir directory where the backups should be placed.

BaseDir base directory. All databases that need to be backed up should be in this directory or one of its subdirectories.

DatabaseListFile name of a file that contains a list of databases to be backed up. The name is relative to the directory specified in `BaseDir`

DBUser the username for the gbak tool.

DBPassword the password for the gbak tool.

GBAK the gbak program to be used.

Old specifies a period (in days) to wait before a backup is deleted.

After this is done, and the databases to be backed up are listed in the file indicated by `DatabaseListFile`, the script can be installed in crontab with e.g. the following line:

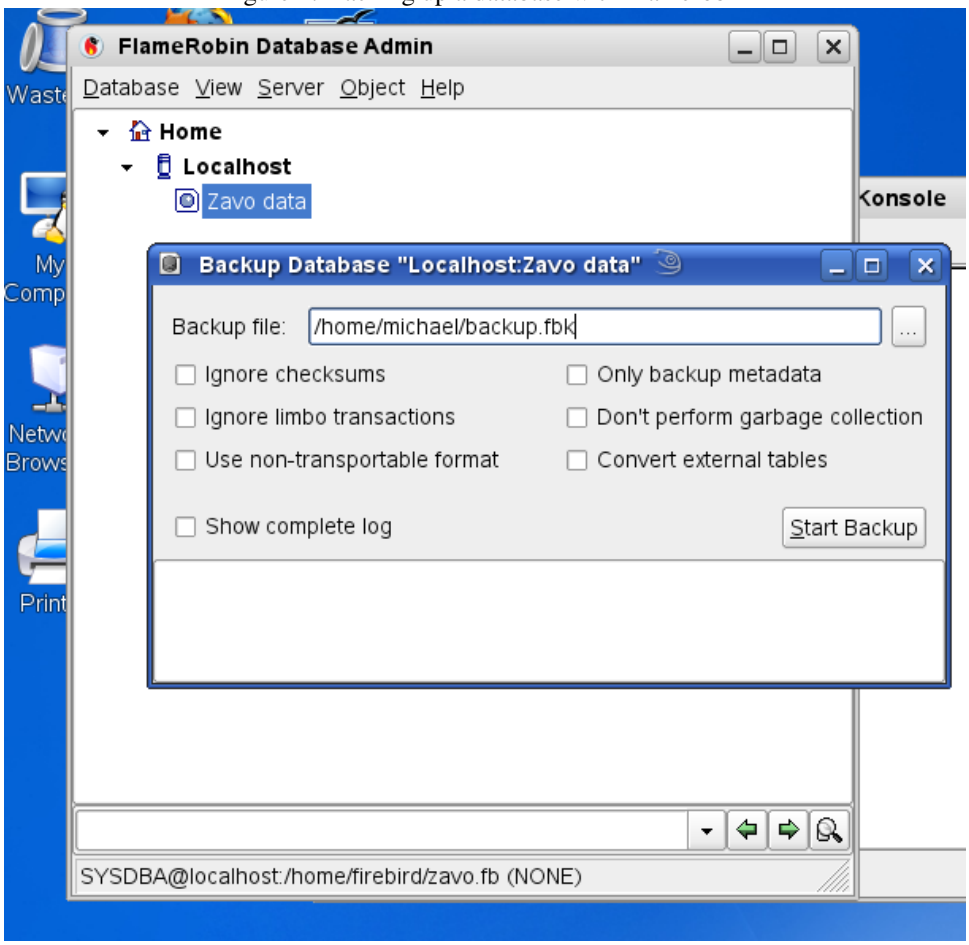
```
~: >crontab -l
# Min Hour min  day month dow command
0 23 * * * /root/bin/backupfb 2>&1
```

This will back up all databases at 23:00 every evening. The backups are compressed using bzip2, and old backups are removed after a certain period. If the database was not touched since the last backup, no backup is performed either.

The script does not attempt to restore databases: while possible to do, this is potentially a very long operation if the database to be restored is very big. If the restore operation were to last too long, multiple instances of the script would be running simultaneously after a day.

The flamerobin and ibWebadmin administration tools have an interface to create and restore backups. The interface is very intuitive, and all options as presented by the gbak tool are also available in these interfaces. The interface of flamerobin is shown in figure 2 on page 6.

Figure 2: Backing up a database with Flamerobin



Note that recent versions of Firebird ship with a second backup tool: `nbackup`, which allows to create incremental backups. It is based on a completely different principle than gbak: it creates an exact file copy of the database. To ensure that users can continue working while the database is being backed up, the database is put in a special mode: all writes to the database are performed in a temporary file, which is merged in the main database file when the backup is finished. Explaining the use of this tool is outside the scope of this introductory article, however.

5 Fixing broken databases

Despite all measures to prevent it, it can sometimes occur that a database file gets corrupted: a UPS may fail, the cleaning personnel may inadvertently switch off the server. In that case there are 2 options:

1. Restore the database from a recent backup. This will in most cases result in loss of data: all changes since the last backup was made.
2. Attempt to repair the database. This is the option to choose when there is no backup, or restoring the backup would lead to a to big loss of data.

In case a database gets corrupted, the `gfix` tool should be used to try and repair the database. The gfix tool can also be used to set certain database parameters. The options for repairing a database are:

-validate this will check the database for structural errors and will try to repair them.

-mend this will mark any corrupt records as invalid, so the next backup operation will skip them.

-full if this option is specified in combination with `-validate`, then the content of the data pages will also be validated.

-ignore when specified, any checksum errors will be ignored.

-no_update if this option is specified in combination with `-validate` then no repairs will be attempted, errors will only be reported.

In case a database is corrupted, the following steps should be taken:

- Stop the database server or make sure no-one can connect to the database any more. If possible, create a copy of the database file and work on this copy. (assume the copy is called thedb.fdb, and the owner of the database is 'Me')
- Validate the database:

```
gfix -user Me -password PWD -validate -full thedb.fdb
```

- If there were any errors on the database, they can be fixed with the `mend` option:

```
gfix -user Me -password PWD -mend -full thedb.fdb
```

- Create a backup of the mended database:

```
gbak -v -ig -user Me -password PWD thedb.fdb thebackup.fbk
```

Possibly also specify the `-g` option to disable garbage collection, a process which can also create problems.

- restore the database

```
gbak -v -ig -user Me -password PWD thebackup.fbk thenewdb.fdb
```

Under most circumstances, this should result in a fixed database, but will almost certainly result in loss of data.

There are other techniques that can be used to try and repair a corrupted database, the website

`http://www.ib-aid.com/`

gives a lot of useful tips for trying to repair broken databases, and also sells some tools to analyse databases. There is also a database repair service in case it is beyond the skills of the system administrator to repair the database.

In general, it is better to prevent than trying to repair: regular backups are a must.

6 Database tuning

Besides repairing broken databases, the gfix tool can also be used to tune some parameters on a database file. The main operations that should be performed from time to time are:

-shut shut down a database: after this, no-one can connect to the database except the owner and SYSBDA.

-online the reverse operation of shutdown: after this, users can again connect to the database.

-housekeeping n sets the automatic sweep interval, in number of transactions. By default this is 20000. Automatic sweep can be disabled with the value 0. When setting the interval to 0, a sweep should be done manually from time to time.

-sweep sweep the database: this command performs a garbage collection sweep. This is a very time-consuming operation.

-write op where `op` is one of `async` (the default) or `sync`: the latter enables forced writes: each data page is written to disk as soon as it is modifed (synchronous operation). The default is `async`: this causes writes to be buffered. The latter is faster, but will lead to bigger errors in case of a crash. The former is safer, but slower since there are a lot more disk operations that must be waited on.

With the exception of the shutdown and online commands these operations to some small degree influence the performance of the database: when an automatic sweep is in progress, the system becomes very unresponsive. Therefor it is often better to disable automatic sweep, and to do a manual sweep from time to time, when the database is not in use.

7 conclusion

In this article the basic tools for firebird server management have been presented: with all tools and techniques presented here, it is possible to operate a 24/7 firebird server. However, no amount tools will help in case a badly configured system still crashes: therefore it is of vital importance to have a regular backup of all databases (including the user database) and to test the integrity of these backups. It is up to the system administrator to set this up: The script presented here is just a first step in this process.