Firebird's gstat Database Statistics Reporting Tool

Norman Dunbar, Mark Rotteveel

Version 1.9, 24 February 2024

Table of Contents

Chapter 1. Introduction

`Gstat` is one of the database utilities supplied with Firebird. It is used to display statistical details about the contents of a database. `Gstat` is not transactionally aware, and some of the statistics it gathers may include data that has been deleted, for example, by normal database transactions.

In this manual, we will discuss:

- Command line options for `gstat`.
- `gstat` commands and their parameters.
- Running `gstat` and interpreting the results.
- Some caveats, gotchas and foibles of `gstat`.

For the header information, `gstat` reads the database file directly. For other statistics, `gstat` may connect to the database using the embedded engine or to a local Firebird server. If the database is currently opened by a Firebird server running in SuperServer (or `ThreadedDedicated`) mode, `gstat` may fall back to connecting using XNET or TCP/IP to the local server, but this may not always work when using a non-default configuration for the IPC name or TCP/IP port.

In that case, to use `gstat`, you may need to terminate any open connections, or use the equivalent commands via the service manager (e.g. through `fbsvcmgr`).

Chapter 2. Command-line Switches

Gstat should be run as either root or the Firebird user. This is because the default operating system permissions when a new database is created, are such that only the owner — firebird — has access to the database file(s). Even members of the firebird group have no read access by default.

`Gstat` is normally called as follows:

```
gstat database_name [switches]
```

The usage instructions shows that `gstat` can also be called as follows:

```
gstat [switches] database_name
```

In practice, switches can occur both before *and* after the database name. However, the `-T[ABLE]` switch can only be used with multiple table names if it is placed *after* the database name.

The database name cannot be a remote database, it must be local, but it can be an alias for a local database. The reason that it must be local is that obtaining the header information of `gstat` is done at the *physical file* level as opposed to making a database connection to the server — it reads the database file directly. Depending on the Firebird version and OS platform, other operations may connect to the local server or use the embedded engine to gather information.

If `gstat` is called with an invalid switch, or with `-?`, the following is displayed to remind you of the valid switches. Only the short form of the switches is displayed.

gstat Usage information for Firebird 5.0, with links to relevant sections

```
./gstat -?
usage:   gstat [options] <database> or gstat <database> [options]
Available switches:
    -a      analyze data and index pages
    -d      analyze data pages
    -e      analyze database encryption
    -h      analyze header page ONLY
    -i      analyze index leaf pages
    -s      analyze system relations in addition to user tables
    -u      username
    -p      password
    -fetch  fetch password from file
    -r      analyze average record and version length
    -t      tablename <tablename2...> (case sensitive)
    -role   SQL role name
    -tr     use trusted authentication
    -z      display version number
option -t accepts several table names only if used after <database>
```

The links are not present in the actual gstat output.

Firebird 1.0 and 1.5 also had a -l[og] switch, which would display information about log pages. Given Firebird does not use log pages, that switch was removed in Firebird 2.0.

The gstat usage instruction only displays abbreviated forms of switches. In the following sections, we will show the full switch names, with the optional part in square brackets ('[' and ']'). For example, for -A[LL], this means that you can enter -A, -AL and -ALL. Switch names are case-insensitive.

2.1. -A[LL]

This is the default switch and is equivalent to -d[ata] -i[ndex]. In the absence of -D[ATA] and/or -I[NDEX], or -H[EADER], gstat will run as if -a[ll] was specified.

2.2. -D[ATA]

Specifying this switch causes gstat to analyse all user tables of the specified database. Indices are not analysed unless -I[NDEX] is also specified.

When combined with -S[YSTEM], the system tables are analysed in addition to the user tables.

If -T[ABLE] is also specified, analysis is restricted to the specified tables.

This switch is implied by -A[LL].

2.3. -E[NCRYPTION]

Displays statistics on how many database pages are encrypted. This can, for example, be used to track the progress of encrypting or decrypting a database.

For example, getting encryption statistics on a non-encrypted employee database:

```
C:\Program Files\Firebird\Firebird5.0>gstat -u sysdba -e employee

Database "C:\Program Files\Firebird\Firebird5.0\examples\empbuild\employee.fdb"
Gstat execution time Fri Feb 23 12:36:37 2024

Database header page information:
[..]

Data pages: total 122, encrypted 0, non-crypted 122
Index pages: total 99, encrypted 0, non-crypted 99
Blob pages: total 0, encrypted 0, non-crypted 0
Generator pages: total 1, encrypted 0, non-crypted 1
Gstat completion time Fri Feb 23 12:36:37 2024
```

And on an encrypted database:

```
C:\Program Files\Firebird\Firebird5.0>gstat -u sysdba -e crypttest

Database "C:\DB\encrypteddb.fdb"
Gstat execution time Fri Feb 23 12:37:55 2024

Database header page information:
[..]

Data pages: total 82, encrypted 82, non-crypted 0
Index pages: total 60, encrypted 60, non-crypted 0
Blob pages: total 0, encrypted 0, non-crypted 0
Generator pages: total 1, encrypted 1, non-crypted 0
Gstat completion time Fri Feb 23 12:37:55 2024
```

This switch cannot be combined with most other switches, specifically it will report an error when combined with `-A[LL]`, `-D[ATA]`, `-H[EADER]`, `-I[NDEX]`, `-R[ECORD]`, `-S[YSTEM]`, or `-T[ABLE]`.

Introduced in Firebird 3.0.

2.4. `-H[EADER]`

This switch displays statistics about the database itself, or more specifically, information from the database header page, and then exits. The header information is also displayed when any other switch is used — so you always get database header details in your output.

Although other switches display the database header as well, it is not possible to combine `-h[eader]` with the other switches (except `-Z` and authentication switches).

The header information is read by direct access to the database file, and contrary to other switches, `-h[eader]` does not require authentication.

2.5. `-I[NDEX]`

Specifying this switch causes `gstat` to analyse all indices of user tables of the specified database. Tables are not analysed unless `-D[ATA]` is also specified.

When combined with `-S[YSTEM]`, the indices of system tables are analysed in addition to the indices of user tables.

If `-T[ABLE]` is also specified, analysis is restricted to indices of the specified tables.

This switch is implied by `-A[LL]`.

2.6. -N[OCREATION]

This switch excludes the “Creation date” entry from the database header output.

> Formally, this is an unsupported switch which could be removed or changed at any time. It exists to stabilize output for tests in the testsuite of Firebird.
>
> In other words, use at your own risk.

2.7. -S[YSTEM]

This switch is a modifier and alters the output from the `-D[ATA]` or `-I[NDEX]` switches by including the system tables (or indices of system tables) in addition to the user-defined tables (or indices). Using this switch on its own is equivalent to calling gstat with `-a[ll] -s[ystem]` specified.

When run, this switch lists statistics for the various `RDB$` tables and indices.

> In Firebird 2.1 only, this will also list information for the various monitoring tables (prefix `MON$`). However, as these are virtual tables, all statistics are 0, so this was removed from the output in Firebird 2.5.

2.8. -R[ECORD]

The `-r[ecord]` switch is a modifier for the `-D[ATA]` switche. It adds data about the average record and version lengths for any data tables (user and/or system) analysed. This switch has no effect on the `-I[NDEX]` switch.

2.9. -RO[LE]

Specifies the role for privileges — for example `RDB$ADMIN`, or another role providing the system privilege `USE_GSTAT_UTILITY` (and `IGNORE_DB_TRIGGERS`).

Syntax

```
-RO[LE] role_name
```

> Introduced in Firebird 3.0.

2.10. -T[ABLE]

This switch allows you to analyse a table, or list of tables, and any indices belonging to the specified tables.

Syntax when placed **before** *the database name*

```
-T[ABLE] table_name
```

Syntax when placed **after** *the database name*

```
-T[ABLE] table_name [table_name ...]
```

If you want to specify multiple tables, the `-t[able]` switch must be specified *after* the database name. See Gstat Caveats for some potential problems with this switch — especially before Firebird 2.5 — and an example of how it should be used.

The `-t[able]` switch should be followed by a list of the table names you wish to analyse. The list must match the table name exactly as stored in the metadata. In other words, case-insensitive table names (i.e. referenced with unquoted identifiers) must be entered in uppercase, while case-sensitive table names (referenced with quoted identifiers) must be entered exactly as-is. If a table name contains spaces or certain special symbols, you may need to enclose the table name in double or single quotes, and/or escape certain symbols, following the requirements of the shell/command prompt you use.

When combined with the `-D[ATA]` switch, only the data pages of the specified tables will be analysed. When combined with the `-I[NDEX]` switch, only the indices of the specified tables will be analysed. When neither `-d[ata]` or `-i[ndex]` is specified, both data pages and indices will be analysed, as if `-A[LL]` is specified. To obtain information on system tables, the `-S[YSTEM]` switch must be specified explicitly.

The database header information is also displayed.

2.11. -TR[USTED]

Use Windows trusted authentication (`Win_Sspi`).

Introduced in Firebird 3.0.

2.12. -U[SERNAME]

Allows the username of the SYSDBA, administrator user, or database owner user to be specified. This need not be supplied if the `ISC_USER` environment variable exists and has a correct value for the username, or if you are logged on to the server as a privileged account.

Syntax

```
-U[SERNAME] username
```

A privileged account is one of the following:

- root

- firebird
- interbase
- interbas (without the final 'e')

If you log in to the server with one of these accounts, you will automatically receive SYSDBA privileges. If you use a different account, you may be required to supply a username and password, and optionally a role, to run `gstat`.

2.13. -P[ASSWORD]

Supplies the password for the username specified above. This need not be supplied if the `ISC_PASSWORD` environment variable exists and has the correct value, or if you are logged on to the server using a privileged account.

Syntax

```
-P[ASSWORD] password
```

Providing a username and password is not necessary when only specifying `-H[EADER]`.

Since Firebird 3.0, providing a password is usually not necessary. In most cases, `gstat` commands other than `-h` will use the embedded engine to access the database, removing the need for a password.

When the database file is in use by a SuperServer instance, or the `gstat` executable is not part of a Firebird installation, and has no access to the Firebird embedded engine, it will connect to the local Firebird server using XNET or TCP/IP, in which case username and password (or another form of authentication) are required.

2.14. -FE[TCH_PASSWORD]

This switch causes the password to be read from a file as opposed to being specified on the command line.

Syntax

```
-FE[TCH_PASSWORD] { password_filename | stdin | /dev/tty }
```

The filename supplied must be readable by the user running `gstat`. If the filename is specified as `stdin`, then the user will be prompted for a password. On POSIX systems, the filename `/dev/tty` will also result in a prompt for the password.

Introduced in Firebird 2.5.

2.15. -Z

This is a modifier switch. Using `-z` displays the version number of the `gstat` utility and of the Firebird installation. If you don’t supply a valid database name and possibly another switch, `gstat` will print out the `gstat` version *and* an error. If a valid database is provided, `gstat` will also print out version information about the database engine, and — if applicable — client library and protocol versions.

The shortest output would be from a `-t non_existent_tablename` if all you need is the version details, as follows:

```
tux> gstat -t non_existing_tablename -z employee
gstat version LI-V2.1.3.18185 Firebird 2.1

Database "/opt/firebird/examples/empbuild/employee.fdb"
Database header page information:
...

Database file sequence:
File /opt/firebird/examples/empbuild/employee.fdb is the only file
        Firebird/linux Intel (access method), version
"LI-V2.1.3.18185 Firebird 2.1"
        Firebird/linux Intel (remote server), version
"LI-V2.1.3.18185 Firebird 2.1/tcp (greenbird)/P11"
        Firebird/linux Intel (remote interface), version
"LI-V2.1.3.18185 Firebird 2.1/tcp (greenbird)/P11"
        on disk structure version 11.1

Analyzing database pages ...
```

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

The output starts by displaying the `gstat` version, followed by the details of the database header. The database file and Firebird details are displayed next and finally, the details for the supplied table name, which of course is not found.

2.16. -?

This switch displays the usage information of `gstat`.

Introduced in Firebird 2.5.

Chapter 3. Gstat Examples And Interpretation

This section contains frequently executed statistics gatherings and explains the output.

3.1. Database Header

This option produces the least amount of output — unless you specify a single nonexistent table name with the `-T[ABLE]` switch — and is included with all other switches, so it is discussed first.

```
tux> gstat employee -header

Database "/opt/firebird/examples/empbuild/employee.fdb"
Database header page information:
        Flags                   0
        Checksum                12345
        Generation              184
        Page size               4096
        ODS version             11.1
        Oldest transaction      166
        Oldest active           167
        Oldest snapshot         167
        Next transaction        170
        Bumped transaction      1
        Sequence number         0
        Next attachment ID      68
        Implementation ID       19
        Shadow count            0
        Page buffers            0
        Next header page        0
        Database dialect        3
        Creation date           Sep 25, 2009 12:50:24
        Attributes              multi-user maintenance

    Variable header data:
        Sweep interval:         20000
        *END*
```

The first line of output displays the database filename(s) and path. This can be useful to resolve a database alias to find out exactly where the database is located. As the employee database is a single-file database, only one file is displayed. Had this been a multi-file database, the end of the listing above would look like the following:

```
...
    Variable header data:
        Continuation file:       /u00/firebird/databases/multi_employee.fdb1
        Last logical page:      162
```

The details of the various header fields are described below:

Flags

Flags are not used on a database header page.

Checksum

All checksums are 12345. Checksums on the various database pages are no longer used.

Generation

The generation number is incremented each and every time this page is rewritten in the database.

Page size

The page size of the entire database. As the database file has to be split into various pages, the SYSDBA or owner can, at creation time, specify how big a page size they desire. Every page in the database has the same size.

ODS version

The On-Disk Structure of a database defines, possibly along with the SQL dialect, which features of the Firebird database system are available to users of that database. These features may be present in the version of Firebird that you are running, but if the database ODS is older, some new features will not be available.

Values you may currently see here are:

- 5.0 for Interbase 3.3
- 8.0 for Interbase 4.0
- 9.0 for Interbase 4.5
- 9.1 for Interbase 5.0
- 10.0 for Firebird 1.0 and Interbase 6.0
- 10.1 for Firebird 1.5
- 11.0 for Firebird 2.0
- 11.1 for Firebird 2.1
- 11.2 for Firebird 2.5
- 12.0 for Firebird 3.0
- 13.0 for Firebird 4.0
- 13.1 for Firebird 5.0

Transaction details

There are a number of different transaction details in the report; these are:

Oldest transaction

The transaction ID of what is known as *Oldest Interesting Transaction* or OIT. This is simply the ID of the longest running transaction that has so far not been completed by way of a *hard* commit. It may have been rolled back, or be in limbo, but if it has been committed, it is no longer interesting.

This value, along with the Oldest Snapshot Transaction, is used to determine if an automatic sweep of the database is required.

> There are two types of commits — commit and commit retaining. Only the first of these is a hard commit, which, when executed, renders the transaction as no longer interesting. Commit retaining leaves the transaction as still interesting. Some database utilities and/or tools that commit actually perform a commit retaining which can leave your database with a lot of still interesting transactions.

Under normal circumstances, a rollback is converted to a commit by undoing all its changes and then marking the transaction as committed, but this is not always possible (e.g. too many changes, or the transaction was explicitly marked as “no auto-undo”).

Oldest active

The ID of the oldest *active* transaction, or OAT. This value shows the transaction ID (TID) of the oldest transaction that is still running. A transaction is considered active if it has not been *hard* committed, is not in a state of limbo, or has not been rolled back.

Oldest snapshot

The ID of the oldest transaction which is currently not eligible to be garbage-collected. Any transaction with this or a higher ID cannot, yet, have old record versions removed by a sweep, for example. Normally, this is the same as the OAT above. The difference between *this* value and the OIT, if greater than the database sweep interval — assuming that automatic sweeping is not disabled — determines if an automatic sweep takes place.

> Many websites, books, and manuals (previously including this one) explain that the automatic sweep is activated when OAT - OIT is greater than the sweep interval. This is *not* the case as explained by Vlad Khorsun, one of the Firebird developers, who explained that it is when OST - OIT is greater than the threshold that the sweep is activated.

Next transaction

The next transaction started on the database will have this ID number.

Bumped transaction

Always 1, no longer used, and removed from the output in Firebird 3.0

If you discover that the difference between the OAT and the Next Transaction ID is growing larger and larger, something in your database is not committing properly and as such, an increasing number of garbage records may be building up. Eventually, you will see that the database startup times take longer and longer and the performance becomes slower and slower.

Check the figures and if a problem is detected, you may be wise to run `gfix` to manually run a database sweep to clear out the garbage and restore normal working to the database.

You may wish to consult with the section entitled *Limbo Transaction Management* in the `gfix` manual for details on how to detect and treat transactions in limbo. These may well be affecting the ability of the database sweep process in clearing out old redundant data from older uninteresting transactions. Limbo transactions are caused when a two-phase commit across multiple databases, fails for some reason. Limbo transactions are still interesting to the database and need to be committed or rolled back using `gfix` as the sweep processing cannot tell whether it is safe to do so without human intervention.

Sequence number

Always zero. This was the sequence number of the database header page, but is no longer used.

Next attachment ID

The ID number of the next attachment to this database. Every time an application connects to the database, this number goes up by one. Starting up and shutting down the database increases this number too. `Gstat` also alters this ID, except for *only* the `-H[EADER]` option as that does not connect in a normal manner.

Implementation ID

When the database was created, it may have been created on a different system — hardware, operating system, etc. — to the one on which it is now running. The implementation ID shows you which hardware architecture the database was *originally* created on.

The implementation ID is used to determine if the database can actually be used on the hardware it is currently running on, or if there is some feature of the original hardware, where the database was created, that makes it incompatible with the current host system.

Shadow count

Displays the number of shadow files attached to this database, or available for use by this database. Sometimes this value is incorrect even when shadow files have been created and/or deleted recently.

Because of the inconsistency between what `gstat` reports and reality, it is best to use `isql` and the `SHOW DATABASE` command to view correct details of the shadow files.

Page buffers

If this value shows as zero, the database is using the server’s default value for the number of pages that can be cached in memory when the database is operating. The setting may be defined in the `firebird.conf` file. On Firebird Superserver 2.1, this setting is the `DefaultDbCachePages` in the configuration file and is set to 2048 pages. You may use `gfix` to change this without editing the configuration file.

Database dialect

The database’s SQL dialect number, normally 1 or 3. This setting can be changed using `gfix` and, alongside the ODS value, helps determine what features of Firebird are available for use when applications use the database.

Creation date

The date that this database was created originally. It may show the date that the database was last restored by `gbak`.

Attributes

This part of the report displays information about various attributes of the database. Examples of what you may see are:

no reserve

All pages will be filled to 100% and will be most useful on read-only databases. No space is reserved in each page for updates and/or deletions.

force write

Disk writes are not cached. They are written out to the hardware at the time of the write request. This is used mainly on Windows databases where the cache management system can lead to lost writes and database corruption.

shutdown

The database has been closed and cannot be used.

read only

The database is running in read-only mode.

multi-user maintenance

The database is closed for maintenance. Multiple connections are allowed by SYSDBA or the database owner only.

single-user maintenance

The database is closed for maintenance. Only one SYSDBA or database owner connection is allowed.

Other values may appear here, depending on the version of Firebird in use.

Variable header data

This part of the report covers information that is not in the fixed part of the database header. For example, the sweep interval is displayed here and information applicable to secondary files, if any, that are attached. If you have backed up the database using the `nbackup` tool, for example, details of the backup GUID will be displayed here — but only for the most recent backup.

3.2. Analyse Entire Database

The analysis of the entire database is the default for `gstat`. When used, all user tables and indices will be analysed and the gathered statistics reported. As the output will most likely be very large, it is advisable to pipe the output to a file:

```
gstat employee >employee.gst
```

The output will consist of an analysis of all user tables and all associated user indices. Interpretation of these results is covered below in the sections on analysis of data and index pages.

3.3. Analyse Data Pages Only

The command to analyse only user tables in the database is:

```
gstat employee -data >employee.gst
```

The output from this command lists the user tables in alphabetical order. No indices will be analysed or listed regardless of how many may exist within the database.

Once the report has been completed, the results can be analysed as follows, looking at one table in particular.

```
CONFIGREVISIONSTORE (213)
    Primary pointer page: 572, Index root page: 573
    Data pages: 2122, data page slots: 2122, average fill: 82%
    Fill distribution:
         0 - 19% = 1
        20 - 39% = 0
        40 - 59% = 0
        60 - 79% = 79
        80 - 99% = 2042
```

The extract, above, from the report begins by displaying the table name — CONFIGREVISIONSTORE — and the table ID — 213. The table's ID is actually the column RDB$RELATION_ID in the system table RDB$RELATIONS, as the following isql session shows:

```
SQL> select rdb$relation_name
CON> from rdb$relations
CON> where rdb$relation_id = 213;

RDB$RELATION_NAME
===================================
CONFIGREVISIONSTORE
```

Primary pointer page

This is the page number, within the database, of the first page with pointers to the data pages of this table. The structure of the database is such that each table has exclusive data pages and a list of those pages is required to be kept somewhere. This statistic gives you the page number for that location.

Index root page

This is the page number where the first page of pointers to the table's indices can be found within the database. Every table in the database has one page, the index root page, that holds pointers to the apex pages for each individual index.

Data pages

The total number of pages allocated to this table. Because gstat doesn't connect to the database in a transaction-aware manner, it cannot determine whether any of these pages are old record versions (garbage) or deleted records in currently uncommitted transactions, so the number may be higher than it needs to be as these additional pages are included in the total.

Data page slots

This value should be the same as the number of data pages. It reports on the number of pointers to pages in this table, that are stored in various pointer pages internal to the database. If the numbers differ, it may be down to the garbage that remains uncollected.

Average fill

The calculated space used in each page of the table, on average. The figure includes space utilised by back versions of records in the table. The fill distribution (below) gives more details.

Fill distribution

This section of the report displays a 5-band histogram where each band represents 20% of the space filled in each page. In the example above, we see that this table has a single page that is filled less than 20%, 79 pages are filled to between 60% and 79% while the vast majority, 2042, are filled to between 80% and 99%.

3.4. Analyse Index Pages Only

The command to analyse only user indices in the database is:

```
gstat employee -index >employee.gst
```

The output from this command lists the user tables in alphabetical order. No tables will be analysed; however, the report will list the table names in alphabetical order and will list all applicable indices beneath the appropriate table name.

Once the analysis has been completed, the results can be interpreted as follows. The following example shows the output from a single index in a database.

```
CONFIGREVISIONSTORE (213)
    Index PK_CONFIGREVISIONSTORE (0)
        Depth: 3, leaf buckets: 174, nodes: 62372
        Average data length: 2.58, total dup: 0, max dup: 0
        Fill distribution:
             0 - 19% = 15
            20 - 39% = 0
            40 - 59% = 55
            60 - 79% = 68
            80 - 99% = 36
```

The above extract from the report begins by displaying the table name — `CONFIGREVISIONSTORE` — and the table ID — 213 as described above.

Following the table's details — and only the name and ID are displayed — the index details are shown. As above, the index name and its ID are displayed. This time, the ID refers to the index's position in the list of all indices created on the table. ID zero is the first index created, ID 1 is the next and so on. The output from `gstat` may not list the indices in ID order and if any indices were created but subsequently dropped, there may be gaps in the ID sequence.

The next two lines, after the index name and ID, show the overall statistics for this index.

Depth

This statistic displays the number of pages that have to be accessed to get at an index entry. In this example we have to read three separate pages into the buffer cache before we can use the index details to access the row we want in the table. This is often referred to as index indirection.

```
Depth: 3
```

On disk, there is a top level *Index Root Page* which is created at the same time as the table. This page holds a list of pointers to the top (apex) page for each index — one page per index. For any given index, this page holds a list of pointers to either:

- intermediate level pages if depth is greater than 1, or,
- to the leaf pages for the actual index data if depth = 1.

The leaf pages store the location of the data that have been indexed. The index depth is the number of levels you have to step down from the index's apex page, to get to the leaf pages. Neither the Index Root Page nor the index's apex page are counted in the depth.

On average, a depth of 2 or less indicates an index that is efficient. If the depth is 3 or more, the index will most likely not be operating at its best. The solution in this situation is to use `gbak` to increase the database page size by taking a backup and restoring it, as follows:

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

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

tux> # Find current page size
tux> gstat employee -header | grep -i "page size"
     page size             4096

tux> # Restore database with a bigger page size
tux> gbak -recreate overwrite -page 8192 /backups/employee.fbk employee

tux> # Check new page size
tux gstat employee -header | grep -i "page size"
     page size              8192

tux> #Open the database
tux> gfix -online normal employee
```

Once the above has been carried out, you should find that the depth of the index is 2 or less. If this is not the case, simply repeat the process above using an even bigger page size.

The above command to restore the backup *overwrites* the original database file. This works by deleting the original file and recreating it, so you really need to be sure that your database backup actually works and that the backup file produced is usable *before* attempting to overwrite a database. See the gbak manual for more details.

Leaf buckets

This statistic informs us of the number of leaf pages that this particular index uses. A page and a bucket are synonymous but page tends to be the more modern term in wide use.

```
leaf buckets: 174
```

In our example index, we see that there are 174 pages in the database holding the details of the indexed values for this table — all of these pages contain pointers to the data.

The number of leaf pages should match up to the sum of the total number of pages in each histogram bar in the fill distribution, shown below.

Nodes

This is the total number of records in the table that have been indexed. However, it is possible — because gstat doesn't work in a transaction-aware manner — that this figure will possibly include rows that have been deleted (and not garbage-collected) and/or it may count records more than once if they have been modified in such a way that the indexed column(s) have been changed.

```
nodes: 62372
```

Because of the above, it is advisable to carry out a sweep, or a database backup and restore, prior to running gstat to ensure that the statistics gathered are accurate and reflect the true state of the database.

Average data length

This statistic indicates the average length of the key column(s) in bytes.

```
Average data length: 2.58
```

This is most likely less than the actual sum of the column sizes as Firebird uses index compression to reduce the amount of data held in an index leaf page.

Duplicates

Duplicates are not permitted in a primary key or unique index. Other indexes do permit duplicates and these statistics report on the number of duplicates the index holds. The following `isql` query shows the details of duplicates for an indexed column in a different table to the one being used so far — which has no duplicates.

```
SQL> SELECT IDX, COUNT(*)
CON> FROM NORMAN_TEST
CON> GROUP BY IDX;

         IDX        COUNT
============ ============
           1           10
           2            4
           3            1
```

From the above we see a total of 15 rows, of which there are 14 duplicated values (all those with a 1 or 2 in the IDX column). The following is the extract for the duplicates for this table:

```
Index NORMANX (0)
        Depth: 1, leaf buckets: 1, nodes: 15
        Average data length: 0.27, total dup: 12, max dup: 9
```

Total dup is the total number of duplicates in the index. Note from the above that only 12 duplicates are listed, but we already know that there are 14 duplicates rows in the index. How is this possible?

The first occurrence of a 1 and the first occurrence of a 2 are not counted, by `gstat`, as duplicates. Only the second and subsequent copies are considered duplicates.

In *my* opinion this is not quite correct behaviour. In the table above there are 15 rows and only three unique values in the IDX column, which is indexed. My index therefore holds 14 duplicate values rather than just 12.

You can, however, use the total dup value to extract the number of unique values in the index by subtracting it from the nodes value.

Max dup reports on the number of index entries which share the longest chain of duplicates. In other words — for the above index — there are 9 index entries that share the *same* value in the indexed column. We can see this to be true as the rows where IDX is 1 has 9 duplicate entries.

If max dup is getting close to total dup, then it is a reasonable assumption to conclude that it may be that the index is so poor in selectivity that it may never be used in queries.

Fill distribution

The remainder of the report for our original example index shows how the pages are used within the index.

```
Fill distribution:
             0 - 19% = 15
            20 - 39% = 0
            40 - 59% = 55
            60 - 79% = 68
            80 - 99% = 36
```

The figures represent a graph (or histogram) of how the space in the index's pages are being utilised. Each value of the histogram represents the number of pages in the whole index, which have been filled to a certain percentage. Each bar of the histogram represents the percentage filled for the page.

The example index's fill distribution is shown above and from these figures we see that the vast majority of the pages are filled to between 40 and 99%. The individual numbers at the end of each line above show the number of pages in this band. The example shows that:

- 15 pages have been filled to less than 20%; and
- 0 pages have been filled to between 20% and 39%; and
- 55 pages have been filled to between 40% and 59%; and
- 68 pages have been filled to between 60% and 79%; and
- 36 pages are filled to between 80% and 99%.

The sum of all these pages should add up to the same figure shown above for leaf nodes.

This index shows reasonably good space usage as the majority of pages are well filled. Ideally, you would like to see all the pages being filled to between 80 and 99%. If, on the other hand, the report showed that the pages were all lightly filled — say less than 60% — the index would be a good candidate for a rebuild exercise.

Be sure to consider the total number of nodes before starting a rebuild — if there are only a few nodes in the index, then rebuilding will not help the space usage as there may not be enough records to actually fill the index pages.

3.5. Selecting Tables To Analyse

To limit the analysis to a specific list of tables, rather than all user tables, you can use the `-T[ABLE]` switch to specify the ones you wish to analyse. Note that specifying table names in this manner also analyses all indices associated with those tables, unless only `-D[ATA]` or `-I[NDEX]` is specified.

```
gstat employee -t EMPLOYEE JOB COUNTRY >employee.gst
```

The resulting output is interpreted as described above.

If you have a table name that has been created by a user wishing to preserve the letter case of the table name, rather than having it converted to uppercase, for example:

```
tux> isql myMusic
Database:  mymusic

SQL> CREATE TABLE "MyMusic_Artists" (
CON> art_id integer,
CON> art_name ....);

SQL> COMMIT;
```

... then you must supply the table names in *exactly* the same letter case as the name of the table within the database:

```
gstat mymusic -t MyMusic_Titles MyMusic_Artists > MyMusic.gst
```

If you supply a non-existing table name, or get the name in the wrong case, etc., `gstat` will output an error and not analyse any table.

3.6. Including The System Tables & Indices

Normal use of `gstat` doesn't include the system tables and indices in the output. Calling `gstat` with the `-S[YSTEM]` switch causes these tables to be included in the analysis.

```
gstat employee -system >employee.gst
```

The interpretation of the results for the various system tables and indices is exactly as described above for user tables and indices.

3.7. Record & Version Details

When you run `gstat` with either the default switches, or `-D[ATA]` or `-T[ABLE]` and add the `-R[ECORD]` switch, you get additional information in the report that shows the average record length and average version details for the table(s) in question:

```
Average record length: 96.55, total records: 62372
    Average version length: 0.00, total versions: 0, max versions: 0
```

Average record length

Simply the average record length, in bytes, of all the records in the table. If this figure is 0.00 then you can be reasonably sure that all your records have been deleted, or that you have no records in the table.

Total records

The total number of records in the table. The value may include records in currently active transactions and may include records which have been deleted.

```
tux> # In session 1.
tux> gstat test -r -t NORMAN

...
Analyzing database pages ...
NORMAN (142)
    Primary pointer page: 268, Index root page: 269
    Average record length: 9.00, total records: 15
    Average version length: 0.00, total versions: 0, max versions: 0
    Data pages: 1, data page slots: 1, average fill: 10%

tux> isql tset -user norman -password secret
Database:  employee

SQL> SELECT COUNT(*) FROM NORMAN;

       COUNT
============
          15
```

At this point, we can see that there are 15 records in the `NORMAN` table and that the average length of these 15 records is 9.00 bytes. Next, we start another `isql` session and delete all the records from the `NORMAN` table.

```
tux> # In session 2.
tux> isql test -user norman -password secret
Database:  employee

SQL> DELETE FROM NORMAN;
SQL> COMMIT;
SQL> shell;
```

Still in the second session, we execute `gstat` to fetch statistics for the NORMAN table, the results are shown below.

```
tux> gstat test -r -t NORMAN

...
```

```
Analyzing database pages ...
NORMAN (142)
    Primary pointer page: 268, Index root page: 269
    Average record length: 0.00, total records: 15
    Average version length: 9.00, total versions: 15, max versions: 1
    Data pages: 1, data page slots: 1, average fill: 16%
...

tux> # Return to isql.
tux> exit
```

Comparing the report above with the one taken before we deleted the records, we can see straight away that:

- The average record length indicates that there are no records in the table, but the total record count shows that there are (still) 15. This is a good indicator that a session has deleted all the records but garbage collection has yet to run.
- The versioning details have all changed, there are now statistics for average version length, total versions and max versions.
- The average fill for the page(s) in this table has risen from 10% to 16% even though everything has been deleted. The extra space is being used by the back versions of the deleted records.

Continuing in the second session, if we execute a full table scan of the NORMAN table we will not see any results, but we will garbage collect the back versions.

```
SQL> SELECT * FROM NORMAN;

SQL> shell;

tux> gstat test -r -t NORMAN

...
Analyzing database pages ...
NORMAN (142)
    Primary pointer page: 268, Index root page: 269
    Average record length: 0.00, total records: 0
    Average version length: 0.00, total versions: 0, max versions: 0
    Data pages: 0, data page slots: 0, average fill: 0%
```

Everything has now returned to zero. There are no back versions, no current versions and the page is no longer filled.

Average version length

This is similar to the average record length, but for the back versions of the record. For example, if you have deleted a number of records and updated others, the old — back — versions of these records will be reported here. If the figure is 0.00 then garbage collection has taken place and removed the back versions — see above for an example.

Total versions

The same as total records above, but includes only the back versions. If the figure is 0 then garbage collection has taken place and removed the back versions — see above for an example.

Max versions

If a record has been updated many times, the max versions statistic shows you the number of back versions of the record (or records) in question. In a table where all the rows have been updated 7 times, but one has been updated 20 times, this statistic will report a value of 20. If the figure is 0.00 then garbage collection has taken place and removed the back versions — see above for an example.

3.8. If You Have Database Corruption

In the unlikely event of a database corruption, your `gstat` output may have the following within the report:

```
Database file sequence:
File /opt/firebird/examples/empbuild/corrupt.fdb is the only file

Analyzing database pages ...
    Expected b-tree bucket on page 337334 from 146314
```

If you ever see a message like the above, displayed just after the header information, you are advised to immediately shut down all connections to the database, make an operating system level copy of the database file(s) and attempt to run `gbak` against the database to take a full backup. Using `nbackup` may copy the database happily, but not report any errors. `Gbak`, on the other hand, will flag up errors.

Chapter 4. Gstat Caveats

The following is a brief list of gotchas and funnies that I have detected in my own use of `gstat`. 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.

4.1. The `-t[able]` Switch Can Cause Problems

The `-T[ABLE]` switch expects a list of table names to be supplied.

In older versions, if you supply the database name *after* a table name, it is assumed to be a table name, and you are prompted for a database name.

```
tux> gstat -t EMPLOYEE JOB employee
please retry, giving a database name
```

In Firebird 2.5 and higher, using `-t` *before* the database name accepts only one table name, and a second table name is interpreted as the database name, resulting in an error when the actual database name is encountered.

```
> gstat -u sysdba -t EMPLOYEE JOB employee
database name was already specified
```

For this reason, call `gstat` with the database name as the very *first* parameter, or at least put the `-t[able]` option *after* the database name:

```
tux> gstat employee -t EMPLOYEE JOB

Database "/opt/firebird/examples/empbuild/employee.fdb"
Database header page information:
...

Database file sequence:
File /opt/firebird/examples/empbuild/employee.fdb is the only file

Analyzing database pages ...
...
```

In Firebird 2.1 and earlier, you can supply an additional switch *after* the last table name and *before* the database name. This trick no longer works in Firebird 2.5 and higher, as it will only accept one table name when *before* the database name.

```
tux> gstat -t EMPLOYEE JOB -z employee
gstat version LI-V2.1.3.18185 Firebird 2.1
```

```
Database "/opt/firebird/examples/empbuild/employee.fdb"
Database header page information:
...

Database file sequence:
File /opt/firebird/examples/empbuild/employee.fdb is the only file
        Firebird/linux Intel (access method), version
"LI-V2.1.3.18185 Firebird 2.1"
        Firebird/linux Intel (remote server), version
"LI-V2.1.3.18185 Firebird 2.1/tcp (greenbird)/P11"
        Firebird/linux Intel (remote interface), version
"LI-V2.1.3.18185 Firebird 2.1/tcp (greenbird)/P11"
        on disk structure version 11.1

Analyzing database pages ...
```

4.2. The Shadow Count Seems Wrong

It appears that adding and/or dropping shadow files from a database is not always reported by gstat when it produces a database report.

```
tux> # Use gstat to display shadow details
tux> gstat employee -h|grep -i sh[a]dow
        Shadow count            0

tux> isql employee
Database: employee

SQL> SHOW DATABASE;
Database: employee
        Owner: SYSDBA
 Shadow 1: "/u00/firebird/databases/employee.shd1" auto
...
```

Straight away, it is obvious that the report from gstat is incorrect as the employee database has one shadow file. If we use isql to add a new shadow file to this database, as shown below, gstat still insists that there are no shadows.

```
SQL> CREATE SHADOW 7 AUTO '/u00/firebird/databases/employee.shd7';

SQL> SHOW DATABASE;
Database: employee
        Owner: SYSDBA
 Shadow 1: "/u00/firebird/databases/employee.shd1" auto
 Shadow 7: "/u00/firebird/databases/employee.shd7" auto
...

SQL> shell;
```

```
tux> gstat employee -h | grep -i sh[a]dow
        Shadow count            0
```

Appendix A: Document history

The exact file history is recorded in the firebird-documentation git repository; see https://github.com/FirebirdSQL/firebird-documentation

Revision History

1.9	24 Feb 2024	MR	Fixed broken cross-references caused by using prefix `gbak` instead of `gstat`
1.8	24 Feb 2024	MR	• Misc. copy editing • Add links from usage instruction to relevant sections • Include `gstat` name in the document title, making it the same as on https://firebirdsql.org/en/reference-manuals/
1.7	23 Feb 2024	MR	• Reordered document history so most recent changes are on the top • Convert commandline options from definition list to sections • Add switches: `-?`, `-encryption`, `-nocreation`, `-role`, and `-trusted` • Misc. copy editing, and updating information for newer version • Add some links to gfix and gbak documentation • Cross-links between sections • Added word-joiner in commandline switches between *minus* (-) and first character to ensure they aren't broken up on word wrap
1.6	19 Jun 2020	MR	Conversion to AsciiDoc, minor copy-editing
1.5	11 Oct 2011	ND	• Updated for Firebird 2.5. • Spelling errors corrected.
1.4	23 Mar 2011	ND	• Added ODS 9.1 for Interbase 5.0 to the list of known ODS values. • Added reference to Managing Limbo Transactions in the gfix manual. • Corrected explanation of when an automatic database sweep is carried out, based on OIT and OST as opposed to OIT and OAT. As advised by Vlad Khorsun.
1.3	17 Feb 2010	ND	Formatting errors in the command line switches corrected.
1.2	14 Dec 2009	ND	A couple more minor corrections and spelling mistakes corrected.
1.1	30 Nov 2009	ND	Many corrections suggested by Paul Vinkenoog plus a general tidy up and a few more examples added.
1.0	29 Oct 2009	ND	Created a new gstat manual.

Appendix B: License notice