Firebird's nbackup tool

Paul Vinkenoog

21 October 2005 - Document version 0.1 - for Firebird 2.0 beta

Table of Contents

Introduction

nbackup is a new backup utility that comes with Firebird 2.0. It offers possibilities not present in *gbak* - Firebird's pre-existing backup tool - but doesn't replace the latter. Both programmes have their strengths and weaknesses; they will likely coexist for some time to come.

nbackup features - an overview

With nbackup, you can perform two different groups of tasks:

1. Making and restoring of both full and *incremental* backups. An *incremental backup* only contains the mutations since some specific previous backup.

2. Locking the main database file so you can subsequently back it up yourself with copying or backup tools of your own choice. In this mode, nbackup doesn't back up anything; it just creates the conditions under which you can safely make the backup yourself. There's a provision for restoring here, too.

Both modes can operate on an active database, without hindering connected users. The backup created will always reflect the state of the database *at the beginning of the operation*. In these respects nbackup doesn't differ from gbak.

Advantages of nbackup

- *Both modes:* high speed (as high as hardware and OS will allow), because nbackup doesn't look at the actual data. In backup mode the contents are written more or less blindly to the backup file.

- *Backup/restore mode:* time and disk space savings, because you don't need to make a full backup every time. This can make a huge difference with databases in the gigabyte range.

- *Lock/unlock mode:* total freedom in your choice of backup, copy, and/or compression tools.

Limitations of nbackup

- nbackup will not sweep and compact your database the way gbak does.

- You can't change the database owner with an nbackup backup/restore cycle, like you can with gbak.

- nbackup can't make *transportable backups*, that is: backups you can restore on an incompatible platform or under another server version.

- At this moment, nbackup should not be used on multifile databases.

- nbackup can only back up local databases.

We'll discuss nbackup's various functions extensively in the following sections.

Making and restoring backups

To begin with: `nbackup.exe` is located in the `bin` subdirectory of your Firebird folder. Typical locations are e.g. `C:\Program Files\Firebird\Firebird_2_0\bin` (Windows) or `/opt/firebird/bin` (Linux). Just like most of the tools that come with Firebird, nbackup has no graphical interface; you launch it from the command prompt (or from within a batch file or application).

Full backups

Making full backups

To make a full database backup, the command syntax is:

```
nbackup [-U <user> -P <password>] -B 0 <database> [<backupfile>]
```

For instance:

```
C:\Databases> nbackup -B 0 inventory.fdb inventory_1-Mar-2006.nbk
```

Comments:

- The parameter *-B* stands for backup (gee!). The *backup level* 0 indicates a full backup. Backup levels greater than 0 are used for incremental backups; we'll discuss those later on.
- Instead of a database filename you may also specify an alias.
- Instead of a backup filename you may also specify `stdout`. This will send the backup to standard output, from where you can redirect it to e.g. a tape archiver or a compression tool.
- The *-U* (user) and *-P* (password) parameters may be omitted:
  - if you're logged on as superuser (`root`, `Administrator`...), or
  - if the environment variables `ISC_USER` and `ISC_PASSWORD` are set.

  For clarity and brevity, these parameters are not used in the examples.
- The different parameters (*-B*, *-U* and *-P*) may occur in any order. Of course each parameter should be immediately followed by its own argument(s). In the case of *-B* there are three of them: backup level, database, and backup file - in that order!
- If the *-B* parameter comes last, you *may* leave out the name of the backup file. In that case nbackup will compose a filename based on the database name, the backup level, and the current date and time. This can lead to a name clash (and a failed backup) if two backup commands of the same level are issued in the same minute.

> **Warning**
>
> Do *not* use nbackup for multifile databases. This can lead to corruption and loss of data, despite the fact that nbackup will not complain about such a command.

A word on the inner workings

Note: What follows here is not necessary knowledge to use nbackup. It just gives a rough (and incomplete) image of what happens under the hood during execution of nbackup *-B*:

1. First of all, the main database file is locked by changing an internal state flag. From this moment on, any and all mutations in the database are written to a temporary file - the difference file or *delta file*.

2. Then the actual backup is made. This isn't a straight file copy; restoring must be done by nbackup as well.

3. Upon completion of the backup, the contents of the delta file are integrated with the main database file. After that, the database is unlocked (flag goes back to “normal”) and the delta is removed.

The functionality of steps 1 and 3 is provided by two new SQL statements: ALTER DATABASE BEGIN BACKUP and ALTER DATABASE END BACKUP. Contrary to what the names suggest, these statements do *not* take care of making the actual backup; rather, they create the conditions under which the main database file can be safely backed up. And to be clear: you don't need to issue these commands yourself; nbackup will do that for you, at the right moments.

Restoring a full backup

A full backup is restored as follows:

```
nbackup [-U <user> -P <password>] -R <database> [<backupfile>]
```

For instance:

```
C:\Databases> nbackup -R inventory.fdb inventory_1-Mar-2006.nbk
```

Comments:

- You don't specify a level for a restore.
- When restoring, the *-R* parameter *must* come last, for reasons that will become clear later.
- If the specified database exists already and there are no connections, it will be overwritten without any warning! If there are users connected, the restore fails and you get an error message.
- Here too, you may omit the name of the backup file. If you do, nbackup will prompt you for it. *However, at this point in the development of Firebird 2 - the alpha 3 stage - this leads to an error (at least under Windows) and a failed restore.*

Incremental backups

Making incremental backups

To make an incremental (“differential”) backup we specify a backup level greater than 0. An incremental backup of level *N* always contains the database mutations since the most recent level *N-1*

backup.

Examples:

One day after the full backup (level 0), you make one with level 1:

```
C:\Databases> nbackup -B 1 inventory.fdb inventory_2-Mar-2006.nbk
```

This backup will only contain the mutations of the last day.

One day later again, you make another one with level 1:

```
C:\Databases> nbackup -B 1 inventory.fdb inventory_3-Mar-2006.nbk
```

This one contains the mutations of the last *two* days, since the full backup, not only those since the previous level-1 backup.

A couple of hours on we go for a level-2 backup:

```
C:\Databases> nbackup -B 2 inventory.fdb inventory_3-Mar-2006_2.nbk
```

This youngest backup only contains the mutations since the most recent level-1 backup, that is: of the last few hours.

> **Note**
>
> All the comments that have been made about full backups also apply to incremental backups.

> **Warning**
>
> Again: do not use nbackup for multifile databases.

Restoring incremental backups

When restoring incremental backups you must specify the entire chain of backup files, from level 0 through the one you wish to restore. The database is always built up from the ground, step by step. (It is this stepwise adding until the database is restored that gave rise to the term *incremental backup*).

The formal syntax is:

```
nbackup [-U <user> -P <password>]
        -R <database> [<backup0> [<backup1> [...] ] ]
```

So restoring the level-2 backup from the previous example goes as follows:

```
C:\Databases> nbackup -R inventory.fdb inventory_1-Mar-2006.nbk
                 inventory_3-Mar-2006.nbk inventory_3-Mar-2006_2.nbk
```

Of course the line has been split here for layout reasons only - in reality you type the entire command and only hit **Enter** at the end.

Comments (in addition to the comments with restoring a full backup):

- Because it is not known beforehand how many filenames will follow the `-R` switch (as we don't specify a level when restoring), nbackup considers all arguments after the `-R` to be names of backup files. It is for this reason that no other parameter (`-U` or `-P`) may follow the list of filenames.

- There is no formal limit to the number of backup levels, but in practice it will rarely make sense to go beyond 3 or 4.

Non-connecting links

What happens if you accidentally leave out a file, or specify a series of files that don't all belong together? You could imagine that you specify `inventory_2-Mar-2006.nbk` by mistake instead of `inventory_3-Mar-2006.nbk` in the above example. Both are level-1 backup files, so in both cases we get a nice “0, 1, 2” level series. But our level-2 file is incremental to the level-1 backup of 3 March, not to the one of 2 March.

Fortunately such a mistake can never lead to an incorrectly restored database. Each backup file has its own unique ID. Furthermore, each backup file of level 1 or above contains the ID of the backup on which it is based. When restoring, nbackup checks these IDs; if somewhere in the chain the links don't connect, the operation is cancelled and you get an error message.

A practical application

An nbackup-based incremental backup scheme could look like this:

- Each month a full backup (level 0) is made;
- Each week a level-1;
- A level-2 backup daily;
- A level-3 backup hourly.

As long as all backups are preserved, you can restore the database to its state at any hour in the past. For each restore action, a maximum of four backup files is used. Of course you schedule things in such a way that the bigger, time-consuming backups are made during off-peak hours. In this case the levels 0 and 1 could be made at weekends, and level 2 at night.

If you don't want to keep everything for eternity, you can add a deletion schedule:

- Level-3 backups are deleted after 8 days;
- Level-2s after a month;
- Level-1s after six months;
- Full backups after two years, but the first one of each year is kept.

This is only an example of course. What's useful in an individual case depends on the application, the size of the database, its activity, etc.

Read on?

At this point you know everything you need in order to make and restore full and/or incremental backups with nbackup. You only need to read any further if you want to use backup tools of your own choice for your Firebird databases.

If you have no craving for that: good luck in you work with nbackup!

Locking and unlocking

If you prefer to use your own backup tools or just make a file copy, nbackup's lock-unlock mode comes into view. “Locking” means here that the main database file is frozen temporarily, not that there can't be no changes to the database. Just like in backup mode, mutations are directed to a temporary delta file; upon unlocking, the delta file is merged with the main file.

As a reminder: `nbackup.exe` lives in the `bin` subdir of your Firebird folder. Typical locations are e.g. `C:\Program Files\Firebird\Firebird_2_0\bin` (Windows) or `/opt/firebird/bin` (Linux). There's no GUI; you launch it from the command prompt (or from within a batch file or application).

Locking the database and backing up yourself

A typical session in which you make your own backup goes as follows:

1. Lock the database with the `-L` (lock) switch:

   ```
   nbackup [-U <user> -P <password>] -L <database>
   ```

2. Now copy/backup/zip the database file to your heart's content, with your own choice of tools. A simple file copy is also possible.

3. Unlock the database with `-N` (uNlock):

   ```
   nbackup [-U <user> -P <password>] -N <database>
   ```

The last command will also cause any mutations - which have been written to the delta file - to be merged into the main file.

The backup you made contains the data as they were at the moment the database was locked, regardless how long the locked state has lasted, and regardless how long you may have waited before making the actual backup.

> **Warning**
>
> What goes for backup/restore also applies to the lock/unlock switches: do not use them on multifile databases. Until things have changed, don't let nbackup loose on multifile databases at all!

Restoring a backup made after “nbackup -L”

An copy of a locked database is itself a locked database too, so you can't just copy it back and start using it. Should your original database get lost or damaged and the self-made copy needs to be restored (or should you wish to install the copy on another machine), proceed like this:

1. Copy/restore/unzip the backed-up database file yourself with the necessary tools.

2. Now unlock the database, *not* with the `-N` switch, but with `-F` (fixup):

```
nbackup -F <database>
```

Why are there two unlock switches, `-N` en `-F`?

- `-N` first sees that any changes made since the locking by `-L` are merged into the main database file. After that, the database goes back into normal read/write mode and the temporary file is deleted.
- `-F` only changes the state flag of the self-restored database to “normal”.

So you use:

- `-N` after having *made* a copy/backup yourself (to reverse the `-L` issued earlier);
- `-F` after having *restored* such a backup yourself.

> **Note**
>
> It is a bit unfortunate that the last switch should be called `-F` for Fixup. After all, it doesn't fix anything; it only *unlocks* the database. The `-N` (uNlock) flag on the other hand performs not only an unlock, but also a fixup (integration of mutations into the main file). But we'll have to live with that.

Under the hood

Note: This section doesn't contain any necessary knowledge, but provides some extra information which could deepen your understanding of the various switches.

nbackup `-L` does the following:

1. Connect to the database;
2. Start a transaction;
3. Call ALTER DATABASE BEGIN BACKUP (this statement has been discussed in the extra information on nbackup -B);
4. Commit the transaction;
5. Disconnect from the database.

nbackup `-N` follows the same steps, but with “...END BACKUP” in step 3.

nbackup `-F` works as follows:

1. The restored database file is opened;
2. Within the file, the state flag is changed from locked (`nbak_state_stalled`) to normal (`nbak_state_normal`);
3. The file is closed again.

> **Note**
>
> nbackup -F operates purely on file level and can therefore even be performed without a Firebird server running. Any `-U` or `-P` parameters added to the command will be completely ignored.

Document history

The exact file history is (or will be) recorded in the `manual` module in our CVS tree; see http://sourceforge.net/cvs/?group_id=9028

Revision History

0.1	21 October 2005	PV	First edition.

License notice