



Firebird Backup & Restore Utility

Norman Dunbar

02 November 2009 – Document version 1.2

Table of Contents

Introduction	3
Command-line Options	3
Backup Switches	5
Restore Switches	7
Backup Mode	8
Restore Mode	9
Restore Or Recreate?	10
Security Of Backups	10
Backup & Restore Recipies	11
Backup & Restore Prerequisites	11
A Simple Backup & Restore	11
Splitting The Backup	12
Change The ODS	12
Change The Cache Size	13
Change The Page Size	13
Create A Read-Only Database Clone	14
Backup & Restore With & Without Shadow Files.	14
Remote Backups & Restores	15
Using External Tools	16
Gbak Caveats	17
Normal Versus Privileged Users	17
Silent Running?	17
Appendix A: Document history	19
Appendix B: License notice	20

Introduction

Gbak is one of the database backup and restore utilities supplied with Firebird. In Firebird 1.5 it is the only supplied utility of this kind while Firebird 2.x also has the nbackup utility which is described in another document.

In this manual, we will discuss:

- Command-line options for gbak.
- gbak commands and their parameters.
- Running gbak in backup or restore modes.
- Some caveats, gotchas and foibles of gbak.

Command-line Options

Regardless of the mode that gbak is run in - backup or restore - there are a number of options that can be supplied on the command line. These are:

- **-u[ser] <username>**

Allows the username of the SYSDBA or database owner user to be specified if the database is to be backed up, or, in the case of a restore (with the **-c(reate)** switch specified), any valid username can be specified. This need not be supplied if `ISC_USER` environment variable exists and has a correct value for the username.

Databases can only be backed up by SYSDBA or the database owner. A restore can also be carried out by SYSDBA or the database owner, however, if the **-c(reate)** switch is used, *any* validated username can create a new database from a backup file.

- **-pas[sword] <password>**

Supplies the password for the username specified above. This need not be supplied if `ISC_PASSWORD` environment variable exists and has the correct value.

- **-ro[le] <SQL role name>**

Allows the specification of the role to be used by the connecting user. Not of much practical use and is not normally used in practice.

- **-z**

Displays the version number of the gbak utility and of the Firebird installation. You must supply a valid database name and backup filename - even though the backup will not actually be carried out - or the command will exit with an error code of 1 rather than zero.

```
tuxgt; gbak -z employee employee.fbk
gbak:gbak version LI-V2.1.3.18185 Firebird 2.1
gbak:   Version(s) for database employee
Firebird/linux (access method),version LI-V2.1.3.18185 Firebird 2.1
Firebird/linux (remote server),version LI-V2.1.3.18185 Firebird 2.1/tcp
(linux2)/P11
Firebird/linux (remote interface), version LI-V2.1.3.18185 Firebird 2.1/tcp
(linux2)/P11
on disk structure version 11.1
```

Note

The output above has been slightly abbreviated to allow it to fit the page width for a pdf.

- **-v(erify)**

Normally gbak operates quietly with no information written to the display. This switch reverses that situation and causes lots of information to be displayed. The default is to display the output to the screen, but you can redirect the output to a logfile using the **-y** switch.

- **-y <filename> or the text "suppress_output"**

Used in conjunction with the **-v(erify)** switch to redirect status messages to a file or device, rather than the screen, or to suppress them altogether. If **-y suppress_output** is used, then no information will be written to screen. This has the effect of turning off the **-v(erify)** switch. If a filename is given, the **-v(erify)** switch will remain on and the file will be written to.

Warning

It appears that the **-y suppress_output** switch doesn't do as advertised. What seems to happen is that a file called `suppress_output` is created and the output from gbak is written to it according to the **-v(erify)** switch settings.

However, if you use **-y suppress** instead, then the output is indeed, suppressed.

- **-help**

Help is actually not a valid option, but can be used to display the following screen of information:

```
gbak:legal switches are:
-B(ACKUP_DATABASE)      backup database to file
-BU(FFERS)              override page buffers default
-C(REATE_DATABASE)     create database from backup file
-CO(NVERT)             backup external files as tables
-E(XPAND)              no data compression
-FA(CTOR)              blocking factor
-G(ARBAGE_COLLECT)     inhibit garbage collection
-I(NACTIVE)            deactivate indexes during restore
-IG(NORE)              ignore bad checksums
-K(ILL)               restore without creating shadows
-L(IMBO)              ignore transactions in limbo
-M(ETA_DATA)          backup metadata only
-MO(DE) <access>      "read_only" or "read_write" access
-N(O_VALIDITY)        do not restore database validity conditions
-NOD(BTRIGGERS)       do not run database triggers
-NT                   Non-Transportable backup file format
-O(NE_AT_A_TIME)      restore one table at a time
-OL(D_DESCRIPTIONS)   save old style metadata descriptions
-P(AGE_SIZE)          override default page size
-PAS(SWORD)           Firebird password
-R(ECREATE_DATABASE)  [O(VERWRITE)] create (or replace if OVERWRITE used)
                      database from backup file
-REP(LACE_DATABASE)   replace database from backup file
-RO(LE)               Firebird SQL role
-SE(RVICE)            use services manager
-T(RANSPORTABLE)      transportable backup -- data in XDR format
-USE_(ALL_SPACE)      do not reserve space for record versions
-USER                 Firebird user name
```

```
-V(ERIFY)          report each action taken
-Y <path>          redirect/suppress status message output
-Z                print version number
```

The parentheses shown in the above indicates how much of the switch name you need to use in order to avoid ambiguity. Once you have specified the absolute minimum - the part before the opening '(' - you can use as much of what follows as you wish. For example, to use the **-B(ACKUP_DATABASE)** switch the minimum you must supply on the command line is **-B** but anything between **-B** and **-BACKUP_DATABASE** will be accepted.

Using the **-help** switch like this, or any other invalid switch, will cause gbak to exit with an error code of 1 on Linux and Windows.

Backup Switches

When carrying out a backup of a database, the following switches will be of use:

backup database to file

- **-B(ACKUP_DATABASE)**

This switch is required whenever you wish to take a backup of a database.

- **-CO(NVERT)**

This switch causes any tables, defined as external, to be backed up as if they were normal (non-external) tables. When this dump file is used to restore a database, the tables that were external in the original database will no longer be external.

- **-E(XPAND)**

Normally, gbak will compress the output file. This switch prevents that compression from taking place.

- **-FA(CTOR) <block size>**

If backing up to a physical tape device, this switch lets you specify the tape's blocking factor.

- **-G(ARBAGE_COLLECT)**

Normally gbak will ignore 'garbage' data when creating the backup. This is useful when you might want to reduce the size of your database by dumping it, then immediately, restoring it. (See the manual on gfix for details of what garbage entails.) The use of this switch prevents garbage collection from taking place and the resulting dump file is larger.

- **-IG(NORE)**

This switch causes gbak to ignore bad checksums in the database. This can be used to attempt to backup a database that failed due to checksum errors. There is no guarantee that the data will be usable though, so it is best to take other precautions to preserve your data.

- **-L(IMBO)**

If you have a two-phase transaction (across two different databases) that failed because a server died before the commit, but after the changes were made, you have a limbo transaction. This switch forces the backup

to ignore data from such broken transactions. It should not be used for normal backups and only used, like the **-IG(NORE)** switch to attempt to recover from a failure.

- **-M(ETA_DATA)**

This switch causes your data to be ignored and not backed up. Only the database metadata are backed up. The dump file will effectively contain a 'script' that will enable you to recreate a database containing only the database objects but no actual data.

- **-NT**

This switch turns off the **-T(RANSPORTABLE)** switch (which is on by default) and causes the dumpfile to be created using platform dependent formats. If you use this switch to create a backup then you can only restore the backup on a similar platform. You cannot, for example, take a dumpfile from Linux over to a Windows server.

- **-OL(D_DESCRIPTIONS)**

This switch is unlikely to be used. It has been deprecated. Its purpose is to force the backup to be made using the older InterBase versions' format of metadata descriptions.

- **-PAS(SWORD) <password>**

See above.

- **-RO(LE) <role name>**

See above.

- **-SE(RVICE) <servicename>**

This switch causes gbak to backup a remote database via the service manager. This causes the backup file to be created on the remote server, so the path format and filename must be valid on the remote server.

- **-T(RANSPORTABLE)**

The default dumpfile format is transportable. Transportable backup files are written in a format known as *external data representation* (XDR) format and it is this format which allows a dump taken on a server of one type to be successfully restored on a server of another type.

- **-USER <username>**

See above.

- **-V(ERIFY)**

See above.

- **-Y <full name of logfile> or the text "suppress_output"**

See above.

Restore Switches

When carrying out a restore or replacement of a database, the following switches will be of use:

- **-BU(FBERS) <number of buffers>**

This switch sets the default database cache size (in number of database pages) for the database being restored. If a database is being overwritten then this setting will overwrite the previous setting for the cache size.

- **-C(REATE_DATABASE)**

This switch causes a new database to be created from the backup file. The database file must not exist or the restore will fail. Either this switch or **-REP(LACE_DATABASE)** or **-R(ECREATE_DATABASE)** must be specified.

- **-I(NACTIVE)**

This switch can be used to restore a database when a previous restore attempt failed due to index errors. All indexes in the restored database will be inactive.

- **-K(ILL)**

This switch restores the database but doesn't recreate any shadow files that existed previously.

- **-MO(DE) <access>**

This switch allows the database being restored to be set to the given access mode when opened. By default, the mode is taken from the database that was dumped.

- **-N(O_VALIDITY)**

This switch is similar to the **-I(NACTIVE)** switch above, except, it disabled all CHECK constraints in the restored database.

- **-NOD(BTRIGGERS)**

New switch from Firebird 2.1 which prevents the *database triggers* from firing on a restore. Database triggers are a new feature in Firebird 2.0 onwards and are different from *table triggers* which will continue to fire.

- **-O(NE_AT_A_TIME)**

This switch causes the restore to restore one table at a time. This can be useful when a previous restore failed due to data errors.

- **-P(AGE_SIZE) <new page size>**

Use this switch to change the default database page size. By default, the database is restored using a page size the same as the one that was in use when the database was dumped.

- **-PAS(SWORD) <password>**

See above.

- **-R(ECREATE_DATABASE) [O(VERWRITE)]**

New from Firebird 2.x. Create (or replace if **O(VERWRITE)** is used) the named database from the backup file. The database filename should not already exist or an error will occur. This is not the case if the **O(VERWRITE)** option is also used.

This is a new switch and is deliberately abbreviated to **-r** to try to prevent unsuspecting DBAs from overwriting an existing database thinking that the **-r** was abbreviated from **-restore**. Now, it takes special effort to manage this especially as **-restore** was never actually a valid switch; **-r** was in fact an abbreviation of **-replace_database** and it did this by deleting the database file and then recreating it from the backup.

Using **-r(ecrate_database) o(verwrite)** is effectively the same as using **-rep(lace_database)**.

- **-REP(LACE_DATABASE)**

Replace database from backup file. This switch used to be abbreviated to **-r** in previous (to Firebird 2.x) versions. This switch will be removed in a version of Firebird later than 2.1.3 (where it still exists). You are advised to use the **-r(ecrate_database) o(verwrite)** switch instead.

- **-SE(RVICE) <servicename>**

Use the services manager on a remote database to restore a remote database.

- **-USE_(ALL_SPACE)**

This switch forces the restore to use 100% of each database page and thus not leave room for changes. By default, 80% of a page is used and 20% kept for changes. This switch is likely to be only of practical use where the database is created and used in read-only mode.

- **-USER <username>**

See above.

- **-V(ERIFY)**

See above.

- **-Y <name of logfile>**

See above.

Backup Mode

Before you consider using other tools to take a backup of your Firebird database, make sure that you know what the tools do and how a running database will be affected by them. For example, if you use Winzip to create a compressed copy of a database and you do it when users are accessing the system, the chances of a successful restore of that database are slim. You must either always use the **gbak/nbackup** tools which know how the database works, or, use **gfix** to shut the database down completely before you even attempt to backup the database file(s).

Gbak creates a consistent backup of the database by starting a transaction that spans the backup period. When the backup is complete, the transaction is ended and this means that the backup process can be run while users

are working in the database. However, any transactions started after the backup process begins will not have any of the changed data written to the backup file. The backup will represent a copy of the entire database at the moment the backup began.

The dump file created by a default gbak backup is cross platform (transportable) so a backup taken on a Windows server can be used to recreate the same database on a Linux server, or on any other platform supported by Firebird. This is not true of the copies of your database taken (while the database was closed!) with tools such as Winzip etc. Those copies should only ever be used to restore a database on the same platform as the one copied.

Important

Always backup the database with the version of gbak supplied with the running database server.

And one final thought on backups, regardless of the fact that the backup finished with no errors, exited with an error code of zero and all appears to be well, how do you actually know that the backup file created is usable? The short answer is, you don't. Whenever you have a valuable database - and they all should be - you are strongly advised to take your backup files and use them to create a test restore of a database either on the same server or even better, on a different one. Only by doing this can you be certain of a successful backup.

The following example shows a backup being taken on a server named linux and used to create a new clone of the database on another Linux server named tux to make sure that all was well. First of all, the backup on linux:

```
linux> gbak -backup -verify -y backup.log employee employee.fbk
linux> gzip -9 employee.fbk
```

Then, on the tux server:

```
tux> scp norman@linux:employee.fbk.gz ./
Using keyboard-interactive authentication.
Password:
employee.fbk.gz          |          19 kB |   19.3 kB/s | ETA: 00:00:00 | 100%
tux> gunzip employee.fbk.gz
tux> gbak -replace -verify -y restore.log employee.fbk employee.restore.test
```

At this point, the restore has worked and has overwritten the previous database known as employee.restore.test.

The actual location of the database for the database employee.restore.test is defined in the aliases.conf file in /opt/firebird on the server. In this test, it resolves to /opt/firebird/databases/employee.restore.fdb.

For further proof of reliability, the application may be tested against this clone of the live database to ensure all is well.

Restore Mode

Backups taken with the gbak application from one version of Firebird - or InterBase - can be used by later versions of Firebird to restore the database, however, while this may result in an upgrade to the On Disc Structure (ODS) for the database in question, the SQL Dialect will never be changed. If you backup an InterBase dialect 1 database and then use the dump file to recreate it under Firebird 2.1, for example, the ODS will be updated to 11.1 but the SQL Dialect will remain as 1.

Important

Always restore the database with the version of gbak supplied with the database server you wish to run the (new) database under. However, gbak from Firebird 2.1 can be used to restore a database onto any version of Firebird.

You can change the SQL Dialect using gfix.

Restore Or Recreate?

Should a database be restored or replaced? Restoring a database is the process by which you take the existing file and delete prior to replacing it on disc with a backup copy. Gbak does this when you specify the `-r(ecreate_database) o(verwrite)` switch or the `-rep(lace_database)` switch. What is the difference?

If a database exists on disc and you ask gbak to restore it using one of the two switches above, you might corrupt the database especially if the database is in use and has not been shut down using gfix. In addition, if you have only partially completed the restore of a database, and some users decide to see if they can login, the database may well be corrupted.

Finally, if the restore process discovers that the dump file is corrupt, the restore will fail and your previously working database will be gone forever.

It can be seen that restoring a database can be a fraught experience.

For best results, always recreate the database with a new name - a clone - and update the `aliases.conf` to reflect the new name. This way, your users will always refer to the database by the alias regardless of the actual filename on the server.

Security Of Backups

As you have seen above anyone, with a valid username and password, can restore a gbak database dump file provided that they are not overwriting an existing database. This means that your precious data can be stolen and used by nefarious characters on their own servers, to create a copy of your database and see what your sales figures, for example, are like.

To try and prevent this from happening, you are advised to take precautions. You should also try and prevent backups from being accidentally overwritten before they have expired. Some precautions you can take are:

- Always set the dumpfile to be read-only after the backup is complete. This helps prevent the file from being overwritten.
- Keep backups in a safe location on the server. Storing backups in a location with restricted access helps reduce the chances of your backup files 'escaping' into the wild.
- Keep tape copies of your backups very secure. A locked safe or off-site location with good security is advisable. The off-site location will also be of use after a total disaster as the backups are stored in a separate location to the server they are required on.
- Backup to a partition or disc that has encryption enabled.
- Make sure that only authorised staff have access to areas where backups are kept.
- Always test your backups by cloning a database from a recent backup.

In Firebird 2.1, there is an additional security feature built into `gbak` and all the other command-line utilities. This new feature automatically hides the password if it is supplied on the command line using the `-password` switch. `Gbak` replaces the password with spaces - one for each character in the password. This prevents other users on the system, who could run the `ps` command and view your command line and parameters, from viewing any supplied password. In this manner, unauthorised users are unable to obtain the supplied password.

```
tux> gbak -b -user SYSDBA -passw secret employee /backups/employee.fbk
```

```
tux> ps efx| grep -i gba[k]
20724 ... gbak -backup -user SYSDBA -passw          employee employee.fbk
... (lots more data here)
```

You can see from the above that the password doesn't show up under Firebird 2.1 as each character is replaced by a single space. This does mean that it is possible for someone to work out how *long* the password *could* be and that might be enough of a clue to a dedicated cracker. Knowing the length of the required password does make things a little easier, so for best results use a random number of spaces between `-passw` and the actual password. The more difficult you make things for the bad people on your network, the better.

Backup & Restore Recipes

The following recipes show examples of backup and restore tasks using `gbak`. These are probably the commonest cases that you will encounter as a DBA. All the examples use the `employee` database supplied with Firebird and the actual location is correctly configured in `aliases.conf`. Each of the following recipes is run with the assumption that the environment variables `ISC_USER` and `ISC_PASSWORD` have been given suitable values.

Backup & Restore Prerequisites

If you replace an open and running database, there is a good chance that you will corrupt it. For best results and minimal chance of corrupting a database, you should close it before replacing it. To close a database and force all current users out (not actually a nice thing to do) use `gfix` as follows:

```
gfix -shut -tran 60 employee
```

The example above prevents any new transaction from being started which prevents new queries being executed or new sessions connecting to the database. It will wait for up to 60 seconds for everyone to logout and for all current transactions to complete before shutting down the database. If any long running transactions have not completed by the end of 60 seconds, the shutdown will timeout and the database will remain open.

After the restore of the database has completed, the database will automatically be opened again for use.

A Simple Backup & Restore

```
tux> # Backup the database.
tux> gbak -backup employee /backups/employee.fbk

tux> # Restore the database.
tux> gfix -shut -tran 60 employee
```

```
tux> gbak -replace overwrite /backups/employee.fbk employee
```

Splitting The Backup

The `gsplit` filter application, documented in its own manual, doesn't actually work anymore. This filter was supplied with old versions of InterBase and Firebird to allow large database backups to be split over a number of files so that file system limits could be met. Such limits could be the size of a CD, the 2GB limit on file sizes on a DVD, some Unix file systems have a 2 GB limit and so on.

Gbak allows the dump files to be split into various sizes (with a minimum of 2048 bytes) and will only create files it needs.

```
tux> # Backup the database to multiple files.
tux> gbak -backup employee /backups/emp.a.fbk 600m /backups/emp.b.fbk 600m
```

The sizes after each filename indicate how large that particular file is allowed to be. The default size is bytes, but you can specify a suffix of k, m or g to use units of kilo, mega or gigabytes.

If the dump completes before writing to some files, then those files are not created. A dump file is only ever created when it must be.

The size of the final dump file will be quietly ignored if the database has grown too large to allow a truncated backup to complete. If, in the example above, the backup needs a total of 1500M, then the last file would be written to a final size of 900m rather than the 600m specified.

To restore such a multi-file backup requires that you specify all of the filenames in the dump and in the correct order. The following example shows the employee database above being restored from the two files dumped above:

```
tux> # Restore the database from multiple files.
tux> gfix -shut -tran 60 employee
tux> gbak -replace /backups/employee.a.fbk /backups/employee.b.fbk employee
```

Change The ODS

Normally the ODS used is the one in force by the version of Firebird used to restore the database. So, the examples above will actually change the ODS when the database is restored. The backup should be taken using the `gbak` utility supplied by the old ODS version of InterBase or Firebird. The restore should be carried out using `gbak` from the newer version of Firebird.

```
tux> setenv_firebird 2.0
Firebird environment set for version 2.0.

tux> # Check current ODS version (as root user!)
tux> gstat -h employee|grep ODS
      ODS version          11.0

tux> # Backup the (old) database.
tux> gbak -backup employee /backups/employee.2_0.fbk

tux> setenv_firebird 2.1
```

```
Firebird environment set for version 2.1.

tux> # Recreate the database and upgrade the ODS.
tux> gfix -shut -tran 60 employee
tux> gbak -replace overwrite /backups/employee.2_0.fbk employee

tux> # Check new ODS version (as root user!)
tux> gstat -h employee|grep ODS
      ODS version          11.1
```

After the above, the old 2.0 Firebird database will have been recreated - wiping out the old database - as a Firebird 2.1 database with the corresponding upgrade to the ODS from 11.0 to 11.1.

The script `setenv_firebird` is not supplied with Firebird and simply sets the path etc to use the correct version of Firebird as per the supplied parameter.

Change The Cache Size

The default database cache is created when the database is created, or subsequently by using `gfix`. `Gbak` can restore a database and reset the default cache size as well. The process is as follows:

```
tux> # Check current cache size (as root user!)
tux> gstat -h employee | grep -i buffer
      Page buffers          0

tux> # Restore the database & change the cache size.
tux> gfix -shut -tran 60 employee
tux> gbak -replace overwrite -buffer 200 /backups/employee.fbk employee

tux> # Check the new cache size (as root user!)
tux> gstat -h employee | grep -i buffer
      Page buffers          200
```

The default cache size is used when the number of buffers is zero, as in the first example above. `Gbak` allows this to be changed if desired. `Gbak`, however, cannot set the cache size back to zero. You must use `gfix` to do this.

Change The Page Size

Similar to the example above to change the default database cache size, the database page size can also be changed using `gbak`.

```
tux> # Check current page size (as root user!)
tux> gstat -h employee | grep -i "page size"
      Page size             4096

tux> # Restore the database & change the page size.
tux> gfix -shut -tran 60 employee
tux> gbak -replace overwrite -page_size 8192 /backups/employee.fbk employee

tux> # Check the new page size (as root user!)
tux> gstat -h employee | grep -i "page size"
      Page size             8192
```

Create A Read-Only Database Clone

Sometimes you do not want your reporting staff running intensive queries against your production database. To this end, you can quite easily create a clone of your production database on a daily basis, and make it read-only. This allows the reporting team to run as many intensive reports as they wish with no ill effects on the production database and it prevents them from inadvertently making changes.

The following example shows the production employee database running on Linux server tux, being cloned to the reporting team's Linux server named tuxrep. First on the production tux server:

```
tux> # Backup the production database.
tux> gbak -backup employee /backups/employee.fbk
```

Then on the reporting team's tuxrep server:

```
tuxrep> # Scp the dump file from tux.
tuxrep> scp fbuser@tux:/backups/employee.fbk ./
Using keyboard-interactive authentication.
Password:
employee.fbk          |          19 kB |  19.3 kB/s | ETA: 00:00:00 | 100%

tuxrep> # Restore the employee database as read-only.
tuxrep> gfix -shut -tran 60 employee
tuxrep> gbak -replace overwrite -mode read_only employee.fbk employee

tuxrep> # Check database mode (as root user)
tuxrep> gstat -h employee|grep -i attributes
Attributes              no reserve, read only
```

Backup & Restore With & Without Shadow Files.

Databases can have shadow files attached in normal use. Gbak happily backs up and restores those as well and in normal use, shadow files will be recreated. Should you wish to restore the database only and ignore the shadows, gbak can do that for you as the following example shows.

```
tux> # Check current shadows, use isql as gstat is broken.
tux> greenbird:/home/norman # isql employee

Database:  employee
SQL> show database;
Database:  employee
          Owner:  SYSDBA
  Shadow 1:  "/opt/firebird/shadows/employee.shd1" manual
  Shadow 2:  "/opt/firebird/shadows/employee.shd2" manual
  ...

SQL> quit;

tux> # Restore the database preserving shadow files.
tux> gfix -shut -tran 60 employee
tux> gbak -replace overwrite -kill /backups/employee.fbk employee
```

```
tux> # Check shadows again, use isql as gstat is broken.
tux> greenbird:/home/norman # isql employee

Database:  employee
SQL> show database;
Database:  employee
          Owner:  SYSDBA
  Shadow 1:  "/opt/firebird/shadows/employee.shd1" manual
  Shadow 2:  "/opt/firebird/shadows/employee.shd2" manual
  ...

SQL> quit;

tux> # Restore the database killing shadow files.
tux> gfix -shut -tran 60 employee
tux> gbak -replace overwrite -kill /backups/employee.fbk employee

tux> # Check shadows again, use isql as gstat is broken.
tux> greenbird:/home/norman # isql employee

Database:  employee
SQL> show database;
Database:  employee
          Owner:  SYSDBA
  ...

SQL> quit;
```

I use `isql` in the above examples as `gstat -h` seems to get confused about how many shadows there are on a database. It reports zero when there are two, eventually it catches up and reports that there are two, then, if you kill a shadow, it reports that there are now three!

Remote Backups & Restores

Firebird's `gbak` utility can make backups of a remote database. To do this, you need to connect to the service manager running on the database, this is normally called `service_mgr`. The following example shows the Firebird `employee` database on server `tuxrep` being backed up from the server `tux`. The backup will be written to the remote server, in other words, the backup file will be created on the `tuxrep` server and not on the `tux` one. The network protocol in use is TCP.

```
tux> # Backup the reporting database on remote server tuxrep.
tux> gbak -backup -service tuxrep:service_mgr employee /backups/remote_backup.fbk
```

The backup file will have the same owner and group as the Firebird database server - on Unix systems at least.

It is also possible to restore a remote database in this manner as well, and `gbak` allows this.

```
tux> # Restore the read-only reporting database on remote server tuxrep.
tux> gbak -replace -mode read_only -service tuxrep:service_mgr \
        /backups/remote_backup.fbk employee
```

The above example uses the handy Unix ability to split a long line over many shorter ones using a back slash as the final character on the line.

As ever, you are advised to beware of replacing a database in case there are problems during the restore. The example above recreates the existing database in read-only mode but this need not always be the case.

A remote backup can also be run on the database server itself! On Windows, this makes no difference, but on Unix systems, this local-remote method of backups and restores reduces network traffic. The 'remote' server, in this case, is not actually remote it is just the method of running the backup - connecting to the service manager - that implies remoteness.

```
tux> # Backup the employee database on this server, but pseudo-remotely!  
tux> gbak -backup -service tux:service_mgr employee /backups/remote_backup.fbk
```

And corresponding restores can also be run 'remotely':

```
tux> # Restore the employee database on this server, but pseudo-remotely!  
tux> gbak -replace -service tux:service_mgr /backups/remote_backup.fbk employee
```

The format of the parameter used for the `-service` switch is different according to the nature of the network protocol in use:

- TCP

When using TCP networks, the parameter separator is a colon, as in `-service server_name:service_mgr`.

- Named pipes

When using named pipes, the parameter requires two leading back slashes and the separator is another back slash, as in `-service \\server_name\service_mgr`.

Using External Tools

Gbak and nbackup are the best tools to use when backing up and/or restoring Firebird databases. They have been extensively tested and know the internals of the database and how it works, so the chances of these tools corrupting your valuable data are very slim. However, some DBAs still like to use external tools (those not supplied with Firebird) to make backups for whatever reason.

Because it is not possible for external tools to know where a database is to be found, given the alias name, the script writer and/or DBA must explicitly find out the correct location of the database file(s) and supply these to the external tool. To make this simpler for script writers, my own installation uses a standard in my `aliases.conf` file as follows:

- The database alias must start in column one.
- There must be a single space before the equals sign (=).
- The database filename must begin in column twenty.
- Double quotes around the database filename is not permitted - it doesn't work for the Firebird utilities either.
- Databases are all single file databases.

The last rule applies to my installation only and means that the following simple backup script will work. If multiple file databases were used, more coding would be required to take a backup using external tools.

```
tux> cat /opt/firebird/aliases.conf
# -----
# WARNING: Fbak standards require that:
#         The database name starts in column 1.
#         There is a single space before the equals sign.
#         The path must begin in column 20.
#         The path has no double quotes (they don't work!)
# -----
employee =          /opt/firebird/examples/empbuild/employee.fdb
```

The following shows the use of the `gzip` utility on a Linux server to take and compress a backup of a running database. The following is run as the root user due to the requirement to run `gfix` to shut down the database.

```
tux> # Backup the production database using gzip.
tux> gfix -shut -tran 60 employee
tux> DBFILE=`grep -i "^employee =" /opt/firebird/aliases.conf | cut -c20-`
tux> gzip -9 --stdout $DBFILE > /backups/employee.fdb.gz
```

The restore process for this database would be the reverse of the above. Again, the following runs as root.

```
tux> # Restore the production database from a gzip backup.
tux> gfix -shut -tran 60 employee
tux> DBFILE=`grep -i "^employee =" /opt/firebird/aliases.conf | cut -c20-`
tux> gunzip --stdout /backups/employee.fdb.gz > $DBFILE

tux> # Make sure firebird can see the file.
tux> chown firebird:firebird $DBFILE
```

Gbak Caveats

The following is a brief list of gotchas and funnies that I have detected in my own use of `gbak`. Some of these are mentioned above, others may not be. By collecting them all here in one place, you should be able to find out what's happening if you have problems.

Normal Versus Privileged Users

Only a `SYSDBA` or the owner of a database can take a backup of the database, however, *any* authenticated user can restore a database backup using the `-c[reate]` switch. This means that you must make sure you prevent your backup files from falling into the wrong hands because there is nothing then to stop unauthorised people from seeing your data by the simple process of restoring *your* backups onto *their* server.

The database restore will fail, of course, if the user carrying it out is not the database owner and a database with the same filename already exists.

Silent Running?

The `-y suppress_output` switch is supposed to cause all output to be suppressed. Similar in fact to running with `-v(erify)` not specified. However, all it seems to do is cause the output (according to the `-v(erify)`

switch setting) to be written to a file called `suppress_output`, however this only works once because the next run of `gbak` with `-y suppress_output` will fail because the file, `suppress_output`, already exists.

It is possible that this problem was introduced at version 2 for Firebird, because both 2.0 and 2.1 versions actually use the `-y suppress` switch rather than `-y suppress_output`. Using this (shorter) option does work as intended and the output is suppressed.

Appendix A: Document history

The exact file history is recorded in the manual module in our CVS tree; see http://sourceforge.net/cvs/?group_id=9028. The full URL of the CVS log for this file can be found at http://firebird.cvs.sourceforge.net/viewvc/firebird/manual/src/docs/firebirddocs/fbutil_gbak.xml?view=log

Revision History

1.0	10 October 2009	ND	Created as a chapter in the Command Line Utilities manual.
1.1	20 October 2009	ND	More minor updates and converted to a stand alone manual.
1.2	02 November 2009	ND	Copy and paste error detected and fixed. In the section on changing the page size.

Appendix B: License notice

The contents of this Documentation are subject to the Public Documentation License Version 1.0 (the “License”); you may only use this Documentation if you comply with the terms of this License. Copies of the License are available at <http://www.firebirdsql.org/pdfmanual/pdl.pdf> (PDF) and <http://www.firebirdsql.org/manual/pdl.html> (HTML).

The Original Documentation is titled *Firebird Backup & Restore Utility*.

The Initial Writer of the Original Documentation is: Norman Dunbar.

Copyright (C) 2009. All Rights Reserved. Initial Writer contact: NormanDunbar at users dot sourceforge dot net.