Contents

Audience ................................................................................. 1
Performance Considerations ................................................ 3
Managing System Resources ............................................... 5
  Optimize Memory Use .................................................... 5
  Paging Increases Available Memory ...................... 5
  Utilities to Monitor Swapping ........................................ 6
  Server Memory .............................................................. 6
  Manage Buffer Caches ................................................. 7
  Determine the Sizes of the Buffer Caches ............. 7
  Set the Buffer Cache Sizes ...................................... 10
  Specify the Page Size ........................................... 11
  Optimize for Large Numbers of Users ....................... 11
  Platform-Specific Memory Options ...................... 13
The Process Threading Model ........................................... 16
Balancing I/O ................................................................. 17
  Raw I/O (on UNIX Operating Systems) ..................... 17
  Sybase IQ and Disk Striping ................................ 18
  Internal Striping ....................................................... 19
  Strategic File Locations ..................................... 19
Options for Tuning Resource Use .................................. 21
  Restricting Concurrent Queries ................................ 21
  Setting the Number of CPUs Available ............... 22
  Limiting Temporary dbspace Use By a Query ....... 22
  Limiting Queries by Rows Returned .................... 23
  Forcing Cursors to be Non-Scrolling ................. 24
  Limiting the Number of Cursors ....................... 25
  Limiting the Number of Statements .................. 25
  Prefetching Cache Pages .................................... 26
  Optimizing for Typical Usage ................................ 26
  Controlling the Number of Prefetched Rows ...... 27
Other Ways to Improve Resource Use ....................... 27
Contents

Managing Disk Space in Multiplex Databases .....27
Managing Multiplex Resources Using Logical
Servers ............................................................28
Load Balancing Among Query Servers ...........28
Managing Database Size and Structure ..........29
Network Performance .......................................30

Monitoring and Tuning Performance ..................33
Getting Information Using Stored Procedures ....33
Profiling Database Procedures .........................34
Viewing Procedure Profiling Statistics ..........34
Database Object Profiles .................................35
Procedure Profiling Statistics .........................36

Data Model Recommendations .........................39
Indexing Tips ................................................39
When and Where to use Indexes ...................40
Simple Index Selection Criteria .....................41
HG Index Loads ............................................42
Multi-Column Indexes ................................44
Join Column ...............................................45
Primary Keys ..............................................46
Foreign Keys ..............................................46
Proper Data Type Sizing ................................47
IQ UNIQUE and MINIMIZE_STORAGE ............48
Null Values ...............................................49
Unsigned Data Types ..................................49
LONG VARCHAR and LONG VARBINARY ........50
Large Object Storage ....................................51
Temporary Tables .......................................52
Denormalizing for Performance ....................53
UNION ALL Views for Faster Loads ................54

Monitoring Performance Statistics ..................56
Monitoring Performance at the Server Level ......56
Memory Usage Statistics ...............................57
Cache Statistics .........................................58
CPU Usage Statistics ..................................59
Contents

Optimizing Delete Operations ...........................................96
HG Delete Operations .................................................96
WD Delete Operations .................................................97
TEXT Delete Operations .............................................98

Index ................................................................................101
Audience

This document is intended for database administrators, database designers, and developers who want to configure Sybase® IQ for improved performance.
Performance Considerations

Performance is usually measured in response time and throughput. A good design and indexing strategy leads to the largest performance gains.

**Response Time**
Response time is the time it takes for a single task to complete. Several factors affect response time:

- Reducing contention and wait times, particularly disk I/O wait times
- Using faster components
- Reducing the amount of time the resources are needed (increasing concurrency)

**Throughput**
Throughput refers to the volume of work completed in a fixed time period. Throughput is commonly measured in transactions per second (tps), but can be measured per minute, per hour, per day, and so on.

**Design Considerations**
To realize the largest performance gains run Sybase IQ on a correctly configured system, establish a good design, and choose the correct indexing strategy.

Other considerations, such as hardware and network analysis, can locate bottlenecks in your installation.
Managing System Resources

Tuning your hardware and software configuration provides better performance and faster queries.

Optimize Memory Use

Understanding how Sybase IQ allocates memory can help you get the best performance from your system.

Paging Increases Available Memory

Although paging increases the amount of available memory, avoid or minimize page swapping for good memory management.

When there is not enough memory on your system, performance can degrade severely. If this is the case, you need to find a way to make more memory available. The more memory you can allocate to Sybase IQ, the better.

However, there is always a fixed limit to the amount of memory in a system, so sometimes operating systems can have only part of the data in memory and the rest on disk. When the operating system must go out to disk and retrieve any data before a memory request can be satisfied, it is called paging or swapping. The primary objective of good memory management is to avoid or minimize paging or swapping.

The most frequently used operating system files are swap files. When memory is exhausted, the operating system swaps pages of memory to disk to make room for new data. When the pages that were swapped are called again, other pages are swapped, and the required memory pages are brought back. This is very time-consuming for users with high disk usage rates. In general, try to organize memory to avoid swapping and, thus, to minimize use of operating system files.

To make the maximum use of your physical memory, Sybase IQ uses buffer caches for all reads and writes to your databases.

Note: Your swap space on disk must be at least large enough to accommodate all of your physical memory. Having swap/paging space striped across fast disks is essential.

See also

- Utilities to Monitor Swapping on page 6
- Server Memory on page 6
- Manage Buffer Caches on page 7
- Determine the Sizes of the Buffer Caches on page 7
- Set the Buffer Cache Sizes on page 10
Managing System Resources

- Specify the Page Size on page 11
- Optimize for Large Numbers of Users on page 11
- Platform-Specific Memory Options on page 13

Utilities to Monitor Swapping

Use the utilities on your operating system to find out if your system is paging excessively.

Use the UNIX `vmstat` command, the UNIX `sar` command, or the Windows Task Manager, to get statistics on the number of running processes and the number of page-outs and swaps. Use this information to find out if the system is paging excessively, then make any necessary adjustments. You may want to put your swap files on special fast disks.

See also
- Paging Increases Available Memory on page 5
- Server Memory on page 6
- Manage Buffer Caches on page 7
- Determine the Sizes of the Buffer Caches on page 7
- Set the Buffer Cache Sizes on page 10
- Specify the Page Size on page 11
- Optimize for Large Numbers of Users on page 11
- Platform-Specific Memory Options on page 13

Server Memory

Sybase IQ allocates heap memory for buffers, transactions, databases, and servers. Shared memory may also be used, but in much smaller quantities.

At the operating system level, Sybase IQ server memory consists of heap memory. For the most part, you do not need to be concerned with whether memory used by Sybase IQ is heap memory or shared memory. All memory allocation is handled automatically. However, you may need to make sure that your operating system kernel is correctly configured to use shared memory before you run .Sybase IQ.

Managing Multiplex Memory

Each server in the multiplex can be on its own host or share a host with other servers. Two or more servers on the same system consume no more CPU time than a single combined server handling the same workload, but separate servers might need more physical memory than a single combined server, because the memory used by each server is not shared by any other server.

See also
- Paging Increases Available Memory on page 5
- Utilities to Monitor Swapping on page 6
- Manage Buffer Caches on page 7
Manage Buffer Caches

Default cache sizes (16MB for the main and 12MB for the temporary cache) are too low for most databases. Allocate as much memory as possible to the IQ main and temporary buffer caches.

Sybase IQ needs more memory for buffer caches than for any other purpose. Sybase IQ has two buffer caches, one for the IQ store and one for the temporary store. It uses these two buffer caches for all database I/O operations—for paging, for insertions into the database, and for backup and restore. Data is stored in one of the two caches whenever it is in memory. All user connections share these buffer caches. Sybase IQ keeps track of which data is associated with each connection.

See also
• Paging Increases Available Memory on page 5
• Utilities to Monitor Swapping on page 6
• Server Memory on page 6
• Determine the Sizes of the Buffer Caches on page 7
• Set the Buffer Cache Sizes on page 10
• Specify the Page Size on page 11
• Optimize for Large Numbers of Users on page 11
• Platform-Specific Memory Options on page 13

Determine the Sizes of the Buffer Caches

Calculating the correct buffer cache size depends on several factors.

• The total amount of physical memory on your system
• How much of this memory Sybase IQ, the operating system, and other applications need to do their tasks
• Whether you are doing loads, queries, or both
• The schema configuration and query workload

See also
• Paging Increases Available Memory on page 5
• Utilities to Monitor Swapping on page 6
• Server Memory on page 6
• Manage Buffer Caches on page 7
Managing System Resources

- Set the Buffer Cache Sizes on page 10
- Specify the Page Size on page 11
- Optimize for Large Numbers of Users on page 11
- Platform-Specific Memory Options on page 13

Operating System and Other Applications
Most operating systems use a large percent of available memory for file system buffering. Understand the buffering policies for your operating system to avoid over-allocating memory.

See your application and operating system documentation for memory requirements for applications that run in conjunction with Sybase IQ.

See also
- Memory Overhead on page 8
- Main and Temp Buffer Caches on page 9

Memory Overhead
After you determine how much physical memory the operating system and other applications require, calculate how much of the remaining memory is required by Sybase IQ.

Raw Partitions Versus File Systems
For UNIX systems, databases using file systems rather than raw partitions may require another 30% of the remaining memory to handle file buffering by the operating system. On Windows, file system caching should be disabled by setting

```
OS_FILE_CACHE_BUFFERING = ‘OFF’
```

(the default for new databases).

Multiuser Database Access
For multiuser queries of a database, Sybase IQ needs about 10MB per “active” user. Active users are defined as users who simultaneously access or query the database. For example, 30 users may be connected to Sybase IQ, but only 10 or so may be actively using a database at any one time.

Memory for Thread Stacks
Processing threads require a small amount of memory. The more Sybase IQ processing threads you use, the more memory needed. The `-iqmt` server switch controls the number of threads for Sybase IQ. The `-iqtss` and `-gss` server switches control the amount of stack memory allocated for each thread. The total memory allocated for IQ stacks is roughly equal to:

```
(-gn * (-gss + -iqtss)) + (-iqmt * -iqtss )
```

If you have a large number of users, the memory needed for processing threads increases. The `-gn` switch controls the number of tasks (both user and system requests) that the database server can execute concurrently. The `-gss` switch controls—in part—the stack size for server execution threads that execute these tasks. IQ calculates the stack size of these worker threads using the following formula:

```
(-gss + -iqtss)
```
The total number of threads (-iqmt plus -gn) must not exceed the number allowed for your platform.

Other Memory Use
All commands and transactions use some memory. The following operations are the most significant memory users in addition to those discussed previously:

- Backup. The amount of virtual memory used for backup is a function of the IQ PAGE SIZE specified when the database was created. It is approximately \(2 \times \text{number of CPUs} \times 20 \times (\frac{\text{IQ PAGE SIZE}}{16})\). On some platforms you may be able to improve backup performance by adjusting BLOCK FACTOR in the BACKUP command, but increasing BLOCK FACTOR also increases the amount of memory used.

- Database validation and repair. When you check an entire database, the \texttt{sp_iqcheckdb} procedure opens all Sybase IQ tables, their respective fields, and indexes before initiating any processing. Depending on the number of Sybase IQ tables and the cumulative number of columns and indexes in those tables, \texttt{sp_iqcheckdb} may require very little or a large amount of virtual memory. To limit the amount of memory needed, use the \texttt{sp_iqcheckdb} options to check or repair a single index or table.

- Dropping leaked blocks. The drop leaks operation also needs to open all Sybase IQ tables, files, and indexes, so it uses as much virtual memory as \texttt{sp_iqcheckdb} uses when checking an entire database. It uses the Sybase IQ temp buffer cache to keep track of blocks used.

See also
- Operating System and Other Applications on page 8
- Main and Temp Buffer Caches on page 9

Main and Temp Buffer Caches
A general guideline for cache sizes is 40% for the main buffer cache and 60% for the temp buffer cache. Start with this guideline, monitor server performance, then adjust the cache size as necessary.

Buffer Caches and Physical Memory
The total memory used for Sybase IQ main and temporary buffer caches, plus Sybase IQ memory overhead, and memory used for the operating system and other applications, must not exceed the physical memory on your system.

For optimal performance, allocate as much memory as possible to the IQ main and temporary buffer caches. For example, if you have 4GB of physical memory on your machine available to Sybase IQ, you can split that amount between the main and temporary shared buffer caches.

Other Considerations
Buffer cache size requirements depend on use. For maximum performance, change the settings between inserting, querying the database, and mixed use. In a mixed-use environment, however, it is not always feasible to require all users to exit the database so that
you can reset buffer cache options. In those cases, you may need to favor either load or query performance.

**Note:**

- These guidelines assume you have one active database on your system at a time. If you have more than one active database, you need to further split the remaining memory among the databases you expect to use.
- On some UNIX platforms, you may need to set other server switches to make more memory available for buffer caches.

---

**See also**

- *Operating System and Other Applications* on page 8
- *Memory Overhead* on page 8

---

**Set the Buffer Cache Sizes**

Sybase IQ initially sets the size of the main and temporary buffer caches to 16MB and 12MB respectively. Change the default size of the main and temporary buffer caches to accommodate your applications.

<table>
<thead>
<tr>
<th>Method</th>
<th>When to use it</th>
<th>How long the setting is effective</th>
</tr>
</thead>
<tbody>
<tr>
<td>-iqmc and -iqtc</td>
<td>Recommended method. Sets cache sizes at startup.</td>
<td>From the time the server is started until it is stopped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The -iqmc and -iqtc server startup options only remain in effect while the server is running, so you need to include them every time you restart the server.</td>
</tr>
</tbody>
</table>

---

**See also**

- *Paging Increases Available Memory* on page 5
- *Utilities to Monitor Swapping* on page 6
- *Server Memory* on page 6
- *Manage Buffer Caches* on page 7
- *Determine the Sizes of the Buffer Caches* on page 7
- *Specify the Page Size* on page 11
- *Optimize for Large Numbers of Users* on page 11
- *Platform-Specific Memory Options* on page 13
Specify the Page Size

Page size and buffer cache size affect memory use and disk I/O throughput for the database.

**Note:** The page size cannot be changed and determines the upper size limit on some database objects and whether LOB features can be used.

**Page Size**

Sybase IQ performs I/O in units of page size. When you create a database, you specify a separate page size for the catalog store and the IQ store. The temporary store has the same page size as the IQ store.

Page size for the catalog store has no real impact on performance. The default value of 4096 bytes should be adequate. The IQ page size determines two other performance factors, the default I/O transfer block size, and the maximum data compression for your database.

**Data Compression**

Sybase IQ compresses all data. The amount of compression is determined on the IQ page size.

**Saving Memory**

If your machine does not have enough memory, increase the memory and decrease the buffer cache sizes. Decreasing the buffer caches too much, however, can make your data loads or queries inefficient or incomplete due to insufficient buffers.

**See also**

- *Paging Increases Available Memory* on page 5
- *Utilities to Monitor Swapping* on page 6
- *Server Memory* on page 6
- *Manage Buffer Caches* on page 7
- *Determine the Sizes of the Buffer Caches* on page 7
- *Set the Buffer Cache Sizes* on page 10
- *Optimize for Large Numbers of Users* on page 11
- *Platform-Specific Memory Options* on page 13

Optimize for Large Numbers of Users

To support the maximum number of users, you may need to increase the temporary dbspace, adjust the operating system parameters, and change the startup parameters.

**See also**

- *Paging Increases Available Memory* on page 5
- *Utilities to Monitor Swapping* on page 6
- *Server Memory* on page 6
Managing System Resources

- Manage Buffer Caches on page 7
- Determine the Sizes of the Buffer Caches on page 7
- Set the Buffer Cache Sizes on page 10
- Specify the Page Size on page 11
- Platform-Specific Memory Options on page 13

**Startup Options**
Use the following startup options for operations with large numbers of users.

- **-gm**
  Sets the default number of connections.
  
  `-gm #_connections_to_support`

  Although this represents the total number of connections the server will support, not all connections will be active at any one time.

- **-iqgovern**
  Places a ceiling on the maximum number of queries to execute at once. If more users than the `-iqgovern` limit have submitted queries, new queries will be queued until one of the active queries is finished.

  `-iqgovern #_ACTIVE_ queries_to_support`

  The optimal value for `-iqgovern` depends on the nature of your queries, number of CPUs, and size of the Sybase IQ buffer cache. The default value is $2 \times \text{numCPU} + 10$. With a large number of connected users, you may find that setting this option to $2 \times \text{numCPU} + 4$ provides better throughput.

- **-gn**
  Sets the number of execution threads for the catalog store and connectivity while running with multiple users.

  `-gn number of tasks (both user and system requests) that the database server can execute concurrently`

  The correct value for `-gn` depends on the value of `-gm`. The `start_iq` utility calculates `-gn` and sets it appropriately. Setting `-gn` too low can prevent the server from operating correctly. Setting `-gn` above 480 is not recommended.

- **-c**
  Sets the catalog store cache size.

  `-c catalog_store_cache_size`

  The catalog cache size is highly dependent on schema size and the number of objects. The catalog store buffer cache is also the general memory pool for the catalog store. To specify in MB, use the form `-c nM`, for example, `-c 64M`. Sybase recommends these values:
Table 2. Catalog Buffer Cache Settings

<table>
<thead>
<tr>
<th>For this many users</th>
<th>Set -c to this minimum value or higher</th>
</tr>
</thead>
<tbody>
<tr>
<td>up to 1000</td>
<td>64MB</td>
</tr>
<tr>
<td>up to 200</td>
<td>48MB (start_iq default for 64-bit); larger numbers of users may benefit from 64MB</td>
</tr>
</tbody>
</table>

-cl and -ch
Set upper (-ch) and lower (-cl) limits for the catalog store cache size.

-cl minimum cache size  -ch maximum cache size

If the standard catalog cache size is too small, set -cl and -ch parameters. On 32-bit platforms, try these settings:

-<code>-cl 128M</code>
-<code>-ch 256M</code>

Do not use -c in the same configuration file or command line with -ch or -cl. For related information, see the -ch cache-size option.

**Warning!** To control catalog store cache size explicitly, you must do *either* of the following, but not both, in your configuration file (.cfqg) or on the UNIX command line for server startup:

- Set the -c parameter
- Set specific upper and lower limits for the catalog store cache size using the -ch and -cl parameters

Specifying different combinations of the parameters above can produce unexpected results.

-iqmt
Sets the number of processing threads.

If -iqmt is set too low for the -gm setting, then thread starvation can occur.

**Platform-Specific Memory Options**

The total amount of usable memory is limited only by the virtual memory of the system.

**Wired Memory Pool**

On HP and Solaris platforms, you can designate a specified amount of memory as wired memory. Wired memory is shared memory that is locked into physical memory. The kernel cannot page this memory out of physical memory.

Wired memory may improve Sybase IQ performance when other applications are running on the same machine at the same time. Dedicating wired memory to Sybase IQ, however, makes it unavailable to other applications on the machine.
To create a pool of wired memory on these UNIX platforms only, specify the `-iqwmem` command-line switch, indicating the number of MB of wired memory. (You must be user root to set `-iqwmem`, except on Solaris.) On 64-bit platforms, the only upper limit on `-iqwmem` is the physical memory on the machine.

For example, on a machine with 14GB of memory, you may be able to set aside 10GB of wired memory. To do so, you specify:

```
-iqwmem 10000
```

**Note:** Use `-iqwmem` only if you have enough memory to dedicate the amount you specify for this purpose. Otherwise, you can cause serious performance degradation.

- On Solaris, `-iqwmem` always provides wired memory.
- On HP, `-iqwmem` provides wired memory if you start the server as root. It provides unwired memory if you are not root when you start the server. This behavior may change in a future version.

**Impact of Other Applications and Databases**

Server memory comes out of a pool of memory used by all applications and databases. If you try to run multiple servers or multiple databases on the same machine at the same time, or if you have other applications running, you may need to reduce the amount of memory your server requests.

You can also issue the UNIX command `ipcs -mb` to see the actual number of segments.

**Troubleshooting HP Memory Issues**

On HP-UX, check the value of the `maxdsiz_64bit` kernel parameter. This parameter restricts the amount of virtual memory available to Sybase IQ on 64-bit HP processors. See your *Installation and Configuration Guide* for the recommended value.

**See also**

- *Paging Increases Available Memory* on page 5
- *Utilities to Monitor Swapping* on page 6
- *Server Memory* on page 6
- *Manage Buffer Caches* on page 7
- *Determine the Sizes of the Buffer Caches* on page 7
- *Set the Buffer Cache Sizes* on page 10
- *Specify the Page Size* on page 11
- *Optimize for Large Numbers of Users* on page 11
Controlling File System Buffering

On some file systems, you can turn file system buffering on or off. Turning file system buffering off usually reduces paging and improves performance.

To disable file system buffering for IQ Main dbspaces of existing databases, issue the following statement:

```sql
SET OPTION "PUBLIC".OS_FILE_CACHE_BUFFERING = OFF
```

To disable file system buffering for IQ Temporary dbspaces of existing databases, issue the following statement:

```sql
SET OPTION "PUBLIC".OS_FILE_CACHE_BUFFERING_TEMPDB = OFF
```

You can only set this option for the PUBLIC group. Shut down the database and restart it for the change to take effect.

Multiplex databases do not support direct I/O file system devices. The direct I/O performance option is only available for simplex databases.

This direct I/O performance option is available on Solaris UFS, Linux, Linux IBM, AIX, and Windows file systems only. This option has no effect on HP-UX and HP-UXi and does not affect databases on raw disk. In Linux, direct I/O is supported in kernel versions 2.6.x

To enable direct I/O on Linux kernel version 2.6 and AIX, also set the environment variable `IQ_USE_DIRECTIO` to 1. Direct I/O is disabled by default in Linux kernel version 2.6 and AIX. `IQ_USE_DIRECTIO` has no effect on Solaris and Windows.

Note:

- Sybase IQ does not support direct I/O on Linux kernel version 2.4. If you set the `IQ_USE_DIRECTIO` environment variable on Linux kernel version 2.4, the Sybase IQ server does not start. The error “Error: Invalid Block I/O argument, maybe <pathname> is a directory, or it exceeds maximum file size limit for the platform, or trying to use Direct I/O on unsupported OS” is reported.
- Solaris does not have a kernel parameter to constrain the size of its file system buffer cache. Over time, the file system buffer cache grows and displaces the IQ buffer cache pages, leading to excess operating system paging activity and reduced Sybase IQ performance. Because of this, Sybase strongly recommends raw devices for databases on Solaris.
- Windows can bias the paging algorithms to favor applications at the expense of the file system. This bias is recommended for Sybase IQ performance.

See also

- Options for Java-Enabled Databases on page 16
Options for Java-Enabled Databases

Setting the `JAVA_HEAP_SIZE` option prevents run-away Java applications from using too much memory.

The `JAVA_HEAP_SIZE` option of the `SET OPTION` command sets the maximum size (in bytes) of that part of the memory that is allocated to Java applications on a per connection basis. Per connection memory allocations typically consist of the user's working set of allocated Java variables and Java application stack space. While a Java application is executing on a connection, the per connection allocations come out of the fixed cache of the database server, so it is important that a run-away Java application is prevented from using up too much memory.

See also
- *Controlling File System Buffering* on page 15

The Process Threading Model

Sybase IQ uses operating system kernel threads for best performance. By default, Sybase IQ allocates the number of threads based on the number of CPUs on the system.

Lightweight processes are underlying threads of control that are supported by the kernel. The operating system decides which lightweight processes (LWPs) should run on which processor and when. It has no knowledge about what the user threads are, but does know if they are waiting or able to run.

The operating system kernel schedules LWPs onto CPU resources. It uses their scheduling classes and priorities. Each LWP is independently dispatched by the kernel, performs independent system calls, incurs independent page faults, and runs in parallel on a multiprocessor system.

A single, highly threaded process serves all Sybase IQ users. Sybase IQ assigns varying numbers of kernel threads to each user connection, based on the type of processing being done by that connection, the total number of threads available, and the various option settings.

Insufficient Threads Error

If there are insufficient threads for a query, Sybase IQ generates this error:

```
Not enough server threads available for this query
```

This condition may well be temporary. When some other query finishes, threads are made available and the query may succeed the next time. If the condition persists, you may need to restart the server and specify more Sybase IQ threads. It is also possible that `-iqmt` is set too low for the number of connections.
Sybase IQ Options for Managing Thread Usage

- Use the server start-up option `-iqmt` to set the maximum number of threads. The default value is calculated from the number of connections and the number of CPUs and is usually adequate.
- Use the server start-up option `-iqtss` to set the stack size of the internal execution threads. The default value is generally sufficient, but may be increased if complex queries return an error indicating that the depth of the stack exceeded this limit.
- Use the `SET OPTION MAX_IQ_THREADS_PER_CONNECTION` command to set the maximum number of threads for a single user. The `SET OPTION MAX_IQ_THREADS_PER_TEAM` sets the number of threads available to a team of threads.

Use these options to control the amount of resources a particular operation consumes. For example, the DBA can set this option before issuing an `INSERT`, `LOAD`, `BACKUP`, or `RESTORE` command.

Balancing I/O

Discusses disk striping, random and sequential file disk access, and ways to control the size of the message log file.

Raw I/O (on UNIX Operating Systems)

You can create a database or dbspace on a raw device or a file system file.

Disk partitions are typically accessed in two modes: file system mode (for example through the UFS file system) or raw mode. Raw mode does unbuffered I/O, generally making a data transfer to or from the device with every read or write system call. UFS is the default UNIX file system, and is a buffered I/O system which collects data in a buffer until it can transfer an entire buffer at a time.

You create a database or dbspace on a raw device or a file system file. Sybase IQ determines automatically from the path name you specify whether it is a raw partition or a file system file. Raw partitions can be any size.

For more information, see “Working with database objects” in System Administration Guide: Volume 1 > Working with Database Objects.

See also

- *Sybase IQ and Disk Striping* on page 18
- *Internal Striping* on page 19
- *Strategic File Locations* on page 19
Sybase IQ and Disk Striping

Striping data across multiple disks is an essential technique for good performance.

Disk striping can be performed at different places in a system, often as part of RAID hardware or software, for example:

- At the device layer, such as on a disk array or controller.
- In the operating system or dedicated device management software, such as Veritas.
- In the application.

By default, IQ internally stripes pages across all files within a dbspace, so additional striping at the software or hardware level are not needed for performance. Of course, additional striping may be necessary as part of implementing storage redundancy for the database, for example if RAID-5 is used.

Best performance in Sybase IQ with storage redundancy is achieved with simple mirroring or “RAID-1”. As stated above, Sybase IQ will distribute the data across all of the 2-disk mirror sets within a dbspace.

Due to cost, most Sybase IQ databases will not use mirroring, and will be implemented with RAID-5 or a similar RAID level to achieve redundancy. With RAID-5, choosing an appropriate chunk size (how much data is written to one disk before moving on to the next disk) will have a significant performance impact on the system, since RAID-5 has a significant write overhead. If your application does frequent or time-sensitive loads, updates, or deletes, or if queries often do temp dbspace I/O, a smaller chunk size in the range of 25-50% of the size of a Sybase IQ database page will likely give best performance. If your application is mostly reads, with little write activity, a larger chunk size 75-100% of an IQ page size will likely provide best performance.

Since Sybase IQ normally attempts to prefetch multiple reads or flush multiple writes in parallel, even with only a single active query, using a very small chunk size to spread each page read or write across many disks will have little benefit, and will usually hurt performance.

When using RAID, best performance is usually achieved using hardware (such as controller or array) based RAID. Software based RAID tools will work well, but may add a modest additional performance load on the server’s CPUs.

See also

- Raw I/O (on UNIX Operating Systems) on page 17
- Internal Striping on page 19
- Strategic File Locations on page 19
Internal Striping

Disk striping takes advantage of multiple disk spindles and provides the speed of parallel disk writes.

Sybase IQ provides disk striping, options without using third-party software. If you already have a disk striping solution through third-party software and hardware, use that method instead. Disk striping can be enabled by specifying the STRIPING ON option to the CREATE DBSPACE command.

Turning Disk Striping On or Off

To change the default striping when creating a dbspace:

```sql
SET OPTION "PUBLIC".DEFAULT_DISK_STRIPING = { ON | OFF }
```

The default for the `DEFAULT_DISK_STRIPING` option is ON for all platforms. When disk striping is ON, incoming data is spread across all dbspaces with space available. When disk striping is OFF, dbspaces (disk segments) are filled up from the front on the logical file, filling one disk segment at a time.

If you change the value of DEFAULT_DISK_STRIPING, it will affect all subsequent CREATE DBSPACE operations that do not specify a striping preference.

You can remove a file from a dbspace using the ALTER DBSPACE DROP command when disk striping is on. Before dropping the dbspace, however, you must relocate all of the data in the dbspace using the `sp_iqemptyfile` stored procedure. Because disk striping spreads data across multiple files, the `sp_iqemptyfile` process may require the relocation of many tables and indexes. Use the `sp_iqdbspaceinfo` and `sp_iqdbspace` stored procedures to determine which tables and indexes reside on a dbspace.

See also

- Raw I/O (on UNIX Operating Systems) on page 17
- Sybase IQ and Disk Striping on page 18
- Strategic File Locations on page 19

Strategic File Locations

Provide additional storage resources to improve file disk I/O.

Performance related to randomly accessed files can be improved by increasing the number of disk drives devoted to those files, and therefore, the number of operations per second performed against those files. Random files include those for the IQ store, the temporary store, the catalog store, programs (including the Sybase IQ executables, user and stored procedures, and applications), and operating system files.

Conversely, performance related to sequentially accessed files can be improved by locating these files on dedicated disk drives, thereby eliminating contention from other processes. Sequential files include the transaction log and message log files.
To avoid disk bottlenecks, follow these suggestions:

- Keep random disk I/O away from sequential disk I/O. Also for best performance, use only one partition from a physical device (disk or HW RAID set) per dbspace.
- Isolate Sybase IQ database I/O from I/O in other databases, such as Adaptive Server Enterprise, or any other I/O intensive application.
- Place the database file, temporary dbspace, and transaction log file on the same physical machine as the database server.

Place the transaction log on a separate device or partition from the database to avoid database file fragmentation when using the `-m` option to truncate the transaction log.

See also

- Raw I/O (on UNIX Operating Systems) on page 17
- Sybase IQ and Disk Striping on page 18
- Internal Striping on page 19

**Transaction Log**

The transaction log file contains recovery and auditing information.

The transaction log stores all changes to the database. By default, all databases use transaction logs. Running a database with a transaction log provides greater protection against failure and better performance.

The timestamp of a transaction log file is updated only when the file grows or when it is closed. If database operations cause the transaction log file to grow without the database file growing, the timestamp of the transaction log file is more recent than the timestamp of the database file. If the database is shut down, the transaction log file and the database timestamps are updated.

To move or rename the transaction log file, use the Transaction Log utility (`dblog`). See `Transaction Log utility (dblog)` in the Utility Guide.

The size of the transaction log can also affect recovery times. To control transaction log file growth, ensure that all your tables have compact primary keys. If you perform updates or deletes on tables that do not have a primary key or a unique index not allowing NULL, the entire contents of affected rows are entered in the transaction log. If you define a primary key, the database server needs to store only the primary key column values to uniquely identify a row. If the table contains many columns or wide columns, the transaction log pages fill up much faster if no primary key is defined. Writing extra data affects performance and consumes disk space.

If a primary key does not exist, the server looks for a UNIQUE NOT NULL index on the table (or a UNIQUE constraint). A UNIQUE index that allows NULL is not enough.

To truncate the transaction log, start the server with the `-m` server startup switch. After truncation, shut down the server and restart it without the switch. See `-m iqserv15 server option` in the Utility Guide.
Warning! The Sybase IQ transaction log file is different from most relational database transaction log files. If for some reason you lose your database files, then you lose your database (unless it is the log file that is lost). However, if you have an appropriate backup, then you can reload the database.

See also
• Message Log on page 21

Message Log
Limit the size of the message log to conserve disk space.

Sybase IQ logs all messages in the message log file, including error, status, and insert notification messages. You can turn off notification messages using parameters in the LOAD and INSERT statements.

At some sites the message log file tends to grow rapidly, due to the number of insertions, LOAD option and low NOTIFY_MODULUS database option settings, or certain other conditions. Sybase IQ lets you limit the size of this file by wrapping the message log or by setting a maximum file size and archiving log files when the active Sybase IQ message log is full.

For information on setting the maximum log file size, archiving message log files, and enabling message log wrapping, see “Message logging” in System Administration Guide: Volume I > Overview of Sybase IQ System Administration.

See also
• Transaction Log on page 20

Options for Tuning Resource Use
Tune your resources for faster query execution.

Restricting Concurrent Queries
Set the -iqgovern switch to specify the number of concurrent queries on a particular server. This is not the same as the number of connections, which is controlled by your license.

There is an optimal value for -iqgovern that will provide the correct number of concurrent query access to provide optimal throughput. If -iqgovern is set over this threshold, contention or resource starvation occurs, slowing down all requests.

By specifying the -iqgovern switch, you can help Sybase IQ optimize paging of buffer data out to disk, and avoid over committing memory. The default value of -iqgovern is (2 x the number of CPUs) + 10. You may need to experiment to find an ideal value. For sites with large numbers of active connections, try setting -iqgovern slightly lower.
See also

- Setting the Number of CPUs Available on page 22
- Limiting Temporary dbspace Use By a Query on page 22
- Limiting Queries by Rows Returned on page 23
- Forcing Cursors to be Non-Scrolling on page 24
- Limiting the Number of Cursors on page 25
- Limiting the Number ofStatements on page 25
- Prefetching Cache Pages on page 26
- Optimizing for Typical Usage on page 26
- Controlling the Number of Prefetched Rows on page 27

Setting the Number of CPUs Available

Set the `iqnumbercpus` startup switch to specify the number of CPUs available. This parameter overrides the physical number of CPUs for resource planning purposes.

Using the `iqnumbercpus` switch is recommended only:

- On machines with Intel® CPUs and hyperthreading enabled, set `iqnumbercpus` to the actual number of cores
- On machines where an operating system utility has been used to restrict Sybase IQ to a subset of the CPUs within the machine

See “Setting the number of CPUs” in System Administration Guide: Volume 1 > Running Sybase IQ.

See also

- Restricting Concurrent Queries on page 21
- Limiting Temporary dbspace Use By a Query on page 22
- Limiting Queries by Rows Returned on page 23
- Forcing Cursors to be Non-Scrolling on page 24
- Limiting the Number of Cursors on page 25
- Limiting the Number of Statements on page 25
- Prefetching Cache Pages on page 26
- Optimizing for Typical Usage on page 26
- Controlling the Number of Prefetched Rows on page 27

Limiting Temporary dbspace Use By a Query

Set the `QUERY_TEMP_SPACE_LIMIT` to specify the maximum estimated amount of temp space before a query is rejected.

The `QUERY_TEMP_SPACE_LIMIT` option causes queries to be rejected if their estimated temp space usage exceeds the specified size. By default, there is no limit on temporary store usage by queries.
Sybase IQ estimates the temporary space needed to resolve the query. If the estimate exceeds the current `QUERY_TEMP_SPACE_LIMIT` setting, Sybase IQ returns an error:

```
Query rejected because it exceeds total space resource limit
```

If this option is set to 0 (the default), there is no limit, and no queries are rejected based on their temporary space requirements.

To limit the actual temporary store usage per connection, set the `MAX_TEMP_SPACE_PER_CONNECTION` option for all DML statements, including queries. `MAX_TEMP_SPACE_PER_CONNECTION` monitors and limits the actual run time temporary store usage by the statement. If the connection exceeds the quota set by the `MAX_TEMP_SPACE_PER_CONNECTION` option, an error is returned and the current statement rolls back.

**See also**

- *Restricting Concurrent Queries* on page 21
- *Setting the Number of CPUs Available* on page 22
- *Limiting Queries by Rows Returned* on page 23
- *Forcing Cursors to be Non-Scrolling* on page 24
- *Limiting the Number of Cursors* on page 25
- *Limiting the Number of Statements* on page 25
- *Prefetching Cache Pages* on page 26
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 27

**Limiting Queries by Rows Returned**

Set the value of the `QUERY_ROWS_RETURNED_LIMIT` option to prevent the optimizer from rejecting queries with large result sets.

The `QUERY_ROWS_RETURNED_LIMIT` option tells the query optimizer to reject queries that might otherwise consume too many resources. If the query optimizer estimates that the result set from a query will exceed the value of this option, it rejects the query with the message:

```
Query rejected because it exceed resource: Query_Rows_Returned_Limit
```

If you use this option, set it so that it only rejects queries that consume vast resources.

**See also**

- *Restricting Concurrent Queries* on page 21
- *Setting the Number of CPUs Available* on page 22
- *Limiting Temporary dbspace Use By a Query* on page 22
- *Forcing Cursors to be Non-Scrolling* on page 24
- *Limiting the Number of Cursors* on page 25
Forcing Cursors to be Non-Scrolling

Eliminate the temporary store node in queries that return a very large result set to improve performance.

When you use scrolling cursors with no host variable declared, Sybase IQ creates a temporary store node where query results are buffered. This storage is separate from the temporary store buffer cache. The temporary store node enables efficient forward and backward scrolling when your application searches through a result set.

However, if the query returns very large numbers (such as millions) of rows of output, and if your application performs mostly forward-scrolling operations, the memory requirements of the temporary store node may degrade query performance. To improve performance, eliminate the temporary store node by issuing the following command:

```
SET TEMPORARY OPTION FORCE_NO_SCROLL_CURSORS = 'ON'
```

**Note:** If your application performs frequent backward-scrolling, setting the `FORCE_NO_SCROLL_CURSORS` option to `ON` may actually degrade query performance, as the absence of the temporary cache forces Sybase IQ to re-execute the query for each backward scroll.

If your application rarely performs backward-scrolling, make `FORCE_NO_SCROLL_CURSORS = ‘ON’` a permanent `PUBLIC` option. It will use less memory and improve query performance.

See also

- *Restricting Concurrent Queries* on page 21
- *Setting the Number of CPUs Available* on page 22
- *Limiting Temporary dbspace Use By a Query* on page 22
- *Limiting Queries by Rows Returned* on page 23
- *Limiting the Number of Cursors* on page 25
- *Limiting the Number of Statements* on page 25
- *Prefetching Cache Pages* on page 26
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 27
Limiting the Number of Cursors

Set the `MAX_CURSOR_COUNT` option to prevent a single connection from taking too much available memory or CPU resources.

The `MAX_CURSOR_COUNT` option limits the maximum number of cursors that a connection can use at once. The default is 50. Setting this option to 0 allows an unlimited number of cursors.

See also
- *Restricting Concurrent Queries* on page 21
- *Setting the Number of CPUs Available* on page 22
- *Limiting Temporary dbspace Use By a Query* on page 22
- *Limiting Queries by Rows Returned* on page 23
- *Forcing Cursors to be Non-Scrolling* on page 24
- *Limiting the Number of Statements* on page 25
- *Prefetching Cache Pages* on page 26
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 27

Limiting the Number of Statements

Set the `MAX_STATEMENT_COUNT` option to limit the number of prepared statements for a connection can make.

The `MAX_STATEMENT_COUNT` option limits the maximum number of prepared statements that a connection can use at once. If a server needs to support more than the default number (50) of prepared statements at any one time for any one connection, then you can set the `MAX_STATEMENT_COUNT` option to a higher value.

See also
- *Restricting Concurrent Queries* on page 21
- *Setting the Number of CPUs Available* on page 22
- *Limiting Temporary dbspace Use By a Query* on page 22
- *Limiting Queries by Rows Returned* on page 23
- *Forcing Cursors to be Non-Scrolling* on page 24
- *Limiting the Number of Cursors* on page 25
- *Prefetching Cache Pages* on page 26
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 27
Prefetching Cache Pages

Set the BT_PREFETCH_MAX_MISS option to control prefetch memory behavior.

The BT_PREFETCH_MAX_MISS option determines whether to continue prefetching pages for a given query. If queries using HG indexes run more slowly than expected, try gradually increasing the value of this option.

See also

- Restricting Concurrent Queries on page 21
- Setting the Number of CPUs Available on page 22
- Limiting Temporary dbspace Use By a Query on page 22
- Limiting Queries by Rows Returned on page 23
- Forcing Cursors to be Non-Scrolling on page 24
- Limiting the Number of Cursors on page 25
- Limiting the Number of Statements on page 25
- Optimizing for Typical Usage on page 26
- Controlling the Number of Prefetched Rows on page 27

Optimizing for Typical Usage

Set the USER_RESOURCE_RESERVATION option to adjust memory use for the number of current users.

Sybase IQ tracks the number of open cursors and allocates memory accordingly. In certain circumstances, USER_RESOURCE_RESERVATION option can be set to adjust the minimum number of current cursors that Sybase IQ thinks is currently using the product and hence allocate memory from the temporary cache more sparingly.

This option should only be set after careful analysis shows it is actually required. Contact Sybase Technical Support with details if you need to set this option.

See also

- Restricting Concurrent Queries on page 21
- Setting the Number of CPUs Available on page 22
- Limiting Temporary dbspace Use By a Query on page 22
- Limiting Queries by Rows Returned on page 23
- Forcing Cursors to be Non-Scrolling on page 24
- Limiting the Number of Cursors on page 25
- Limiting the Number of Statements on page 25
- Prefetching Cache Pages on page 26
- Controlling the Number of Prefetched Rows on page 27
Controlling the Number of Prefetched Rows

Set the PrefetchRows and PrefetchBuffer parameters to improve performance on cursors under certain conditions. This is a client option that you can set on the ODBC connection dialog, or in the .odbc.ini file.

Prefetching improves performance on cursors that only fetch relative 1 or relative 0. Two connection parameters let you change cursor prefetch defaults. PrefetchRows (PROWS) sets the number of rows prefetched; PrefetchBuffer (PBUF) sets the memory available to this connection for storing prefetched rows. Increasing the number of rows you prefetch may improve performance under certain conditions:

- The application fetches many rows (several hundred or more) with very few absolute fetches.
- The application fetches rows at a high rate, and the client and server are on the same machine or connected by a fast network.
- Client/server communication is over a slow network, such as a dial-up link or wide area network.

See also
- Restricting Concurrent Queries on page 21
- Setting the Number of CPUs Available on page 22
- Limiting Temporary dbspace Use By a Query on page 22
- Limiting Queries by Rows Returned on page 23
- Forcing Cursors to be Non-Scrolling on page 24
- Limiting the Number of Cursors on page 25
- Limiting the Number of Statements on page 25
- Prefetching Cache Pages on page 26
- Optimizing for Typical Usage on page 26

Other Ways to Improve Resource Use

There are several ways to adjust your system for maximum performance or better use of disk space.

Managing Disk Space in Multiplex Databases

Get users to commit their current transactions periodically, and allow the write server to drop old table versions to free disk blocks. Specifying the auto_commit option helps minimize space due to minimize version buildup.

Sybase IQ cannot drop old versions of tables while any user on any server might be in a transaction that might need the old versions. Sybase IQ may therefore consume a very large amount of disk space when table updates and queries occur simultaneously in a multiplex
Managing System Resources

database. The amount of space consumed depends on the nature of the data and indexes and the update rate.

You can free disk blocks by allowing the write server to drop obsolete versions no longer required by queries. All users on all servers should commit their current transactions periodically to allow recovery of old table versions. The servers may stay up and are fully available. The `sp_iqversionuse` stored procedure can be used to display version usage for remote servers.

See also
- Managing Multiplex Resources Using Logical Servers on page 28
- Load Balancing Among Query Servers on page 28

Managing Multiplex Resources Using Logical Servers

Logical servers enable you to manage the use of multiplex resources most effectively. Use logical servers to assign different sets of multiplex servers to different applications to meet their individual performance requirements.

In a multiplex, each connection operates under a single logical server context. When you submit a query to a multiplex server, its execution may be distributed to one or more multiplex servers, depending upon the configuration of the connection's logical server. To dynamically adjust the resources assigned to a logical server, add or remove multiplex servers from the logical server to meet the changing needs of the applications that it serves.

See also
- Managing Disk Space in Multiplex Databases on page 27
- Load Balancing Among Query Servers on page 28

Load Balancing Among Query Servers

Using the IQ Network Client to balance the query load among multiplex query servers requires an intermediate system that is able to dispatch the client connection to a machine in a pool.

To use this method, on the client system you create a special ODBC DSN, with the IP address and port number of this intermediate load balancing system, a generic server name, and the `VerifyServerName` connection parameter set to `NO`. When a client connects using this DSN, the load balancer establishes the connection to the machine it determines is least loaded.

For details on how to define an ODBC DSN for use in query server load balancing, see “VerifyServerName parameter [Verify]” in System Administration Guide: Volume 1 > Connection and Communication Parameters.

Note: Third-party software is required. `VerifyServerName` simply allows this method to work.
Managing Database Size and Structure

Database size depends largely on indexing and data quantity. Create indexes for faster queries. Drop unnecessary objects to free disk space and shorten load times.

Index Fragmentation

- Internal index fragmentation occurs when index pages are not being used to their maximum volume.
- Row fragmentation occurs when rows are deleted. Deleting an entire page of rows frees the page, but if some rows on a page are unused, the unused space remains on the disk.
- DML operations (INSERT, UPDATE, DELETE) on tables can cause index fragmentation.

Run these stored procedures for information about fragmentation issues:

- `sp_iqrowdensity` reports row fragmentation at the FP index level. See “sp_iqrowdensity procedure,” in “System Procedures,” in Reference: Building Blocks, Tables, and Procedures.
- `sp_iqindexfragmentation` reports internal fragmentation within supplemental indexes. See “sp_iqindexfragmentation procedure,” in Reference: Building Blocks, Tables, and Procedures > System Procedures.

Review the output and decide whether you want to recreate, reorganize, or rebuild the indexes. You can create other indexes to supplement the FP index.

Minimizing Catalog File Growth

Growth of the catalog files is normal and varies depending on the application and catalog content. The size of the .db file does not affect performance, and free pages within the .db file are reused as needed.

To minimize catalog file growth:

- Avoid using IN SYSTEM on CREATE TABLE statements
- Issue COMMIT statements after running system stored procedures
- Issue COMMIT statements during long-running transactions
Network Performance

Minor changes in your environment can solve some network performance issues.

To improve network throughput, provide multiple network adaptors. Classes of users can be assigned to different networks depending on service level agreements.

In case A in Figure 12-4, clients accessing two different database servers use one network card. That means that clients accessing Servers A and B have to compete over the network and past the network card. In the case B, clients accessing Server A use a different network card than clients accessing Server B.

It would be even better to put your database servers on different machines. You may also want to put heavy users of different databases on different machines.

Figure 1: Isolating heavy network users
**Put Small Amounts of Data in Small Packets**
If you send small amounts of data over the network, keep the default network packet size small (default is 512 bytes). The `-p` server start-up option lets you specify a maximum packet size. Your client application may also let you set the packet size.

**Put Large Amounts of Data in Large Packets**
If most of your applications send and receive large amounts of data, increase default network packet size. This will result in fewer (but larger) transfers.

**Process at the Server Level**
Filter as much data as possible at the server level.
Monitoring and Tuning Performance

Describes the tools you use to determine whether your system is making optimal use of available resources.

Getting Information Using Stored Procedures

Several stored procedures display database information.

**Table 3. Name Statistics**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp_iqconnection</td>
<td>Shows information about connections and versions, including which users are using temporary dbspace, which users are keeping versions alive, what the connections are doing inside Sybase IQ, connection status, database version status, and so on. See Reference: Building Blocks, Tables, and Procedures &gt; System Procedures &gt; System stored procedures &gt; sp_iqconnection procedure</td>
</tr>
<tr>
<td>sp_iqcontext</td>
<td>Tracks and displays, by connection, information about statements that are currently executing. See Building Blocks, Tables, and Procedures &gt; System Procedures &gt; System stored procedures &gt; sp_iqcontext procedure</td>
</tr>
<tr>
<td>sp_iqcheckdb</td>
<td>Checks validity of the current database. Optionally corrects allocation problems for dbspaces or databases. See Building Blocks, Tables, and Procedures &gt; System Procedures &gt; System stored procedures &gt; sp_iqcheckdb procedure</td>
</tr>
<tr>
<td>sp_iqdbstatistics</td>
<td>Reports results of the most recent sp_iqcheckdb. See Building Blocks, Tables, and Procedures &gt; System Procedures &gt; System stored procedures &gt; sp_iqdbstatistics procedure</td>
</tr>
<tr>
<td>sp_iqdbsize</td>
<td>Displays the size of the current database. See Building Blocks, Tables, and Procedures &gt; System Procedures &gt; System stored procedures &gt; sp_iqdbsize procedure</td>
</tr>
<tr>
<td>sp_iqspaceinfo</td>
<td>Displays space usage by each object in the database. See Building Blocks, Tables, and Procedures &gt; System Procedures &gt; System stored procedures &gt; sp_iqspaceinfo procedure</td>
</tr>
</tbody>
</table>
### Profiling Database Procedures

Procedure profiling tracks the execution times of procedures, triggers, and other system events. Use profiling in Sybase Central to identify performance issues for the database or database object.

### Viewing Procedure Profiling Statistics

Set the database profiling options in Sybase Central to monitor the execution times of stored procedures, functions, events, and triggers.

#### Database Profile Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Lists the name of the object.</td>
</tr>
<tr>
<td>Owner</td>
<td>Lists the owner of the object.</td>
</tr>
<tr>
<td>Table</td>
<td>Lists which table a trigger belongs to (this column only appears on the database Profile tab).</td>
</tr>
<tr>
<td>Event</td>
<td>Shows the type of trigger for system triggers. This can be Update or Delete.</td>
</tr>
<tr>
<td>Type</td>
<td>Lists the type of object, for example, a procedure.</td>
</tr>
<tr>
<td># Exes.</td>
<td>lists the number times each object has been called.</td>
</tr>
<tr>
<td>#msecs.</td>
<td>Lists the total execution time for each object.</td>
</tr>
</tbody>
</table>
Database Object Profiles

Database objects include stored procedures, functions, events, and triggers. Database object profile properties appear line by line, and summarize execution times.

Table 5. Object Profile Properties.

<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calls</td>
<td>Lists the number of times the object has been called.</td>
</tr>
<tr>
<td>Milliseconds</td>
<td>Lists the total execution time for each object.</td>
</tr>
<tr>
<td>Line</td>
<td>Lists the line number beside each line of the procedure.</td>
</tr>
<tr>
<td>Source</td>
<td>Displays the SQL procedure, line by line.</td>
</tr>
</tbody>
</table>

Setting Database Profiling Properties in Sybase Central

Setting database profiling properties in Sybase Central requires DBA authority. Your server must be running, and you must be connected to a database.

1. In Sybase Central, right-click your database, choose Properties.
2. Click the Profiling Settings tab.
3. See online help for other profiling options.

Viewing Profiling Information For a Class of Database Objects

To display profiling information in Sybase Central about a class of database objects, click the parent folder, then review the object’s profile.

1. Open an object folder:
   - Procedures and Functions
   - Events
Monitoring and Tuning Performance

- Triggers
- System Triggers

2. Click the Profile tab in the right pane.
   Profiling information about the object appears on the Profile tab in the right pane.

**See also**

- *Setting Database Profiling Properties in Sybase Central* on page 35
- *Viewing Profiling Information For a Specific Database Object* on page 36

**Viewing Profiling Information For a Specific Database Object**

To display profiling information in Sybase Central about a specific database object, choose an object, then review the object’s profile.

1. Open an object folder:
   - Procedures and Functions
   - Events
   - Triggers
   - System Triggers
2. Click an object in the parent folder.
3. Click the Profile tab in the right pane.
   Profiling information about the object appears on the Profile tab in the right pane.

**See also**

- *Setting Database Profiling Properties in Sybase Central* on page 35
- *Viewing Profiling Information For a Class of Database Objects* on page 35

**Procedure Profiling Statistics**

Set the database profiling options, then use the profiling options to return performance statistics for stored procedures, functions, events, and triggers.

*sa_procedure_profile_summary*

*sa_procedure_profile_summary* is a system procedure that reports summary information about the execution times for all procedures, functions, events, or triggers that have been executed in a database. This procedure provides the same information for these objects as the Profile tab in Sybase Central.
Table 6. sa_procedure_profile_summary Statistics

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object_type</td>
<td>Identifies the object type:</td>
</tr>
<tr>
<td></td>
<td>• P (Stored procedure)</td>
</tr>
<tr>
<td></td>
<td>• F (Function)</td>
</tr>
<tr>
<td></td>
<td>• T (Trigger)</td>
</tr>
<tr>
<td></td>
<td>• E (Event)</td>
</tr>
<tr>
<td></td>
<td>• S (System trigger)</td>
</tr>
<tr>
<td>object_name</td>
<td>Lists the name of the object.</td>
</tr>
<tr>
<td>executions</td>
<td>Lists the number times each object has been called.</td>
</tr>
<tr>
<td>owner_name</td>
<td>Lists the owner of the object.</td>
</tr>
<tr>
<td>table_name</td>
<td>Specifies which table to profile triggers.</td>
</tr>
<tr>
<td>executions</td>
<td>Lists the number of times the object has been called.</td>
</tr>
<tr>
<td>Milliseconds</td>
<td>Identifies the time to execute the line, in milliseconds.</td>
</tr>
<tr>
<td>foreign_owner</td>
<td>Identifies the database user who owns the foreign table for a system trigger.</td>
</tr>
<tr>
<td>foreign_table</td>
<td>Identifies The name of the foreign table for a system trigger.</td>
</tr>
</tbody>
</table>

See also
- Viewing Procedure Profiling Statistics on page 34
- Database Object Profiles on page 35

Procedure Profile
sa_procedure_profile reports information about the execution time for each line within procedures, functions, events, or triggers executed in a database.

Table 7. sa_procedure_profile Statistics

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object_type</td>
<td>Identifies the object type:</td>
</tr>
<tr>
<td></td>
<td>• P (Stored procedure)</td>
</tr>
<tr>
<td></td>
<td>• F (Function)</td>
</tr>
<tr>
<td></td>
<td>• T (Trigger)</td>
</tr>
<tr>
<td></td>
<td>• E (Event)</td>
</tr>
<tr>
<td></td>
<td>• S (System trigger)</td>
</tr>
<tr>
<td>Column name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>object_name</td>
<td>Lists the name of the object.</td>
</tr>
<tr>
<td>owner_name</td>
<td>Lists the owner of the object.</td>
</tr>
<tr>
<td>table_name</td>
<td>Identifies the table associated with a trigger (the value is NULL for other</td>
</tr>
<tr>
<td></td>
<td>object types).</td>
</tr>
<tr>
<td>Line_number</td>
<td>Identifies the line number within the procedure.</td>
</tr>
<tr>
<td>executions</td>
<td>Lists the number of times the object has been called</td>
</tr>
<tr>
<td>Milliseconds</td>
<td>Lists the objects execution time</td>
</tr>
<tr>
<td>percentage</td>
<td>Identified the percentage of the total execution time required for the spe-</td>
</tr>
<tr>
<td></td>
<td>cific line.</td>
</tr>
<tr>
<td>foreign_owner</td>
<td>Identifies the database user who owns the foreign table for a system trigger.</td>
</tr>
<tr>
<td>foreign_table</td>
<td>Identifies the name of the foreign table for a system trigger.</td>
</tr>
</tbody>
</table>

**Setting Database Profiling Options with Interactive SQL**

Use **sa_server_option** to set database profiling options in Interactive SQL. Your server must be running, and you must have DBA authority, and be connected to a database.

In Interactive SQL, run **sa_server_option**, and set the procedure_profiling options.

For example:

```sql
CALL sa_server_option ('procedure_profiling', 'ON')
```

For other options, see *SQL Anywhere Server - SQL Reference > System procedures > Alphabetical list of system procedures > sa_server_option system procedure.*

**Note:** This reference points to SQL Anywhere documentation.

**See also**

- *Generating Profiling Information with Interactive SQL* on page 38

**Generating Profiling Information with Interactive SQL**

**sa_procedure_profile** and **sa_procedure_profile_summary** generate execution statistics for procedures, functions, events, and triggers.

In Interactive SQL, run **sa_procedure_profile** or **sa_procedure_profile_summary**. For example:

```sql
CALL sa_server_option ('procedure_profiling', 'ON')
```

For other options, see *SQL Anywhere Server – SQL Reference.*

**Note:** This reference points to SQL Anywhere documentation.
Data Model Recommendations

Good database performance begins with good database design. Take the time to incorporate Sybase IQ's unique design features into your schema during development for better response time and faster query results.

Indexing Tips

Choose the correct column index type to make your queries run faster.

Sybase IQ provides some indexes automatically—an index on all columns that optimizes projections, and an HG index for UNIQUE and PRIMARY KEYS and FOREIGN KEYS. While these indexes are useful for some purposes, you may need other indexes to process certain queries as quickly as possible.

Index Advisor

The index advisor generates messages when the optimizer would benefit from an additional index on one or more columns in your query.

To activate the index advisor, set the INDEX_ADVISOR option ON. Messages print as part of a query plan or as a separate message in the message log (.iqmsg) if query plans are not enabled, and output is in OWNER.TABLE.COLUMN format. For details, see “INDEX_ADVISOR option,” in “Database Options,” in Reference: Statements and Options.

LF or HG Indexes

Consider creating either an LF or HG index on grouping columns referenced by the WHERE clause in a join query if the columns are not using enumerated FP storage. The Sybase IQ optimizer may need metadata from the enumerated FP or HG/LF index to produce an optimal query plan. Non-aggregated columns referenced in the HAVING clause may also benefit from a LF or HG index to help with query optimization. For example:

```
SELECT c.name, SUM(l.price * (1 - l.discount))
FROM customer c, orders o, lineitem l
WHERE c.custkey = o.custkey
    AND o.orderkey = l.orderkey
    AND o.orderdate >= "1994-01-01"
    AND o.orderdate < "1995-01-01"
GROUP by c.name
HAVING c.name NOT LIKE "I%"
    AND SUM(l.price * (1 - l.discount)) > 0.50
ORDER BY 2 desc
```

Adding indexes increases storage requirements and load time. Add indexes only if there is a net benefit to query performance.
When and Where to use Indexes

Indexes are the primary tuning mechanisms inside Sybase IQ. Knowing when and where to use indexes can make your queries run faster.

Always use indexes on:

• Join columns (HG index regardless of cardinality)
• Searchable columns (HG or LF index based on cardinality)
• DATE, TIME, and DATETIME/TIMESTAMP columns (DATE, TIME, DTTM)
  The DATE, TIME, or DATETIME/TIMESTAMP column should also have an LF or HG index depending on data cardinality.
• If you are uncertain whether the column will be used heavily, place an LF or HG index on the column. Workload Management can subsequently be enabled to monitor the use of indexes.
• Use PRIMARY KEY, UNIQUE CONSTRAINT, or UNIQUE HG indexes where appropriate, as they provide IQ with additional information about the unique data in the indexed column(s).
• A column with an HNG or CMP index should have a corresponding LF or HG index
• Indexes are not needed on columns whose data is ONLY returned to the client (projected)

See also

• Indexing Tips on page 39
• Simple Index Selection Criteria on page 41
Simple Index Selection Criteria

Answers to some simple questions can help you choose the right index for a column.

To determine the best indexes for your datamodel without regard for queries, ask yourself these simple questions about each column:

- Is the cardinality greater than 1500-2000?
  If the answer is yes, place an HG index on this column. If not, place an LF index on the column.

- Does the column contain DATE, TIME, DATETIME, or TIMESTAMP data?
  If the answer is yes, place a DATE, TIME, or DTM index on this column. You should also place an LF or HG on the column.

- Will the column be used in range searches or aggregations?
  If the answer is yes, place an HNG index on the column. You should also place an LF or HG on the column. If the aggregation contains more than just the column, an HNG may not be appropriate. In most cases an HNG index is not needed as the LF or HG indexes have more than enough capability to perform the aggregations. This does not apply to DATE, TIME, or DATETIME types.

- Will this column be used for word searching?
  If the answer is yes, place a WD index on the column. An LF or HG index is not necessary and would consume significant space.

- Will this column be used for full text searching?
  If the answer is yes, place a TEXT index on the column. An LF or HG is not necessary and would consume significant space.

- Will two columns in the same table be compared to each other (A = B, A < B, A > B, A <= B, A > + B)?
If the answer is yes, place a CMP index on the two columns.

- Will this column, or set of columns, be used in `GROUP BY` or `ORDER BY` statements?
  - If the answer is yes, place an HG index on the column, or columns in the `GROUP BY` or `ORDER BY` statement. Each column should also have a corresponding HG or LF index.

- Is this column part of a multicolumn primary key, constraint, or index?
  - If the answer is yes, place an HG or LF index on each column in the multicolumn index.

See also

- *Indexing Tips* on page 39
- *When and Where to use Indexes* on page 40
- *HG Index Loads* on page 42
- *Multi-Column Indexes* on page 44
- *Join Column* on page 45
- *Primary Keys* on page 46
- *Foreign Keys* on page 46
- *Proper Data Type Sizing* on page 47
- *IQ UNIQUE and MINIMIZE_STORAGE* on page 48
- *Null Values* on page 49
- *Unsigned Data Types* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52
- *Denormalizing for Performance* on page 53
- *UNION ALL Views for Faster Loads* on page 54

**HG Index Loads**

Relative to other IQ indexes, the HG indexes are more expense to maintain during data loads and deletions. A main contributor to the performance of the HG index is the location of the data within the HG index structure: the sparsity or density of the operation.

Dense HG operations are those in which the affected rows are tightly grouped around certain keys. Sparse operations are those where there may be just a few rows per key that must be affected. For instance, dates on data are typically grouped around the time the operation was logged, data modified, etc. This means that new data will be placed at the end of the HG index structure. When deleting data in the date HG index, said data would typically come off in chunks of days, weeks, months, etc and thus be removed from the beginning of the HG btree or be tightly grouped around a few keys for deletion. These operations are very fast, relatively speaking, as IQ will operate on a few pages and affect a tremendous number of rows.

Data that is rather sparse, like Prices, Customer IDs, City, Country, etc., are very different. As “pricing” data, for instance, is loaded each value will vary widely across all data already in the index. If the column is tracking stock prices the numeric field to store that data will be densely...
updated because the data being changed will be across the nearly the entire range of values already loaded. These operations are slower due to the amount of index pages that must be maintained for each row being affected. A worst case scenario is that IQ is forced to read and write 1 page for each row being loaded or deleted. While this can be less than optimal, Sybase IQ has been design to parallel process phase 2 of the HG index loads and the deletes so that the impact is greatly reduced.

All of this is well and good, but how does it affect the data model design and indexing? Typical tuning and optimization within Sybase IQ generally boils down to indexes or the lack thereof. Knowing how the indexes can be affected by the data and loading is an important aspect when deciding which indexes to put in place and which to leave off. Because HG indexes take, relatively, more time to load than other indexes they are often the subject of focus when it comes to use and design. Certainly, HG indexes can help with query performance. There are times, though, where adding an index may have a slight positive impact on queries but have more of an impact to data loads. In these situations, it is important to understand why the load or delete took longer and what can be done about it.

The sparsity or density of new data with respect to currently loaded data plays a critical role in this. If a relatively random column of a Customer ID must be indexed for fast query performance and an index must be on that column. Suppose, though, that a primary key exists on the table and it is the Customer ID and a Date field storing a transaction datetime. If the ordering were left as (customer_id, transaction_date) the data would be sparsely loaded or deleted from the table in most case. Data being loaded will be done so by transaction date. Since the Customer ID column is first in the multicolumn index, though, it will force IQ to touch data throughout the entire HG index structure.

A simple change in order to (transaction_date, customer_id) changes this behavior. The index is still in place to control referential integrity for the primary key. The ordering of the columns is immaterial for primary key enforcement. As such, we can change the column order without causing any downstream ill effects. This simple change will now force all new data being loaded by transaction date to be inserted at the end of the HG index structure in a very dense manner. Over time the loads will perform consistently as the data is, generally, always going to the end of the HG structure.

Simply changing the column ordering in a multicolumn index can have drastic impacts on performance. The size of the HG index shouldn't change much as the data is still the same width regardless of order. What will change is how fast the data is loaded or deleted from the table.

See also
• Indexing Tips on page 39
• When and Where to use Indexes on page 40
• Simple Index Selection Criteria on page 41
• Multi-Column Indexes on page 44
• Join Column on page 45
• Primary Keys on page 46
Multi-Column Indexes

Currently, only HG, UNIQUE HG, UNIQUE CONSTRAINT, and PRIMARY KEY indexes support multiple columns in index creation, but multi-column indexes are also useful for GROUP BY and ORDER BY statements.

From a statistics point of view, multi-column indexes provide enough information in multi-column table joins to let the optimizer know the exact statistics of the join and whether or not it is a many-to-many or one-to-many join. The optimizer is also smart enough to use the statistics for optimization, but use individual HG/LF indexes for the actual work. The optimizer costs out all join and sort scenarios and decides which index(es) is best for that operation. The statistics help it get to that point.

Some items to keep in mind about the HG indexes:

- HG inserts are the most expensive in Sybase IQ
- Try to guarantee that inserts will happen at the end of the index
  Place generally incrementing data, like a transaction date or batch number (sequential data), at the beginning of the index list. Something that will try to guarantee a sequential key

See the previous section the HG index loads.

See also
- Indexing Tips on page 39
- When and Where to use Indexes on page 40
- Simple Index Selection Criteria on page 41
- HG Index Loads on page 42
- Join Column on page 45
- Primary Keys on page 46
- Foreign Keys on page 46
- Proper Data Type Sizing on page 47
Join Column

For joins, keep the data types as narrow as possible to reduce disk I/O and memory requirements.

Because integer comparisons are quicker than character comparisons, use integer data types (unsigned if possible) in joins. Keeping the data types as narrow as possible improves join performance by reducing disk I/O and memory requirements. Because the $HG$ index has slightly more capability from a join perspective, use an $HG$ index on join columns rather than a cardinality appropriate index ($LF$ or $HG$). This should be weighed against the potential increase in time to load the $HG$ index as compared to the $LF$ index.

See also

- Indexing Tips on page 39
- When and Where to use Indexes on page 40
- Simple Index Selection Criteria on page 41
- HG Index Loads on page 42
- Multi-Column Indexes on page 44
- Primary Keys on page 46
- Foreign Keys on page 46
- Proper Data Type Sizing on page 47
- IQ UNIQUE and MINIMIZE_STORAGE on page 48
- Null Values on page 49
- Unsigned Data Types on page 49
- LONG VARCHAR and LONG VARBINARY on page 50
- Large Object Storage on page 51
- Temporary Tables on page 52
- Denormalizing for Performance on page 53
- UNION ALL Views for Faster Loads on page 54
Primary Keys

Multi-column primary keys should have an additional LF or HG index placed on each column specified in the primary key. This must be done manually as IQ only creates an HG index on the composite columns.

**UNIQUE constraint**, **UNIQUE HG**, and primary key share an identical structure. That structure uses an HG index with no G-Array to store the row ids. When possible, use primary keys on tables. This helps the optimizer make more informed query path decisions even if the index is not used. The index structure provides detailed statistics to help the optimizer make better choices as well as providing a structure to traverse the data.

See also
- **Indexing Tips** on page 39
- **When and Where to use Indexes** on page 40
- **Simple Index Selection Criteria** on page 41
- **HG Index Loads** on page 42
- **Multi-Column Indexes** on page 44
- **Join Column** on page 45
- **Foreign Keys** on page 46
- **Proper Data Type Sizing** on page 47
- **IQ UNIQUE and MINIMIZE_STORAGE** on page 48
- **Null Values** on page 49
- **Unsigned Data Types** on page 49
- **LONG VARCHAR and LONG VARBINARY** on page 50
- **Large Object Storage** on page 51
- **Temporary Tables** on page 52
- **Denormalizing for Performance** on page 53
- **UNION ALL Views for Faster Loads** on page 54

Foreign Keys

As with primary keys, use foreign keys to improve query join performance. This gives IQ one more piece of information on how tables are joined and the statistics behind those joins. IQ automatically creates an HG Index on the foreign key column, so no additional HG or LF index is necessary. A foreign key requires that a primary key exists on referenced table.

See also
- **Indexing Tips** on page 39
- **When and Where to use Indexes** on page 40
Proper Data Type Sizing

Size all data types as accurately as possible, especially character-based data types.

To decide which data type to use for a column, consider these factors:

- Sybase IQ includes a large number of data types. Using the correct data types for your application leads to optimal performance gains.
- If HOUR, MINUTE and SECOND information is not necessary, use DATE instead of DATETIME.
- If the data will fit within a TINYINT or SMALLINT datatype use that rather than INTEGER or BIGINT.
- Do not over allocate storage when defining NUMERIC() or DECIMAL() as it can be costly for data that does not need all that level of precision.
- CHAR() and VARCHAR() types are fixed width in the default Flat FP index. The only difference is the addition of 1 byte to each VARCHAR() row that represents the number of bytes in use.

Sybase IQ includes new compression algorithms that compress large repeating patterns often seen in BINARY(), CHAR(), VARCHAR(), and VARBINARY() data types.

See also

- Indexing Tips on page 39
- When and Where to use Indexes on page 40
- Simple Index Selection Criteria on page 41
- HG Index Loads on page 42
- Multi-Column Indexes on page 44
- Join Column on page 45
IQ UNIQUE and MINIMIZE_STORAGE

Using IQ UNIQUE and MINIMIZE_STORAGE can save disk space and improve performance.

Use IQ UNIQUE and MINIMIZE_STORAGE whenever possible to help minimize storage usage for the default FP index types. By default, optimal data compression is not enabled via these options which can lead to some additional space consumption. Employing either of these options in the data model helps compress the data as much as possible to achieve maximum storage and performance.
Null Values

Defining columns as NULL or NOT NULL helps the optimizer work more efficiently.

Specifying NULL or NOT NULL allows the optimizer a more educated guess at joins and search criteria by having one more piece of information about the characteristics of the data. NULL data does not save space on the database page, as it would in other databases. NULL data will, however, be compressed out when stored on disk due to the IQ compression algorithms and optimized indexes.

- Always specify NULL or NOT NULL
- Open Client and ODBC connections have different default behavior when table is created
- Give the optimizer an additional piece of information about the characteristics of the data for joins and search arguments

See also

- Indexing Tips on page 39
- When and Where to use Indexes on page 40
- Simple Index Selection Criteria on page 41
- HG Index Loads on page 42
- Multi-Column Indexes on page 44
- Join Column on page 45
- Primary Keys on page 46
- Foreign Keys on page 46
- Proper Data Type Sizing on page 47
- IQ UNIQUE and MINIMIZE_STORAGE on page 48
- Unsigned Data Types on page 49
- LONG VARCHAR and LONG VARBINARY on page 50
- Large Object Storage on page 51
- Temporary Tables on page 52
- Denormalizing for Performance on page 53
- UNION ALL Views for Faster Loads on page 54

Unsigned Data Types

In some cases, using unsigned data types can eliminate sign comparisons and create faster queries.

Use unsigned data types when the sign of the data does not matter as all data will always be greater than or equal to zero. The lack of sign storage results in column comparisons that no longer have to perform sign comparison. This increases performance and eliminates a step in the joining and searching of data, particularly for key columns.
See also

- Indexing Tips on page 39
- When and Where to use Indexes on page 40
- Simple Index Selection Criteria on page 41
- HG Index Loads on page 42
- Multi-Column Indexes on page 44
- Join Column on page 45
- Primary Keys on page 46
- Foreign Keys on page 46
- Proper Data Type Sizing on page 47
- IQ UNIQUE and MINIMIZE_STORAGE on page 48
- Null Values on page 49
- LONG VARCHAR and LONG VARBINARY on page 50
- Large Object Storage on page 51
- Temporary Tables on page 52
- Denormalizing for Performance on page 53
- UNION ALL Views for Faster Loads on page 54

**LONG VARCHAR and LONG VARBINARY**

Use VARCHAR() and VARBINARY() to increase column storage without using large object storage mechanisms.

Typically, developers and DBAs think of VARBINARY() and VARCHAR() data as being limited to 255 bytes. IQ supports VARCHAR() and VARBINARY() widths of up to 32K (also known as LONG VARCHAR or LONG VARBINARY). This allows for much larger storage of text or binary data without needing to move into the highly specialized large objects storage mechanism of BLOB/CLOB or IMAGE/TEXT data types.

- Can be used to store moderate amounts of text or binary data
- Maximum width is 32K (64K ASCII hex for VARBINARY())
- The WORD and TEXT index is the only index allowed on VARCHAR() data wider than 255 bytes
- Storage will be allocated in 256 byte chunks
- A 257 byte string will require 512 bytes of storage
- A 511 byte string will also require 512 bytes of storage

See also

- Indexing Tips on page 39
- When and Where to use Indexes on page 40
- Simple Index Selection Criteria on page 41
- HG Index Loads on page 42
Large Object Storage

Use Large Object data types for data that requires more than 32K in storage.

- Large object data types store ASCII (TEXT/CLOB) and binary (IMAGE/BLOB) data. Each BLOB/CLOB cell of data is stored on one or more pages.
  - Assuming the page size is 128K
  - If the data is 129K, it will require 2 pages to store the information
  - If the data is 1K, it will require 1 page to store the data
  - In either case, the page(s) are compressed on disk into multiples of the block size
- Can be used to store binary or text based objects
- Extends the long binary data type from a maximum size of 6K to an unlimited size
- The TEXT index is the only viable index
- Can be fully searched with the TEXT index and its search capabilities
- Special function to return the size of an object (byte_length64)
- Special function to return portions of the object, not the entire contents (byte_substr64)
- Can extract contents of a binary object cell to an individual file with the BFILE() function

See also
- Indexing Tips on page 39
- When and Where to use Indexes on page 40
- Simple Index Selection Criteria on page 41
- HG Index Loads on page 42
- Multi-Column Indexes on page 44
- Join Column on page 45
- Primary Keys on page 46
- Foreign Keys on page 46
Temporary Tables

If you want the data to persist through transaction commits, use the `ON COMMIT PRESERVE ROWS` option when you create global temporary tables or declare local temporary tables.

There are three types of Temporary Tables:

- **# tables**
  ```sql
  CREATE TABLE temp_table( col1 int )
  ```

- **Local Temporary Tables**
  ```sql
  DECLARE LOCAL TEMPORARY TABLE temp_table ( col1 int )
  ```
  Local Temporary Tables behave just like # tables

- **Global Temporary Tables**
  ```sql
  CREATE GLOBAL TEMPORARY TABLE table temp_table ( col1 int )
  ```
  Global Temporary Table structure is static across connections and reboots

Normal hash (#) tables do not need the `ON COMMIT PRESERVE ROWS` option because the data in a hash table will always persist through transaction commits.

See also

- **Indexing Tips** on page 39
- **When and Where to use Indexes** on page 40
- **Simple Index Selection Criteria** on page 41
- **HG Index Loads** on page 42
- **Multi-Column Indexes** on page 44
- **Join Column** on page 45
- **Primary Keys** on page 46
- **Foreign Keys** on page 46
- **Proper Data Type Sizing** on page 47
- **IQ UNIQUE and MINIMIZE_STORAGE** on page 48
- **Null Values** on page 49
- **Unsigned Data Types** on page 49
Denormalizing for Performance

Although denormalizing your database can improve performance, there are risks and disadvantages.

**Risks**

Denormalization can be successfully performed only with thorough knowledge of the application and should be performed only if performance issues indicate that it is needed. Consider the effort required to keep your data up-to-date.

This is a good example of the differences between decision support applications, which frequently need summaries of large amounts of data, and transaction processing needs, which perform discrete data modifications. Denormalization usually favors some processing, at a cost to others.

Denormalization has the potential for data integrity problems, which must be carefully documented and addressed in application design.

**Deciding to Denormalize**

Analyze the data access requirements of the applications in your environment and their actual performance characteristics, including:

- What are the critical queries, and what is the expected response time?
- What tables or columns do they use? How many rows per access?
- What is the usual sort order?
- What are concurrency expectations?
- How big are the most frequently accessed tables?
- Do any processes compute summaries?

**See also**

- Indexing Tips on page 39
- When and Where to use Indexes on page 40
- Simple Index Selection Criteria on page 41
- HG Index Loads on page 42
- Multi-Column Indexes on page 44
- Join Column on page 45
- Primary Keys on page 46
- Foreign Keys on page 46
- Proper Data Type Sizing on page 47
UNION ALL Views for Faster Loads

UNION ALL views can improve load performance when it is too expensive to maintain secondary indexes for all rows in a table.

Sybase IQ lets you split the data into several separate base tables (for example, by date). You load data into these smaller tables. You then join the tables back together into a logical whole by means of a UNION ALL view, which you can then query.

This strategy can improve load performance, but may negatively impact the performance of some types of queries. Most types of queries have roughly similar performance against a single base table or against a UNION ALL view over smaller base tables, as long as the view definition satisfies all constraints. However, some types of queries, especially those involving DISTINCT or involving joins with multiple join columns, may perform significantly slower against a UNION ALL view than against a single large base table. Before choosing to use this strategy, determine whether the improvements in load performance are worth the degradation in query performance for your application.

UNION ALL views can be efficient to administer. If the data is partitioned by month, for example, you can drop an entire month’s worth of data by deleting a table and updating the UNION ALL view definition appropriately. You can have many view definitions for a year, a quarter, and so on, without adding extra date range predicates.

To create a UNION ALL view, choose a logical means of dividing a base table into separate physical tables. The most common division is by month. For example, to create a view including all months for the first quarter, enter:

```
CREATE VIEW
SELECT * JANUARY
UNION ALL
SELECT * FEBRUARY
UNION ALL
SELECT * MARCH
UNION ALL
```

Each month, you can load data into a single base table—JANUARY, FEBRUARY, or MARCH in this example. Next month, load data into a new table with the same columns, and the same index types.

For syntax details, see UNION operation in the Reference: Statements and Options.
Note: You cannot perform an \texttt{INSERT \ldots SELECT} into a \texttt{UNION ALL} view. \texttt{UNION ALL} operators are not fully parallel in this release. Their use may limit query parallelism.

See also

- \textit{Indexing Tips} on page 39
- \textit{When and Where to use Indexes} on page 40
- \textit{Simple Index Selection Criteria} on page 41
- \textit{HG Index Loads} on page 42
- \textit{Multi-Column Indexes} on page 44
- \textit{Join Column} on page 45
- \textit{Primary Keys} on page 46
- \textit{Foreign Keys} on page 46
- \textit{Proper Data Type Sizing} on page 47
- \textit{IQ UNIQUE and MINIMIZE\_STORAGE} on page 48
- \textit{Null Values} on page 49
- \textit{Unsigned Data Types} on page 49
- \textit{LONG VARCHAR and LONG VARBINARY} on page 50
- \textit{Large Object Storage} on page 51
- \textit{Temporary Tables} on page 52
- \textit{Denormalizing for Performance} on page 53

\textbf{Optimizing Queries That Reference \texttt{UNION ALL} Views}

To adjust performance for queries that reference \texttt{UNION ALL} views, set the \texttt{JOIN\_PREFERENCE} option, which affects joins between \texttt{UNION ALL} views.

All partitions in a \texttt{UNION ALL} view must have a complete set of indexes defined for optimization to work. Queries with \texttt{DISTINCT} will tend to run more slowly using a \texttt{UNION ALL} view than a base table.

Sybase IQ includes optimizations for \texttt{UNION ALL} views, including:

- Split \texttt{GROUP BY} over \texttt{UNION ALL} view
- Push-down join into \texttt{UNION ALL} view

A \texttt{UNION} can be treated as a partitioned table only if it satisfies all of the following constraints:

- It contains only one or more \texttt{UNION ALL}.
- Each arm of the \texttt{UNION} has only one table in its \texttt{FROM} clause, and that table is a physical base table.
- No arm of the \texttt{UNION} has a \texttt{DISTINCT}, a \texttt{RANK}, an aggregate function, or a \texttt{GROUP BY} clause.
- Each item in the \texttt{SELECT} clause within each arm of the \texttt{UNION} is a column.
The sequence of data types for the columns in the SELECT list of the first UNION arm is identical to the sequence in each subsequent arm of the UNION.

See also
- Managing UNION ALL View Performance on page 56

**Managing UNION ALL View Performance**

Structure queries to evaluate the DISTINCT operator before the ORDER BY, where the sort order is ASC.

Certain optimizations, such as pushing a DISTINCT operator into a UNION ALL view, are not applied when the ORDER BY is DESC because the optimization that evaluates DISTINCT below a UNION does not apply to DESC order. For example, the following query would impact performance:

```
SELECT DISTINCT state FROM testVU ORDER BY state DESC;
```

To work around this performance issue, queries should have the DISTINCT operator evaluated before the ORDER BY, where the sort order is ASC and the optimization can be applied:

```
SELECT c.state FROM (SELECT DISTINCT state
    FROM testVUA) c
ORDER BY c.state DESC;
```

See also
- Optimizing Queries That Reference UNION ALL Views on page 55

**Monitoring Performance Statistics**

Use Performance Monitor in Sybase Central to display statistics for simplex and multiplex servers. Statistics display in a dynamic chart in real time.

*Note:* Topics in this section describes cover simplex servers only. See *Using Sybase IQ* Multiplex level for multiplex servers.

**Monitoring Performance at the Server Level**

Use Performance monitor in Sybase Central to monitor statistics on a simplex or multiplex server.

1. To start Performance monitor, click the server name in the Sybase Central folders view.
2. In the right pane, click the Performance monitor tab.

See Servers > Monitoring performance in Sybase IQ help for more information and options.
See also
- Memory Usage Statistics on page 57
- Cache Statistics on page 58
- CPU Usage Statistics on page 59
- Thread Statistics on page 60
- Connection Statistics on page 61
- Request Statistics on page 62
- Transaction Statistics on page 63
- Store I/O Statistics on page 63
- DBspace Usage Statistics on page 64
- Network Statistics on page 65

Memory Usage Statistics
Memory usage statistics show server memory statistics.

Table 8. Memory Usage

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Allocated</td>
<td>Memory allocated by the IQ server in megabytes</td>
<td>Yes</td>
</tr>
<tr>
<td>Maximum Memory Allocated</td>
<td>Maximum memory allocated by the IQ server in megabytes</td>
<td>No</td>
</tr>
</tbody>
</table>

See also
- Monitoring Performance at the Server Level on page 56
- Cache Statistics on page 58
- CPU Usage Statistics on page 59
- Thread Statistics on page 60
- Connection Statistics on page 61
- Request Statistics on page 62
- Transaction Statistics on page 63
- Store I/O Statistics on page 63
- DBspace Usage Statistics on page 64
- Network Statistics on page 65
Cache Statistics

Cache statistics describe cache use.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog Cache Hits</td>
<td>Number of catalog cache hits per second.</td>
<td>No</td>
</tr>
<tr>
<td>Temporary Cache Hits</td>
<td>Number of temporary cache hits per second.</td>
<td>No</td>
</tr>
<tr>
<td>Main Cache Hits</td>
<td>Number of main cache hits per second.</td>
<td>No</td>
</tr>
<tr>
<td>Catalog Cache Reads</td>
<td>Number of catalog cache page lookups per second.</td>
<td>Yes</td>
</tr>
<tr>
<td>Temporary Cache Reads</td>
<td>Number of temporary cache page lookups per second.</td>
<td>No</td>
</tr>
<tr>
<td>Main Cache Reads</td>
<td>Number of main cache page lookups per second.</td>
<td>No</td>
</tr>
<tr>
<td>Catalog Cache Current Size</td>
<td>Current catalog cache size in megabytes.</td>
<td>No</td>
</tr>
<tr>
<td>Temporary Cache Current Size</td>
<td>Current temporary cache size in megabytes.</td>
<td>No</td>
</tr>
<tr>
<td>Main Cache Current Size</td>
<td>Current main cache size in megabytes.</td>
<td>No</td>
</tr>
<tr>
<td>Catalog Cache in Use Percent</td>
<td>Percentage of catalog cache in use.</td>
<td>No</td>
</tr>
<tr>
<td>Temporary Cache in Use Percent</td>
<td>Percentage of Temporary cache in use.</td>
<td>No</td>
</tr>
<tr>
<td>Main Cache in Use Percent</td>
<td>Percentage of Main cache size in use.</td>
<td>No</td>
</tr>
<tr>
<td>Catalog Cache Pinned</td>
<td>Number of pinned catalog cache pages.</td>
<td>No</td>
</tr>
<tr>
<td>Temporary Cache Pinned</td>
<td>Number of pinned temporary cache pages.</td>
<td>No</td>
</tr>
<tr>
<td>Main Cache Pinned</td>
<td>Number of pinned main cache pages.</td>
<td>No</td>
</tr>
</tbody>
</table>
### Name | Description | Monitored By Default?
---|---|---
Catalog Cache Pinned Percentage | Percentage of catalog cache pinned. | No
Temporary Cache Pinned Percentage | Percentage of temporary cache pinned. | No
Main Cache Pinned Percentage | Percentage of main cache pinned. | No
Catalog Cache Dirty Pages Percentage | Percentage of catalog cache dirty pages. | No
Temporary Cache Dirty Pages Percentage | Percentage of temporary cache dirty pages. | No
Main Cache Dirty Pages Percentage | Percentage of main cache dirty pages. | No

### See also
- Monitoring Performance at the Server Level on page 56
- Memory Usage Statistics on page 57
- CPU Usage Statistics on page 59
- Thread Statistics on page 60
- Connection Statistics on page 61
- Request Statistics on page 62
- Transaction Statistics on page 63
- Store I/O Statistics on page 63
- DBspace Usage Statistics on page 64
- Network Statistics on page 65

### CPU Usage Statistics
CPU usage statistics show the percentage of CPU resources in use.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage</td>
<td>IQ process CPU usage percentage, including both system and user usage.</td>
<td>Yes</td>
</tr>
<tr>
<td>CPU System Usage</td>
<td>IQ process CPU system usage percentage.</td>
<td>No</td>
</tr>
</tbody>
</table>
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU User Usage</td>
<td>IQ process CPU user usage percentage.</td>
<td>No</td>
</tr>
</tbody>
</table>

#### See also
- *Monitoring Performance at the Server Level* on page 56
- *Memory Usage Statistics* on page 57
- *Cache Statistics* on page 58
- *Thread Statistics* on page 60
- *Connection Statistics* on page 61
- *Request Statistics* on page 62
- *Transaction Statistics* on page 63
- *Store I/O Statistics* on page 63
- *DBspace Usage Statistics* on page 64
- *Network Statistics* on page 65

### Thread Statistics

Thread statistics describe thread use.

#### Table 11. Thread Statistics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IQ Threads in Use</td>
<td>Number of threads used by the IQ server</td>
<td>No</td>
</tr>
<tr>
<td>IQ Threads Available</td>
<td>Number of threads available in the IQ server</td>
<td>No</td>
</tr>
<tr>
<td>SA Threads in Use</td>
<td>Number of threads used by the SQL Anywhere engine.</td>
<td>No</td>
</tr>
</tbody>
</table>

#### See also
- *Monitoring Performance at the Server Level* on page 56
- *Memory Usage Statistics* on page 57
- *Cache Statistics* on page 58
- *CPU Usage Statistics* on page 59
- *Connection Statistics* on page 61
- *Request Statistics* on page 62
- *Transaction Statistics* on page 63
- *Store I/O Statistics* on page 63
Connection Statistics

Connection statistics display connection activities.

### Table 12. Connection Statistics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Connections</td>
<td>Total number of connections including user and INC connections.</td>
<td>Yes</td>
</tr>
<tr>
<td>User Connections</td>
<td>Number of user connections.</td>
<td>No</td>
</tr>
<tr>
<td>INC Incoming Connections</td>
<td>Number of INC incoming connections</td>
<td>No</td>
</tr>
<tr>
<td>INC Outgoing Connections</td>
<td>Number of INC outgoing connections</td>
<td>No</td>
</tr>
<tr>
<td>User Connections Per Minute</td>
<td>Number of user connections per minute</td>
<td>No</td>
</tr>
<tr>
<td>User Disconnections Per Minute</td>
<td>Number of user disconnections per minute</td>
<td>No</td>
</tr>
</tbody>
</table>

See also

- Monitoring Performance at the Server Level on page 56
- Memory Usage Statistics on page 57
- Cache Statistics on page 58
- CPU Usage Statistics on page 59
- Thread Statistics on page 60
- Request Statistics on page 62
- Transaction Statistics on page 63
- Store I/O Statistics on page 63
- DBspace Usage Statistics on page 64
- Network Statistics on page 65
Request Statistics

Request statistics describe activities devoted to responding to requests from client applications.

Table 13. Request Statistics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests</td>
<td>Number of times per second the server has been entered to allow it to handle a new request or continue processing an existing request.</td>
<td>No</td>
</tr>
<tr>
<td>Unscheduled Requests</td>
<td>Number of requests that are currently queued up waiting for an available server thread.</td>
<td>No</td>
</tr>
<tr>
<td>IQ Waiting Operations</td>
<td>Number of IQ operations waiting for the resource governor</td>
<td>No</td>
</tr>
<tr>
<td>IQ Active Operations</td>
<td>Number of active IQ operations</td>
<td>No</td>
</tr>
</tbody>
</table>

See also
- Monitoring Performance at the Server Level on page 56
- Memory Usage Statistics on page 57
- Cache Statistics on page 58
- CPU Usage Statistics on page 59
- Thread Statistics on page 60
- Connection Statistics on page 61
- Transaction Statistics on page 63
- Store I/O Statistics on page 63
- DBspace Usage Statistics on page 64
- Network Statistics on page 65
Transaction Statistics

Transaction statistics display transaction activity.

Table 14. Transaction Statistics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Transaction Count</td>
<td>Total number of active transactions including user and INC transactions.</td>
<td>No</td>
</tr>
<tr>
<td>User Transaction Count</td>
<td>Number of active user transactions</td>
<td>No</td>
</tr>
<tr>
<td>INC Transaction Count</td>
<td>Number of active INC transactions</td>
<td>No</td>
</tr>
<tr>
<td>Active Load Table Statements</td>
<td>Number of active load table statements</td>
<td>No</td>
</tr>
</tbody>
</table>

See also

- Monitoring Performance at the Server Level on page 56
- Memory Usage Statistics on page 57
- Cache Statistics on page 58
- CPU Usage Statistics on page 59
- Thread Statistics on page 60
- Connection Statistics on page 61
- Request Statistics on page 62
- Store I/O Statistics on page 63
- DBspace Usage Statistics on page 64
- Network Statistics on page 65

Store I/O Statistics

Store I/O statistics describe disk reads and writes.

Table 15. Store I/O Statistics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog Store Disk Reads</td>
<td>Number of kilobytes per second that have been read from the catalog store.</td>
<td>No</td>
</tr>
</tbody>
</table>
### Name
### Description
### Monitored By Default?

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temporary Store Disk Reads</td>
<td>Number of kilobytes per second that have been read from the temporary store.</td>
<td>No</td>
</tr>
<tr>
<td>Main Store Disk Reads</td>
<td>Number of kilobytes per second that have been read from the main store.</td>
<td>No</td>
</tr>
<tr>
<td>Catalog Store Disk Writes</td>
<td>Number of kilobytes per second that have been written to the catalog store.</td>
<td>No</td>
</tr>
<tr>
<td>Temporary Store Disk Writes</td>
<td>Number of kilobytes per second that have been written to the temporary store.</td>
<td>No</td>
</tr>
<tr>
<td>Main Store Disk Writes</td>
<td>Number of kilobytes per second that have been written to the main store.</td>
<td>No</td>
</tr>
</tbody>
</table>

### See also
- Monitoring Performance at the Server Level on page 56
- Memory Usage Statistics on page 57
- Cache Statistics on page 58
- CPU Usage Statistics on page 59
- Thread Statistics on page 60
- Connection Statistics on page 61
- Request Statistics on page 62
- Transaction Statistics on page 63
- DBspace Usage Statistics on page 64
- Network Statistics on page 65

### DBspace Usage Statistics
DBspace usage statistics identify dbspace availability.

### Table 16. DBSpace Usage

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBSpace File Size in Use</td>
<td>DBSpace size in use. There is one such statistic per dbspace.</td>
<td>No</td>
</tr>
</tbody>
</table>
### Network Statistics

Network statistics display network activity.

#### Table 17. Network Statistics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Monitored By Default?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes Received</td>
<td>Number of bytes per second received during client/server communications.</td>
<td>Yes</td>
</tr>
<tr>
<td>Bytes Received Uncompressed</td>
<td>Number of bytes per second that would have been received during client/server communications if compression was disabled.</td>
<td>No</td>
</tr>
<tr>
<td>Bytes Sent</td>
<td>Number of bytes per second sent during client/server communications.</td>
<td>Yes</td>
</tr>
<tr>
<td>Bytes Sent Uncompressed</td>
<td>Number of bytes per second that would have been sent during client/server communications if compression was disabled.</td>
<td>No</td>
</tr>
</tbody>
</table>
### Name | Description | Monitored By Default?
--- | --- | ---
Free Communication Buffers | Number of available network communication buffers. | No
Total Communication Buffers | Total number of network communication buffers. | No

**See also**
- *Monitoring Performance at the Server Level* on page 56
- *Memory Usage Statistics* on page 57
- *Cache Statistics* on page 58
- *CPU Usage Statistics* on page 59
- *Thread Statistics* on page 60
- *Connection Statistics* on page 61
- *Request Statistics* on page 62
- *Transaction Statistics* on page 63
- *Store I/O Statistics* on page 63
- *DBspace Usage Statistics* on page 64

---

**Monitoring the Buffer Caches**

Buffer cache performance is a key factor in overall performance. Buffer cache monitor logs buffer cache, memory, and I/O statistics.

Use the buffer cache monitor to fine-tune main and temp buffer cache memory allocation. If one cache performs significantly more I/O than the other, reallocate some of the memory in small amounts, such as 10 percent of the cache allocation on an iterative basis. After reallocating, rerun the workload and monitor the performance changes.

---

**Starting the Buffer Cache Monitor**

Run the buffer cache monitor from Interactive SQL. Each time you start the monitor it runs as a separate kernel thread within Sybase IQ.

Use this syntax to start the monitor:

```sql
IQ UTILITIES { MAIN | PRIVATE }
    INTO dummy_table_name
    START MONITOR 'monitor_options [ ... ]'
```

**MAIN** starts monitoring of the main buffer cache, for all tables in the IQ Store of the database you are connected to.
PRIVATE starts monitoring of the temp buffer cache, for all tables in the Temporary Store of the database you are connected to.

You need to issue a separate command to monitor each buffer cache. You must keep each of these sessions open while the monitor collects results; a monitor run stops when you close its connection. A connection can run up to a maximum of two monitor runs, one for the main and one for the temp buffer cache.

dummy_table_name can be any Sybase IQ base or temporary table. The table name is required for syntactic compatibility with other IQ UTILITIES commands. It is best to have a table that you use only for monitoring.

To control the directory placement of monitor output files, set the MONITOR_OUTPUT_DIRECTORY option. If this option is not set, the monitor sends output to the same directory as the database. All monitor output files are used for the duration of the monitor runs. They remain after a monitor run has stopped.

Either declare a temporary table for use in monitoring, or create a permanent dummy table when you create a new database, before creating any multiplex query servers. These solutions avoid DDL changes, so that data stays up on query servers during production runs.

**Note:** To simplify monitor use, create a stored procedure to declare the dummy table, specify its output location, and start the monitor.

**Note:** The interval, with two exceptions, applies to each line of output, not to each page. The exceptions are -cache_by_type and -debug, where a new page begins for each display.

### Output Options

Buffer cache monitor output depends on the switches you include with the monitor_options argument.

**-summary**
Displays summary information for both the main and temp buffer caches. If you do not specify any monitor options, you receive a summary report.

**Usage**

```
monitor_options -summary
```

**Output**

<table>
<thead>
<tr>
<th>Output field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>Number of users connected to the buffer cache</td>
</tr>
<tr>
<td>IO</td>
<td>Combined physical reads and writes by the buffer cache</td>
</tr>
</tbody>
</table>
See also

- -cache on page 68
- -cache_by_type on page 70
- -file_suffix on page 70
- -io on page 71
- -bufalloc on page 72
- -contention on page 73
- -threads on page 74
- -interval on page 75
- -append | - truncate on page 76
- -debug on page 76

(cache)
Displays main or temp buffer cache activity in detail. Critical fields are *Finds, HR%, and BWaits.*

Usage

```
monitor_options -cache
```

Output

**Table 19. -cache Output Fields**

<table>
<thead>
<tr>
<th>Output field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Finds</em></td>
<td>Find requests to the buffer cache. If the <em>Finds</em> value suddenly drops to zero and remains zero, the server is deadlocked. When the server has any activity, the <em>Finds</em> value is expected to be non-zero.</td>
</tr>
<tr>
<td><em>Creats</em></td>
<td>Requests to create a page within the database</td>
</tr>
<tr>
<td><em>Dests</em></td>
<td>Requests to destroy a page within the database</td>
</tr>
<tr>
<td><em>Dirty</em></td>
<td>Number of times the buffer was dirtied (modified)</td>
</tr>
<tr>
<td><em>HR%</em></td>
<td>Hit rate, the percentage of above satisfied by the buffer cache without requesting any I/O. The higher the Hit Rate the better, usually 90% - 100% if you set the cache large enough. For a large query, Hit Rate may be low at first, but increase as prefetching starts to work.</td>
</tr>
<tr>
<td><em>BWaits</em></td>
<td>Find requests forced to wait for a busy page (page frame contention). Usually it is low, but is some special cases it may be high. For example, if identical queries are started at the same time, both need the same page, so the second request must wait for the first to get that page from disk.</td>
</tr>
<tr>
<td>Output field</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>ReReads</strong></td>
<td>Approximate number of times the same portion of the store needed to be reread into the cache within the same transaction. Should always be low, but a high number is not important for Sybase IQ 12.4.2 and above.</td>
</tr>
<tr>
<td><strong>FMiss</strong></td>
<td>False misses, number of times the buffer cache needed multiple lookups to find a page in memory. This number should be 0 or very small. If the value is high, it is likely that a rollback occurred, and certain operations needed to be repeated</td>
</tr>
<tr>
<td><strong>Cloned</strong></td>
<td>Number of buffers that Sybase IQ needed to make a new version for a writer, while it had to retain the previous version for concurrent readers. A page only clones if other users are looking at that page.</td>
</tr>
<tr>
<td><strong>Reads/Writes</strong></td>
<td>Physical reads and writes performed by the buffer cache</td>
</tr>
<tr>
<td><strong>PF/PFRead</strong></td>
<td>Prefetch requests and reads done for prefetch.</td>
</tr>
<tr>
<td><strong>GDirty</strong></td>
<td>Number of times the LRU buffer was grabbed dirty and Sybase IQ had to write it out before using it. This value should not be greater than 0 for a long period. If it is, you may need to increase the number of sweeper threads or move the wash marker.</td>
</tr>
<tr>
<td><strong>Pin%</strong></td>
<td>Percentage of pages in the buffer cache in use and locked.</td>
</tr>
<tr>
<td><strong>Dirty%</strong></td>
<td>Percentage of buffer blocks that were modified. Try not to let this value exceed 85-90%; otherwise, GDirty will become greater than 0.</td>
</tr>
</tbody>
</table>

**See also**
- **summary** on page 67
- **cache_by_type** on page 70
- **file_suffix** on page 70
- **io** on page 71
- **bufalloc** on page 72
- **contention** on page 73
- **threads** on page 74
- **interval** on page 75
- **append | truncate** on page 76
- **debug** on page 76
-cache_by_type
Breaks -cache results down by IQ page type. (An exception is the Bwaits column, which shows a total only.) This format is most useful when you need to supply information to Sybase Technical Support.

Usage
monitor_options -cache_by_type

See also
- summary on page 67
- cache on page 68
- file_suffix on page 70
- io on page 71
- bufalloc on page 72
- contention on page 73
- threads on page 74
- interval on page 75
- append / - truncate on page 76
- debug on page 76

-file_suffix
Creates a monitor output file named <dbname>.<connid>-<main_or_temp>-<suffix>. If you do not specify an optional file extension, the file extension defaults to iqmon.

Usage
monitor_options -file_suffix {extension}

See also
- summary on page 67
- cache on page 68
- cache_by_type on page 70
- io on page 71
- bufalloc on page 72
- contention on page 73
- threads on page 74
- interval on page 75
- append / - truncate on page 76
- debug on page 76
-io
Displays main or temp (private) buffer cache I/O rates and compression ratios during the specified interval. These counters represent all activity for the server; the information is not broken out by device.

Usage

monitor_options -io

Output

<table>
<thead>
<tr>
<th>Output field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reads</td>
<td>Physical reads performed by the buffer cache</td>
</tr>
<tr>
<td>Lrd(KB)</td>
<td>Logical kilobytes read in (page size multiplied by the number of requests)</td>
</tr>
<tr>
<td>Prd(KB)</td>
<td>Physical kilobytes read in</td>
</tr>
<tr>
<td>Rratio</td>
<td>Compression ratio of logical to physical data read in, a measure of the efficiency of the compression to disk for reads</td>
</tr>
<tr>
<td>Writes</td>
<td>Physical writes performed by the buffer cache</td>
</tr>
<tr>
<td>Lwrt(KB)</td>
<td>Logical kilobytes written</td>
</tr>
<tr>
<td>Pwrt(KB)</td>
<td>Physical kilobytes written</td>
</tr>
<tr>
<td>Wratio</td>
<td>Compression ratio of logical to physical data written</td>
</tr>
</tbody>
</table>

See also
- summary on page 67
- cache on page 68
- cache_by_type on page 70
- file_suffix on page 70
- bufalloc on page 72
- contention on page 73
- threads on page 74
- interval on page 75
- append / - truncate on page 76
- debug on page 76
-bufalloc
Displays information on the main or temp buffer allocator, which reserves space in the buffer cache for objects like sorts, hashes, and bitmaps.

Usage
monitor_options -bufalloc

Output

Table 21. -bufalloc Output Fields

<table>
<thead>
<tr>
<th>Output field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OU</td>
<td>User_Resource_Reservation option setting (formerly Optimize_For_This_Man Many_Users)</td>
</tr>
<tr>
<td>AU</td>
<td>Current number of active users</td>
</tr>
<tr>
<td>MaxBuf</td>
<td>Number buffers under control of the buffer allocator</td>
</tr>
<tr>
<td>Avail</td>
<td>Number of currently available buffers for pin quota allocation</td>
</tr>
<tr>
<td>AvPF</td>
<td>Number of currently available buffers for prefetch quota allocation</td>
</tr>
<tr>
<td>Slots</td>
<td>Number of currently registered objects using buffer cache quota</td>
</tr>
<tr>
<td>PinUser</td>
<td>Number of objects (for example, hash, sort, and B-tree objects) using pin quota</td>
</tr>
<tr>
<td>PFUsr</td>
<td>Number of objects using prefetch quota</td>
</tr>
<tr>
<td>Posted</td>
<td>Number of objects that are pre-planned users of quota</td>
</tr>
<tr>
<td>UnPost</td>
<td>Number of objects that are ad hoc quota users</td>
</tr>
<tr>
<td>Locks</td>
<td>Number of mutex locks taken on the buffer allocator</td>
</tr>
<tr>
<td>Waits</td>
<td>Number of times a thread had to wait for the lock</td>
</tr>
</tbody>
</table>

See also

- summary on page 67
- cache on page 68
- cache_by_type on page 70
- file_suffix on page 70
- io on page 71
- contention on page 73
- threads on page 74
- interval on page 75
-contention
Displays many key buffer cache and memory manager locks. These lock and mutex counters show the activity within the buffer cache and heap memory and how quickly these locks were resolved. Timeout numbers that exceed 20% indicate a problem.

Usage
monitor_options -contention

Output

Table 22. -contention Output Fields

<table>
<thead>
<tr>
<th>Output field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU</td>
<td>Current number of active users</td>
</tr>
<tr>
<td>LRULks</td>
<td>Number times the LRU was locked (repeated for the temp cache)</td>
</tr>
<tr>
<td>woTO</td>
<td>Number times lock was granted without timeout (repeated for the temp cache)</td>
</tr>
<tr>
<td>Loops</td>
<td>Number times Sybase IQ retried before lock was granted (repeated for the temp cache)</td>
</tr>
<tr>
<td>TOs</td>
<td>Number of times Sybase IQ timed out and had to wait for the lock (repeated for the temp cache)</td>
</tr>
<tr>
<td>BWaits</td>
<td>Number of Busy Waits for a buffer in the cache (repeated for the temp cache)</td>
</tr>
<tr>
<td>IOLock</td>
<td>Number of times Sybase IQ locked the compressed I/O pool (repeated for the temp cache); can be ignored</td>
</tr>
<tr>
<td>IOWait</td>
<td>Number of times Sybase IQ had to wait for the lock on the compressed I/O pool (repeated for the temp cache); can be ignored</td>
</tr>
<tr>
<td>HTLock</td>
<td>Number of times Sybase IQ locked the block maps hash table (repeated for the temp cache)</td>
</tr>
<tr>
<td>HTWait</td>
<td>Number of times Sybase IQ had to wait for the block maps hash table (repeated for the temp cache); HTLock and HTWait indicate how many block maps you are using</td>
</tr>
<tr>
<td>FLLock</td>
<td>Number of times Sybase IQ had to lock the free list (repeated for the temp cache)</td>
</tr>
</tbody>
</table>
### Output field

<table>
<thead>
<tr>
<th>Output field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FLWait</strong></td>
<td>Number of times Sybase IQ had to wait for the lock on the free list (repeated for the temp cache)</td>
</tr>
<tr>
<td><strong>MemLks</strong></td>
<td>Number of times Sybase IQ took the memory manager (heap) lock</td>
</tr>
<tr>
<td><strong>MemWts</strong></td>
<td>Number of times Sybase IQ had to wait for the memory manager lock</td>
</tr>
</tbody>
</table>

**Note:** Due to operating system improvements, Sybase IQ no longer uses spin locks. As a result, the woTO, Loops, and TOs statistics are rarely used.

### See also

- `-summary` on page 67
- `-cache` on page 68
- `-cache_by_type` on page 70
- `-file_suffix` on page 70
- `-io` on page 71
- `-bufalloc` on page 72
- `-threads` on page 74
- `-interval` on page 75
- `-append` / `-truncate` on page 76
- `-debug` on page 76

### `-threads`

Displays the processing thread manager counts. Values are server-wide (i.e., it does not matter whether you select this option for main or private). They represent new events after the last page of the report.

**Usage**

```
monitor_options -threads
```

**Output**

### Table 23. `-threads` Output Fields

<table>
<thead>
<tr>
<th>Output field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cpus</strong></td>
<td>Number of CPUs Sybase IQ is using; this may be less than the number on the system</td>
</tr>
<tr>
<td><strong>Limit</strong></td>
<td>Maximum number of threads Sybase IQ can use</td>
</tr>
</tbody>
</table>
### Output field

<table>
<thead>
<tr>
<th>Output field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$NTeams$</td>
<td>Number of thread teams currently in use</td>
</tr>
<tr>
<td>$MaxTms$</td>
<td>Largest number of teams that has ever been in use</td>
</tr>
<tr>
<td>$NThrds$</td>
<td>Current number of existing threads</td>
</tr>
<tr>
<td>$Resrvd$</td>
<td>Number of threads reserved for system (connection) use</td>
</tr>
<tr>
<td>$Free$</td>
<td>Number of threads available for assignment. Monitor this value if it is very low, it indicates thread starvation</td>
</tr>
<tr>
<td>$Locks$</td>
<td>Number of locks taken on the thread manager</td>
</tr>
<tr>
<td>$Waits$</td>
<td>Number of times Sybase IQ had to wait for the lock on the thread manager</td>
</tr>
</tbody>
</table>

### See also
- `-summary` on page 67
- `-cache` on page 68
- `-cache_by_type` on page 70
- `-file_suffix` on page 70
- `-io` on page 71
- `-bufalloc` on page 72
- `-contention` on page 73
- `-interval` on page 75
- `-append` / `-truncate` on page 76
- `-debug` on page 76

### -interval

Specifies the reporting interval in seconds. The default is every 60 seconds. The minimum is every 2 seconds. You can usually get useful results by running the monitor at the default interval during a query or time of day with performance problems. Short intervals may not give meaningful results. Intervals should be proportional to the job time; one minute is generally more than enough.

### Usage

```
monitor_options -interval
```

### Output

The first display shows counters from the start of the server. Subsequent displays show the difference from the previous display.
See also

- *summary* on page 67
- *cache* on page 68
- *cache_by_type* on page 70
- *file_suffix* on page 70
- *io* on page 71
- *bufalloc* on page 72
- *contention* on page 73
- *threads* on page 74
- *append | - truncate* on page 76
- *debug* on page 76

**-append | - truncate**

Append or truncate output to existing output file. Truncate is the default.

**Usage**

```
monitor_options -append | -truncate
```

See also

- *summary* on page 67
- *cache* on page 68
- *cache_by_type* on page 70
- *file_suffix* on page 70
- *io* on page 71
- *bufalloc* on page 72
- *contention* on page 73
- *threads* on page 74
- *interval* on page 75
- *debug* on page 76

**-debug**

Displays all information available to the performance monitor, whether or not there is a standard display mode that covers the same information. *-debug* is used mainly to supply information to Sybase Technical Support.

**Usage**

```
monitor_options -debug
```
Output
The top of the page is an array of statistics broken down by disk block type. This is followed by other buffer cache statistics, memory manager statistics, thread manager statistics, free list statistics, CPU utilization, and finally buffer allocator statistics.

The buffer allocator statistics are then broken down by client type (hash, sort, and so on) and a histogram of the most recent buffer allocations is displayed. Memory allocations indicate how much is allocated after the last page of the report.

See also
• -summary on page 67
• -cache on page 68
• -cache_by_type on page 70
• -file_suffix on page 70
• -io on page 71
• -bufalloc on page 72
• -contention on page 73
• -threads on page 74
• -interval on page 75
• -append / - truncate on page 76

Checking Results While the Monitor Runs
On UNIX systems, you can watch monitor output as queries are running.

For example, you could start the monitor using the following command:

```
$ iq utilities main into monitor_tab
start monitor "-cache -interval 2 -file_suffix iqmon"
```

This command sends output to an ASCII file with the name `dbname.conn#-[main|temp]-iqmon`. So, for the database iqdemo, results would be sent to `iqdemo.2-main-iqmon`.

To watch results, issue the following command at the system prompt:

```
$ tail -f iqdemo.2-main-iqmon
```

Stopping the Buffer Cache Monitor
The command you use to stop a monitor run is similar to the one you use to start it, except that you do not need to specify any options.

Use this syntax to stop the Sybase IQ buffer cache monitor:

```
IQ UTILITIES { MAIN | PRIVATE }
    INTO dummy_table_name STOP MONITOR
```
Note: In order for certain option settings to take effect you must restart the database. If the monitor is running you need to shut it down so that the database can be restarted.

Examining and Saving Monitor Results
Buffer cache monitor logs the results of each run.

The default names of the logs:

- `dbname.connection#-main-iqmon` for main buffer cache results
- `dbname.connection#-temp-iqmon` for temp buffer cache results

The prefix `dbname.connection#` represents your database name and connection number. If you see more than one connection number and are uncertain which is yours, you can run the Catalog stored procedure `sa_conn_info`. This procedure displays the connection number, user ID, and other information for each active connection to the database.

You can use the `-file_suffix` parameter on the IQ UTILITIES command to change the suffix `iqmon` to a suffix of your choice.

To see the results of a monitor run, use a text editor or any other method you would normally use to display or print a file.

When you run the monitor again from the same database and connection number, by default it overwrites the previous results. If you need to save the results of a monitor run, copy the file to another location before starting the monitor again from the same database or use the `-append` option.

Buffer Cache Structure
Changing the CACHE_PARTITIONS value may improve load or query performance in a multi-CPU configuration.

Sybase IQ automatically calculates the number of cache partitions for the buffer cache according to the number of CPUs on your system. If load or query performance in a multi-CPU configuration is slower than expected, you may be able to improve it by changing the value of the CACHE_PARTITIONS database option. For details, see CACHE_PARTITIONS option in Reference: Statements and Options.

As buffers approach the Least Recently Used (LRU) end of the cache, they pass a wash marker. Sybase IQ writes the oldest pages—those past the wash marker—out to disk so that the cache space they occupy can be reused. A team of Sybase IQ processing threads, called sweeper threads, sweeps (writes) out the oldest buffers.

When Sybase IQ needs to read a page of data into the cache, it grabs the LRU buffer. If the buffer is still “dirty” (modified) it must first be written to disk. The Gdirty column in the monitor `-cache` report shows the number of times the LRU buffer was grabbed dirty and Sybase IQ had to write it out before using it.
Avoid Buffer Manager Thrashing

Thrashing occurs when the system must write a dirty page before it can read a requested page, which drastically slows down the system. For optimum performance, always allocate enough cache to allow the page writers to keep up with the free space demand.

Buffer Cache Thrashing
Buffer cache thrashing is similar to system thrashing, and occurs when there are not enough clean buffers available for reads. This causes the same kind of ‘write first then read’ delay in the cache, and can happen when the buffer cache is not large enough to accommodate all of the objects referenced in a query.

To eliminate buffer cache thrashing, you must allocate more memory for the buffer caches. Do not over allocate the buffer caches. Allocating too much memory can induce system thrashing by allocating memory for the database buffer cache. In extreme cases, allocating too much memory can introduce multiple levels of thrashing without solving the buffer cache thrashing problem.

Another more subtle form of buffer cache thrashing can occur in multiuser contexts or when skew or uncertainty caused by query complexity causes the optimizer to choose a HASH algorithm in a circumstance where the HASH object needed to be built with significantly larger number of values than fits in the cache available to the query.

Setting Buffer Sizes
When you set buffer sizes, keep in mind the following trade-off:

• If the Sybase IQ buffer cache is too large, the operating system is forced to page as Sybase IQ tries to use all of that memory.
• If the Sybase IQ buffer cache is too small, then Sybase IQ thrashes because it cannot fit enough of the query data into the cache.

If you are experiencing dramatic performance problems, you should monitor paging to determine if thrashing is a problem. If so, then reset your buffer sizes.

Queries and Hash Algorithms
If you monitor paging and determine that thrashing is a problem, you can also limit the amount of thrashing during the execution of a statement which includes a query that involves hash algorithms. Adjusting the HASH_THRASHING_PERCENT database option controls the percentage of hard disk I/Os allowed before the statement is rolled back and an error is returned.
The default value of `HASH_THRASHING_PERCENT` is 10%. Increasing `HASH_THRASHING_PERCENT` permits more paging to disk before a rollback and decreasing `HASH_THRASHING_PERCENT` permits less paging before a rollback.

Queries involving hash algorithms that executed in earlier versions of Sybase IQ may now be rolled back when the default `HASH_THRASHING_PERCENT` limit is reached. Sybase IQ reports the error `Hash insert thrashing detected` or `Hash find thrashing detected`. Take one or more of the following actions to provide the query with the resources required for execution:

- Relax the paging restriction by increasing the value of `HASH_THRASHING_PERCENT`.
- Increase the size of the temporary cache (DBA only). Keep in mind that increasing the size of the temporary cache requires an equal size reduction in main cache allocation to prevent the possibility of system thrashing.
- Attempt to identify and alleviate why Sybase IQ is misestimating one or more hash sizes for this statement. For example, check that all columns that need an LF or HG index have one. Also consider if a multicolon index is appropriate.
- Decrease the value of the database option `HASH_PINNABLE_CACHE_PERCENT`.

For more information on these database options, see the sections “`HASH_THRASHING_PERCENT` option” and “`HASH_PINNABLE_CACHE_PERCENT` option” in Reference: Statements and Options.

To identify possible problems with a query, generate a query plan by running the query with the temporary database options `QUERY_PLAN = 'ON'` and `QUERY_DETAIL = 'ON'`, then examine the estimates in the query plan. The generated query plan is in the message log file.

**Monitoring Paging on Windows Systems**

Use the Windows Performance tool to monitor paging and object memory.

To access System Monitor, select the object Logical Disk, the instance of the disk containing the file `PAGEFILE.SYS`, and the counter Disk Transfers/Sec. Put the Windows page files on different disks than your database dbspace devices. You can also monitor the Object Memory and the counter Pages/Sec. However, this value is the sum of all memory faults which includes both soft and hard faults.

**See also**

- *Monitoring Paging on UNIX-like Operating Systems* on page 81
Monitoring Paging on UNIX-like Operating Systems

Use `vmstat`, `top`, or `topas` to monitor system activity such as paging.

<table>
<thead>
<tr>
<th>Command</th>
<th>Platform</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vmstat</code></td>
<td>Solaris, Linux, HP-UX</td>
<td><code>vmstat</code> displays virtual memory statistics.</td>
</tr>
<tr>
<td><code>top</code></td>
<td>Solaris, Linux, HP-UX</td>
<td><code>top</code> displays top CPU processor activities.</td>
</tr>
<tr>
<td><code>topas</code></td>
<td>AIX</td>
<td><code>topas</code> generates local system statistics.</td>
</tr>
</tbody>
</table>

Note: See your operating system documentation for syntax and options.

See also
- Monitoring Paging on Windows Systems on page 80

Buffer Cache Monitor Checklist

Review this checklist to adjust cache behavior that falls outside the normal range.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Normal behavior</th>
<th>Behavior that needs adjusting</th>
<th>Recommended action</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR% (Cache hit rate)</td>
<td>Above 90%. For individual internal data structures like garray, barray, bitmap (bm), hash object, sort object, variable-length btree (btreev), fixed-length btree (btreef), bit vector (bv), dbext, dbid, vdo, store, checkpoint block (ckpt), the hit rate should be above 90% while a query runs. It may be below 90% at first. Once prefetch starts working (PF or PrefetchReqs &gt; 0), the hit rate should gradually grow to above 90%.</td>
<td>Hit rate below 90% after prefetch is working. <strong>Note:</strong> Some objects do not do prefetching, so their hit rate may be low normally.</td>
<td>Try rebalancing the cache sizes of main versus temp by adjusting <code>-iqmc</code> and <code>-iqtc</code>. Also try increasing the number of prefetch threads by adjusting <code>PRE_FETCH_THREADS_PERCENT</code> option.</td>
</tr>
<tr>
<td>Statistic</td>
<td>Normal behavior</td>
<td>Behavior that needs adjusting</td>
<td>Recommended action</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Gdirty (Grabbed Dirty)            | 0 in a system with a modest cache size (< 10GB). | GDirty > 0  
**Note:** Sweeper threads are activated only when the number of dirty pages reaches a certain percentage of the wash area. If GDirty/GrabbedDirty is above 0 and the I/O rate (Writes) is low, the system may simply be lightly loaded, and no action is necessary. | Adjust `SWEEPER_THREADS_PERCENT` option (default 10%) or `WASH_AREA_BUFFERS_PERCENT` option (default 20%) to increase the size of the wash area. |
| BWaits (Buffer Busy Waits)        | 0                                    | Persistently > 0, indicating that multiple jobs are colliding over the same buffers.           | If the I/O rate (Writes) is high, Busy Waits may be caused by cache thrashing.  
Check Hit Rate in the cache report to see if you need to rebalance main versus temp cache.  
If a batch job is starting a number of nearly identical queries at the same time, try staggering the start times. |
<p>| LRU Waits (LRUNum TimeOuts percentage in debug report) | 20% or less                          | &gt; 20%, which indicates a serious contention problem.                                        | Check the operating system patch level and other environment settings. This problem tends to be an O.S. issue. |
| IOWait (IONumWaits)               | 10% or lower                         | &gt; 10%                                                                                         | Check for disk errors or I/O retries                                              |</p>
<table>
<thead>
<tr>
<th>Statistic</th>
<th>Normal behavior</th>
<th>Behavior that needs adjusting</th>
<th>Recommended action</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLWait (FLMutex-Waits)</td>
<td>20% or lower</td>
<td>&gt; 20%</td>
<td>Check the dbspace configuration: Is the database almost out of space? Is DISK_STRIPING ON? Does sp_iqcheckdb report fragmentation greater than 15%?</td>
</tr>
<tr>
<td>HTWait</td>
<td>10% or lower</td>
<td>&gt; 10%</td>
<td>Contact Sybase Technical Support.</td>
</tr>
<tr>
<td>MemWts (MemNti-mesWaited)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(PFMgrCond VarWaits)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Performance and Tuning Guide  
83
## Monitoring and Tuning Performance

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Normal behavior</th>
<th>Behavior that needs adjusting</th>
<th>Recommended action</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time (CPU Sys Seconds, CPU Total Seconds, in debug report)</td>
<td>CPU Sys Seconds &lt; 20%</td>
<td>CPU Sys Seconds &gt; 20% If CPU Total Seconds also reports LOW utilization, and there are enough jobs that the system is busy, the cache may be thrashing or parallelism may be lost.</td>
<td>Adjust -iqgovern to reduce allowed total number of concurrent queries. Check Hit Rate and I/O Rates in the cache report for cache thrashing. Also check if hash object is thrashing by looking at the hit rate of the has object in cache_by_type (or debug) report: is it &lt;90% while the I/O rate (Writes) is high? Check query plans for attempted parallelism. Were enough threads available? Does the system have a very large number of CPUs? Strategies such as multiplex configuration may be necessary.</td>
</tr>
<tr>
<td>InUse% (Buffers in use)</td>
<td>At or near 100% except during startup</td>
<td>Less than about 100%</td>
<td>The buffer cache may be too large. Try rebalancing the cache sizes of main versus temp by adjusting -iqmc and -iqtc.</td>
</tr>
<tr>
<td>Statistic</td>
<td>Normal behavior</td>
<td>Behavior that needs adjusting</td>
<td>Recommended action</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Pin% (Pinned buffers)</td>
<td>&lt; 90%</td>
<td>&gt; 90 to 95%, indicating system is dangerously close to an Out of Buffers condition, which would cause transactions to roll back</td>
<td>Try rebalancing the cache sizes of main versus temp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If rebalancing buffer cache sizes is not possible, try reducing <code>-iq-govern</code> to limit the number of jobs running concurrently.</td>
</tr>
<tr>
<td>Free threads</td>
<td>Free &gt; Resvd</td>
<td>If the number of free threads drops to the reserved count, the system may be thread starved.</td>
<td>Try one of the following:</td>
</tr>
<tr>
<td>(ThrNum-Free)</td>
<td></td>
<td></td>
<td>Increase the number of threads by setting <code>-iqmt</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reduce thread-related options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>MAX_IQ_THREAD S_PER_CONNECTION,</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>MAX_IQ_THREAD S_PER_TEAM</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Restrict query engine resource allocations by setting <code>USER_RESOURCEReservation</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Limit the number of jobs by setting <code>-iqgovern</code>.</td>
</tr>
<tr>
<td>FLOutOf-Space (debug only)</td>
<td>0, indicating that the free list for this store is not full; unallocated pages are available</td>
<td>1, indicating that this store (main or temporary) is fully allocated</td>
<td>Add more dbspace to that store</td>
</tr>
</tbody>
</table>
System Utilities to Monitor CPU Use

OS-specific utilities are available to monitor CPU usage.

Table 26. OS-Specific Monitoring Utilities

<table>
<thead>
<tr>
<th>OS</th>
<th>Utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX</td>
<td>top (Solaris, Linux, HP-UX), topas (IBM-AIX)</td>
<td>Provides an ongoing look at processor activity in real time.</td>
</tr>
<tr>
<td></td>
<td>ps</td>
<td>Reports process status.</td>
</tr>
<tr>
<td></td>
<td>vmstat</td>
<td>Displays information about system processes, memory, paging, block IQ, traps, and CPU activity.</td>
</tr>
<tr>
<td></td>
<td>iostat -x</td>
<td>Displays disk subsystem information.</td>
</tr>
<tr>
<td>Windows</td>
<td>System Monitor Task Manager</td>
<td>Provide detailed information about computer performance and running applications, processes, CPU usage, and other system services.</td>
</tr>
</tbody>
</table>
Optimizing Queries and Deletions

Recommendations to help you plan, structure, and control your queries.

Tips for Structuring Queries

Improving query structures can make your queries run faster.

- In some cases, command statements that include subqueries can also be formulated as joins and may run faster.
- If you group on multiple columns in a `GROUP BY` clause, list the columns by descending order by number of unique values if you can. This will give you the best query performance.
- You can improve performance by using an additional column to store frequently calculated results.

See also
- Planning Queries on page 89
- Controlling Query Processing on page 92
- Optimizing Delete Operations on page 96

Enhancing ORDER BY Query Performance

Using multicolumn HG indexes can enhance the performance of `ORDER BY` queries.

You can use multicolumn HG indexes to enhance the performance of `ORDER BY` queries with reference to multiple columns in a single table query. This change is transparent to users, but improves query performance.

Queries with multiple columns in the `ORDER BY` clause may run faster using multicolumn HG indexes. For example, if the user has multicolumn index `HG(x, y, z)` on table T, then this index is used for ordered projection:

```sql
SELECT abs (x) FROM T
ORDER BY x, y
```

In the above example, the HG index vertically projects x and y in sorted order.

If the `ROWID()` function is in the `SELECT` list expressions, multicolumn HG indexes are also used. For example:

```sql
SELECT rowid()+x, z FROM T
ORDER BY x, y, z
```
If `ROWID()` is present at the end of an `ORDER BY` list, and if the columns of that list—except for `ROWID()`—exist within the index, and the ordering keys match the leading `HG` columns in order, multicolumn indexes are used for the query. For example:

```sql
SELECT z, y FROM T
ORDER BY x, y, z, ROWID()
```

**See also**
- *Improved Subquery Performance* on page 88
- *Using Caching Methods* on page 88

### Improved Subquery Performance

Use `SUBQUERY_FLATTENING_PREFERENCE` and `SUBQUERY_FLATTENING_PERCENT` to control subquery flattening.

Subquery flattening is an optimization technique in which the optimizer rewrites a query containing a subquery into a query that uses a join. Sybase IQ flattens many but not all subqueries. Use `SUBQUERY_FLATTENING_PREFERENCE` and `SUBQUERY_FLATTENING_PERCENT` to control when the optimizer chooses to use this optimization.

The `FLATTEN_SUBQUERIES` option has been deprecated in Sybase IQ 15.0.

**See also**
- *Enhancing ORDER BY Query Performance* on page 87
- *Using Caching Methods* on page 88

### Using Caching Methods

Set the `SUBQUERY_CACHING_PREFERENCE` option to choose caching methods for a correlated subquery.

A correlated subquery contains references to one or more tables outside of the subquery and is re-executed each time the value in the referenced column changes. Use the `SUBQUERY_CACHING_PREFERENCE` option to choose caching methods for executing the correlated subquery.

**See also**
- *Enhancing ORDER BY Query Performance* on page 87
- *Improved Subquery Performance* on page 88
Planning Queries

Generating a query plan can help you understand the execution plan developed by the optimizer.

If you have created the right indexes, the Sybase IQ query optimizer can usually execute queries in the most efficient way - even if you have not used the most effective syntax.

Before it executes any query, the Sybase IQ query optimizer creates a query plan. Sybase IQ helps you evaluate queries by letting you examine and influence the query plan, using the options described in the sections that follow. For details of how to specify these options, see Reference: Statements and Options.

See also
• Tips for Structuring Queries on page 87
• Controlling Query Processing on page 92
• Optimizing Delete Operations on page 96

Query Evaluation Options

Setting the appropriate options helps you evaluate the query plan.

- **INDEX_ADVISOR** – When set **ON**, the index advisor prints index recommendations as part of the Sybase IQ query plan or as a separate message in the Sybase IQ message log file if query plans are not enabled. These messages begin with the string “Index Advisor:” and you can use that string to search and filter them from a Sybase IQ message file. This option outputs messages in **OWNER.TABLE.COLUMN** format and is **OFF** by default.

  See also the “sp_iqindexadvice procedure” in “System Procedures” in the Reference: Building Blocks, Tables, and Procedures.

- **INDEX_ADVISOR_MAX_ROWS** – Used to limit the number of messages stored by the index advisor. Once the specified limit has been reached, the **INDEX_ADVISOR** will not store new advice. It will, however, continue to update count and timestamps for existing advice.

- **NOEXEC** – When set **ON**, Sybase IQ produces a query plan but does not execute the entire query. When the **EARLY_PREDICATE_EXECUTION** option is **ON**, some portions of a query are still executed.

  If **EARLY_PREDICATE_EXECUTION** is **OFF**, the query plan may be very different than when the query is run normally, so turning it **OFF** is not recommended.

- **QUERY_DETAIL** – When this option and either **QUERY_PLAN** or **QUERY_PLAN_AS_HTML** are both **ON**, Sybase IQ displays additional information about the query when producing its query plan. When **QUERY_PLAN** and **QUERY_PLAN_AS_HTML** are **OFF**, this option is ignored.
• **QUERY_PLAN** – When set ON (the default), Sybase IQ produces messages about queries. These include messages about using join indexes, about the join order, and about join algorithms for the queries.

• **QUERY_PLAN_TEXT_ACCESS** – When this option is turned ON, you can view, save, and print IQ query plans from the Interactive SQL client. When **QUERY_PLAN_ACCESS_FROM_CLIENT** is turned OFF, query plans are not cached, and other query plan-related database options have no affect on the query plan display from the Interactive SQL client. This option is OFF by default.

See “GRAPHICAL_PLAN function [String]” and “HTML_PLAN function [String]” in Reference: Building Blocks, Tables, and Procedures.

• **QUERY_PLAN_AFTER_RUN** – When set ON, the query plan is printed after the query has finished running. This allows the plan to include additional information, such as the actual number of rows passed on from each node of the query. In order for this option to work, **QUERY_PLAN** must be ON. This option is OFF by default.

• **QUERY_PLAN_AS_HTML** – Produces a graphical query plan in HTML format for viewing in a Web browser. Hyperlinks between nodes make the HTML format much easier to use than the text format in the .iqmsg file. Use the QUERY_NAME option to include the query name in the file name for the query plan. This option is OFF by default.

• **QUERY_PLAN_TEXT_CACHING** – Gives users a mechanism to control resources for caching plans. With this option OFF (the default), the query plan is not cached for that user connection.

If the **QUERY_PLAN_TEXT_ACCESS** option is turned OFF for a user, the query plan is not cached for the connections from that user, no matter how **QUERY_PLAN_TEXT_CACHING** is set.

See also “GRAPHICAL_PLAN function [String]” and “HTML_PLAN function [String]” in Reference: Building Blocks, Tables, and Procedures.

• **QUERY_TIMING** – Controls the collection of timing statistics on subqueries and some other repetitive functions in the query engine. Normally it should be OFF (the default) because for very short correlated subqueries the cost of timing every subquery execution can be very expensive in terms of performance.

**Note:** Query plans can add a lot of text to your .iqmsg file. When **QUERY_PLAN** is ON, and especially if **QUERY_DETAIL** is ON, you might want to enable message log wrapping or message log archiving to avoid filling up your message log file. For details, see “Message log wrapping” in “Overview of Sybase IQ System Administration” of the System Administration Guide: Volume 1.

**See also**

• *The Query Tree* on page 91
The Query Tree

A query tree represents the query’s data flow.

The query tree consists of nodes. Each node represents a stage of work. The lowest nodes on the tree are leaf nodes. Each leaf node represents a table in the query.

At the top of the plan is the root of the operator tree. Information flows up from the tables and through any operators representing joins, sorts, filters, stores, aggregation, and subqueries.

See also

- Query Evaluation Options on page 89
- Using Query Plans on page 91

Using Query Plans

Set the QUERY_PLAN_AS_HTML option to generate an HTML version of the query plan that you can view this file in a Web browser.

In the HTML query plan, each node in the tree is a hyperlink to the details. Each box is hyperlinked to the tree. You can click on any node to navigate quickly through the plan.

Users can display, print, and save query plans in Interactive SQL plan window instead of accessing the .iqmsg file or query plan files on the server.

SQL functions GRAPHICAL_PLAN and HTML_PLAN return IQ query plans in XML and HTML format, respectively, as a string result set. Database options QUERY_PLAN_TEXT_ACCESS and QUERY_PLAN_TEXT_CACHING control the behavior of the new functions.

View query plans from the Interactive SQL plan window in the following ways:

- Execute the query and open the plan window. Depending on the plan type you selected from the Plan option (Tools > Options > Plan), the appropriate plan displays in the plan window.
  
  The IQ query plan displays only if the GRAPHICAL_PLAN option is selected. Other plans return the error message, “Plan type is not supported.”

- Enter the query in the SQL statements window and select from the menu SQL > Get Plan. Depending on the plan type you selected from the Plan option (Tools > Options > Plan), the appropriate plan displays in the plan window.
  
  The IQ query plan displays only if the GRAPHICAL_PLAN option is selected. Other plans return the error message, “Plan type is not supported.”

- Use the SQL functions, GRAPHICAL_PLAN and HTML_PLAN, to return the query plan as a string result.

To access query plans, use the SQL functions, GRAPHICAL_PLAN and HTML_PLAN, for the following queries: SELECT, UPDATE, DELETE, INSERT SELECT, and SELECT INTO.
To save query plans from Interactive SQL, use `GRAPHICAL_PLAN` or `HTML_PLAN` to retrieve the query plan and save the output to a file using the `OUTPUT` statement.

To view saved plans, select File > Open from the Interactive SQL client menu and navigate to the directory where you saved your plan. You can also print plans displayed on the plan window by selecting File > Print.

See “GRAPHICAL_PLAN function [String]” and “HTML_PLAN function [String]” in Reference: Building Blocks, Tables, and Procedures for details. For the options that support these query plan functions, see “QUERY_PLAN_TEXT_ACCESS option” and “QUERY_PLAN_TEXT_CACHING option” in Reference: Statements and Options.

**See also**
- Query Evaluation Options on page 89
- The Query Tree on page 91

## Controlling Query Processing

Any user can set limits on the amount of time spent processing a particular query. Users with DBA privileges can give certain users’ queries priority over others, or change processing algorithms to influence the speed of query processing.

**See also**
- Tips for Structuring Queries on page 87
- Planning Queries on page 89
- Optimizing Delete Operations on page 96

### Setting Query Time Limits

Set the `MAX_QUERY_TIME` option to limit the time a query can run. If a query takes longer to execute than the `MAX_QUERY_TIME`, Sybase IQ stops the query with an appropriate error.

**Note:** Sybase IQ truncates all decimal `option-value` settings to integer values. For example, the value 3.8 is truncated to 3.

**See also**
- Setting Query Priority on page 93
- Setting Query Optimization Options on page 93
- Setting User-Supplied Condition Hints on page 94
- Monitoring Workloads on page 95
Setting Query Priority

Setting query priority options assigns query processing priorities by user.

Queries waiting in queue for processing are queued to run in order of the priority of the user who submitted the query, followed by the order in which the query was submitted. No queries are run from a lower priority queue until higher priority queries have all been executed.

The following options assign queries a processing priority by user.

- IQGOVERN_PRIORITY – Assigns a numeric priority (1, 2, or 3, with 1 being the highest) to queries waiting in the processing queue.
- IQGOVERN_MAX_PRIORITY – Allows the DBA to set an upper boundary on IQGOVERN_PRIORITY for a user or a group.
- IQ_GOVERN_PRIORITY_TIME – Allows high priority users to start if a high priority (priority 1) query has been waiting in the -iqgovern queue for more than a designated amount of time.

To check the priority of a query, check the IQGovernPriority attribute returned by the sp_iqcontext stored procedure.

See also

- Setting Query Time Limits on page 92
- Setting Query Optimization Options on page 93
- Setting User-Supplied Condition Hints on page 94
- Monitoring Workloads on page 95

Setting Query Optimization Options

Optimization options affect query processing speed.

- AGGREGATION_PREFERENCE – Controls the choice of algorithms for processing an aggregate (GROUP BY, DISTINCT, SET functions). This option is designed primarily for internal use; do not use it unless you are an experienced database administrator.
- DEFAULT_HAVING_SELECTIVITY_PPM – Sets the selectivity for all HAVING predicates in a query, overriding optimizer estimates for the number of rows that will be filtered by the HAVING clause.
- DEFAULT_LIKE_MATCH_SELECTIVITY_PPM – Sets the default selectivity for generic LIKE predicates, for example, LIKE 'string%string' where % is a wildcard character. The optimizer relies on this option when other selectivity information is not available and the match string does not start with a set of constant characters followed by a single wildcard.
- DEFAULT_LIKE_RANGE_SELECTIVITY_PPM – Sets the default selectivity for leading constant LIKE predicates, of the form LIKE 'string%' where the match
string is a set of constant characters followed by a single wildcard character (%). The optimizer relies on this option when other selectivity information is not available.

- **MAX_HASH.Rows** – Sets the maximum estimated number of rows the query optimizer will consider for a hash algorithm. The default is 2,500,000 rows. For example, if there is a join between two tables, and the estimated number of rows entering the join from both tables exceeds this option value, the optimizer will not consider a hash join. On systems with more than 50MB per user of TEMP_CACHE_MEMORY_MB, you may want to consider a higher value for this option.

- **MAX_JOIN_ENUMERATION** – Sets the maximum number of tables to be optimized for join order after optimizer simplifications have been applied. Normally you should not need to set this option.

**See also**
- *Setting Query Time Limits* on page 92
- *Setting Query Priority* on page 93
- *Setting User-Supplied Condition Hints* on page 94
- *Monitoring Workloads* on page 95

**Setting User-Supplied Condition Hints**

Selectivity hints help the optimizer choose an appropriate query strategy.

The Sybase IQ query optimizer uses information from available indexes to select an appropriate strategy for executing a query. For each condition in the query, the optimizer decides whether the condition can be executed using indexes, and if so, the optimizer chooses which index and in what order with respect to the other conditions on that table. The most important factor in these decisions is the selectivity of the condition; that is, the fraction of the table’s rows that satisfy that condition.

The optimizer normally decides without user intervention, and it generally makes optimal decisions. In some situations, however, the optimizer might not be able to accurately determine the selectivity of a condition before it has been executed. These situations normally occur only where either the condition is on a column with no appropriate index available, or where the condition involves some arithmetic or function expression and is, therefore, too complex for the optimizer to accurately estimate.

For syntax, parameters, and examples, see “User-supplied condition hints,” in “SQL Language Elements” in *Reference: Building Blocks, Tables, and Procedures*.

**See also**
- *Setting Query Time Limits* on page 92
- *Setting Query Priority* on page 93
- *Setting Query Optimization Options* on page 93
- *Monitoring Workloads* on page 95
Monitoring Workloads

Use the stored procedures that monitor table, column, and index usage for better query performance.

Indexes are often created to provide optimization metadata and to enforce uniqueness and primary/foreign key relationships. Once an index is created, however, DBAs face the challenge of quantifying benefits that the index provides.

Tables are often created in the IQ Main Store for the temporary storage of data that must be accessed by multiple connections or over a long period. These tables might be forgotten while they continue to use valuable disk space. Moreover, the number of tables in a data warehouse is too large and the workloads are too complex to manually analyze usage.

Thus, unused indexes and tables waste disk space, increase backup time, and degrade DML performance.

Sybase IQ offers tools for collecting and analyzing statistics for a defined workload. DBAs can quickly determine which database objects are being referenced by queries and should be kept. Unused tables/columns/indexes can be dropped to reduce wasted space, improve DML performance, and decrease backup time.

Workload monitoring is implemented using stored procedures, which control the collection and report detailed usage of table, column, and, index information. These procedures complement INDEX_ADVISOR functionality, which generates messages suggesting additional column indexes that may improve performance of one or more queries. Once recommended indexes have been added, their usage can be tracked to determine if they are worth keeping.

For details on workload monitoring procedures, see and “sp_iqcolumnuse procedure,” “sp_iqindexadvice procedure,” “sp_iqindexuse procedure,” “sp_iqtableuse procedure,” “sp_iqunusedcolumn procedure,” “sp_iqunusedindex procedure,” “sp_iqunusedtable procedure,” and “sp_iqworkmon procedure” in Reference: Building Blocks, Tables, and Procedures.

See also “INDEX_ADVISOR option” in Reference: Statements and Options.

See also

- Setting Query Time Limits on page 92
- Setting Query Priority on page 93
- Setting Query Optimization Options on page 93
- Setting User-Supplied Condition Hints on page 94
Optimizing Delete Operations

Sybase IQ chooses the best algorithm to process delete operations on columns with HG and WD indexes.

See also
- Tips for Structuring Queries on page 87
- Planning Queries on page 89
- Controlling Query Processing on page 92

HG Delete Operations

Sybase IQ chooses one of three algorithms to process delete operations on columns with an HG (High_Group) index.

- Small delete provides optimal performance when rows are deleted from very few groups. It is typically selected when the delete is only 1 row or the delete has an equality predicate on the columns with an HG index. The small delete algorithm can randomly access the HG. Worst case I/O is proportional to the number of groups visited.
- Mid delete provides optimal performance when rows are deleted from several groups, but the groups are sparse enough or few enough that not many HG pages are visited. The mid delete algorithm provides ordered access to the HG. Worst case I/O is bounded by the number of index pages. Mid delete has the added cost of sorting the records to delete.
- Large delete provides optimal performance when rows are deleted from a large number of groups. The large delete scans the HG in order until all rows are deleted. Worst case I/O is bounded by the number of index pages. Large delete is parallel, but parallelism is limited by internal structure of the index and the distribution of group to deleted from. Range predicates on HG columns can be used to reduce the scan range of the large delete.

HG Delete Costing

The delete cost model considers many factors including I/O costs, CPU costs, available resources, index metadata, parallelism, and predicates available from the query.

Specifying predicates on columns that have HG, LF, or enumerated FP indexes greatly improves costing. In order for the HG costing to pick an algorithm other than large delete, it must be able to determine the number of distinct values (groups) affected by deletions. Distinct count is initially assumed to be lesser of the number of index groups and the number of rows deleted. Predicates can provide an improved or even exact estimate of the distinct count.

Costing currently does not consider the effect of range predicates on the large delete. This can cause mid delete to be chosen in cases where large delete would be faster. You can force the large delete algorithm if needed in these cases, as described in the next section.
**Using HG Delete Performance Option**

You can use the `HG_DELETE_METHOD` option to control HG delete performance.

The value of the parameter specified with the `HG_DELETE_METHOD` option forces the use of the specified delete algorithm as follows:

- 1 = Small delete
- 2 = Large delete
- 3 = Mid delete
- `DML_OPTIONS5 = 4` (Disable Push Delete Predicates) Default 0 — Disables pushing range predicates to the HG large delete.

For more information on the `HG_DELETE_METHOD` database option, see “HG_DELETE_METHOD option” in “Database Options” in Reference: Statements and Options.

**See also**

- *WD Delete Operations* on page 97
- *TEXT Delete Operations* on page 98

**WD Delete Operations**

Sybase IQ chooses one of three algorithms to process delete operations on columns with a WD (Word) index.

- Small delete provides optimal performance when the rows deleted contain few distinct words, so that not many WD pages need to be visited. The WD small delete algorithm performs an ordered access to the WD. Worst case I/O is bounded by the number of index pages. Small delete incorporates the cost of sorting the words and record IDs in the records to delete.
- Mid delete for WD is a variation of WD small delete, and is useful under the same conditions as small delete, that is, when the rows deleted contain few distinct words. Mid delete for WD sorts only words in the records to delete. This sort is parallel, with parallelism limited by the number of words and CPU threads available. For Word index, the mid delete method is generally faster than small delete.
- Large delete provides optimal performance when the rows deleted contain a large number of distinct words, and therefore need to visit a large number of “groups” in the index. The large delete scans the WD in order, until all rows are deleted. Worst case I/O is bounded by the number of index pages. Large delete is parallel, but parallelism is limited by the internal structure of the index and the distribution of groups from which to delete.

**WD Delete Costing**

The WD delete cost model considers many factors including I/O costs, CPU costs, available resources, index metadata, and parallelism.
You can use the `WD_DELETE_METHOD` database option to control WD delete performance.

**Using WD Delete Performance Option**
The value of the parameter specified with the `WD_DELETE_METHOD` option forces the use of the specified delete algorithm as follows:

- 0 = Mid or large delete as selected by the cost model
- 1 = Small delete
- 2 = Large delete
- 3 = Mid delete

For more information on the `WD_DELETE_METHOD` database option, see “WD_DELETE_METHOD option” in “Database Options” of Reference: Statements and Options.

**See also**
- *HG Delete Operations* on page 96
- *TEXT Delete Operations* on page 98

**TEXT Delete Operations**
Sybase IQ chooses one of two algorithms to process delete operations on columns with a `TEXT` index.

- Small delete provides optimal performance when the rows deleted contain few distinct words, so that not many `TEXT` pages need to be visited. The `TEXT` small delete algorithm performs an ordered access to the `TEXT`. Worst case I/O is bounded by the number of index pages. Small delete incorporates the cost of sorting the words and record IDs in the records to delete.

- Large delete provides optimal performance when the rows deleted contain a large number of distinct words, and therefore need to visit a large number of “groups” in the index. The large delete scans the `TEXT` in order, until all rows are deleted. Worst case I/O is bounded by the number of index pages. Large delete is parallel, but parallelism is limited by the internal structure of the index and the distribution of groups from which to delete.

**TEXT Delete Costing**
The `TEXT` delete cost model considers many factors including I/O costs, CPU costs, available resources, index metadata, and parallelism.

You can use the `TEXT_DELETE_METHOD` database option to control `TEXT` delete performance.

**Using TEXT Delete Performance Option**
The value of the parameter specified with the `TEXT_DELETE_METHOD` option forces the use of the specified delete algorithm as follows:
• 0 = Mid or large delete as selected by the cost model
• 1 = Small delete
• 2 = Large delete

For more information on the TEXT_DELETE_METHOD database option, see “TEXT_DELETE_METHOD option” in “TEXT Indexes and Text Configuration Objects” of Unstructured Data Analytics in Sybase IQ.

See also
• HG Delete Operations on page 96
• WD Delete Operations on page 97
Index

-append | - truncate 76
-bufalloc 72
-cache 68
-cache_by_type 70
-ch 13
-cl 13
-contention 73
-debug 76
-file_suffix 70
-gm 12
-interval 75
-io 71
-summary 67
-threads 74

A

AGGREGATION_ALGORITHM_PREFERENCE 93

B

balancing I/O
  internal striping 19
  raw I/O 17
  strategic file locations 19
block size
  relationship to IQ page size 11
BT_PREFETCH_MAX_MISS 26
buffer cache
  block size 11
  cache size 10
  considerations 9
  data compression 11
  database access, multiuser 8
  IQ main and temporary buffers 10
  main 9
  main database 10
  managing 7
  memory use 8
  memory, applications 8
  memory, operating system 8
  memory, saving 11
  monitor 66
  monitor output options 67
  overhead 8
  page size 11
  physical memory 9
  settings, catalog 12
  size requirements 9
  temp 9
  temp orary store 10
  thread stacks 8
buffer cache monitor 66
buffer cache options
  MAIN_CACHE_MEMORY_MB 10
  TEMP_CACHE_MEMORY_MB 10
buffer caches
  determining sizes 7
  layout 78
buffer manager
  thrashing 79
buffer manager thrashing
  actions to take 79
  HASH_PINNABLE_CACHE_PERCENT 79
  HASH_THRASHING_PERCENT 79
buffers
  disabling operating system buffering 13

C

cache
  IQ main and temporary buffer size 10
  prefetching pages 26
  See Also buffer cache 66
  statistics 58
CACHE_PARTITIONS 78
caching methods
  using 88
Catalog buffer cache settings 12
Catalog Store
  file growth 29
columns
  significant number of null values 87
conditions
  user-supplied 94
connections
  connection requests 11
  limiting statements 25
Index

statistics 61
CPU
  availability 22
  monitoring 86
  monitoring (UNIX) 81
  monitoring (Windows) 80
  setting number 22
  statistics 59
cursors
  forcing non-scrolling 24
  limiting number of 25

D

data compression
  page size 11
Data Model Recommendations 39
  Foreign Keys 46
  HG Index Loads 42
  IQ Unique and Minimize_Storage 48
  Join Column 45
  Large Object Storage 51
  LONG VARCHAR and LONG VARBINARY 50
  Multi-Column Indexes 44
  Null Values 49
  Primary Keys 46
  Proper Data Type Sizing 47
  Simple Index Selection Criteria 41
  Temporary Tables 52
  Unsigned Data Types 49
  When and Where to use Indexes 40
Data types
  LONG VARCHAR and LONG VARBINARY 50
  Null Values 49
  Proper Data Type Sizing 47
  Unsigned Data Types 49
database
  profiling settings 35
database access
  multiuser 8
databases
  denormalizing for performance 53
  managing 29
  object profiles 35
  object profiling 35
  procedure profiling 34
  procedures 34
  profiling 36

E

EARLY_PREDICATE_EXECUTION 93
evaluation options
  queries 89
events
  viewing profiling data 34

F

file system buffering 15
files
  locating for best performance 19
FLATTEN_SUBQUERIES 88
FORCE_NO_SCROLL_CURSORS 24
Foreign Keys 46
fragmentation 13
FROM clause 55
functions
  viewing profiling data 34

H

HASH_PINNABLE_CACHE_PERCENT 79
HASH_THRASHING_PERCENT 79
heap
  low-fragmentation 13
HG Index Loads 42
HG indexes
  multicolumn 87
hyptthreading
  server switch 22

I

I/O
direct 13
  performance recommendations 17
IN_SUBQUERY_PREFERENCE 93
INDEX_ADVISOR 89
INDEX_PREFERENCE 93
indexes
  choosing 39
  HG 39, 87
  index advisor 39
  LF 39
  multicolumn 87
  types 39
Indexes
  HG Index Loads 42
  Multi-Column Indexes 44
  Simple Index Selection Criteria 41
  When and Where to use Indexes 40
internal striping 19
IOS_FILE_CACHE_BUFFERING 15
IQ PATH option
  choosing a raw device 17
IQ Store
  buffer cache size 10
IQ Unique and Minimize_Storage 48
IQ_USE_DIRECTIO 15
iqgovern switch
  restricting queries to improve performance 21
IQGOVERN_MAX_PRIORITY option 93
IQGOVERN_PRIORITY 93
IQMSG log
  setting maximum size 21
iqnumbercpu
  setting number of CPUs 22
iqwmem switch 13

J

JAVA_HEAP_SIZE 16
Join Column 45
join indexes
  performance impact 39
JOIN_ALGORITHM_PREFERENCE 93
JOIN_PREFERENCE 55

K

Keys
  Foreign Keys 46
  Primary Keys 46

L

Large Object Storage 51
lightweight processes 16
load balancing
  among query servers 28
logical servers 28
LONG VARCHAR and LONG VARCHAR 50
low-fragmentation heap 13

M

main database
  buffer cache size 10
MAIN_CACHE_MEMORY_MB 10
management, resources 5
  buffer cache 3, 7
MAX_CURSOR_COUNT 25
MAX_HASH_ROWS 93
MAX_QUERY_TIME option 92
MAX_STATEMENT_COUNT 25
memory
  applications 8
  balancing I/O 17
  buffer cache 7
  buffer cache size 7
  connection requests 11
database access, multiuser 8
  file system buffering 15
  fragmentation 13
  increasing 5
  IOS_FILE_CACHE_BUFFERING 15
  IQ_USE_DIRECTIO 15
  JAVA_HEAP_SIZE 16
  Java-enabled databases 16
  lightweight processes 16
  multiplex databases 6
multithreading 16
operating system 8
optimizing 5
optimizing for users 12
overhead 8
platform-specific memory options 13
process threading model 16
raw partitions 8
server 6
startup options 12
swapping 6
thread stacks 8
wired 13
memory usage statistics 57
memory use
other 8
memory, saving
page size 11
message log
Sybase IQ 21
monitor
IQ UTILITIES syntax 66
setting output file location 67
starting and stopping 66
monitor output options
-append | - truncate 76
-bufalloc 72
-cache 68
-cache_by_type 70
-contention 73
-debug 76
-file_suffix 70
-interval 75
-io 71
-summary 67
-threads 74
monitoring
transaction status 56
monitoring workloads 95
Multi-Column Indexes 44
multicolumn indexes 87
multiplex
performance monitor 56
multiplex databases
disk space 27
memory 6
multiplex resources
dynamically adjusting 28
multithreading
performance impact 16

N
network statistics 65
networks
large data transfers 30
networks 30
performance 30
performance suggestions 30
settings 30
NOEXEC 89
Null Values 49

O
optimizing queries 39, 87
Optimizing queries 87
option value
truncation 89
options
AGGREGATION_ALGORITHM_PREFERENCE 93
BT_PREFETCH_MAX_MISS 26
CACHE_PARTITIONS 78
DEFAULT_HAVING_SELECTIVITY 93
DEFAULT_LIKE_MATCH_SELECTIVITY 93
DEFAULT_LIKE_RANGE_SELECTIVITY 93
EARLY_PREDICATE_EXECUTION 93
FLATTEN_SUBQUERIES 88
HASH_PINNABLE_CACHE_PERCENT 79
HASH_THRASHING_PERCENT 79
IN_SUBQUERY_PREFERENCE 93
INDEX_ADVISOR 89
INDEX_PREFERENCE 93
IQ_USE_DIRECTIO 15
JAVA_HEAP_SIZE 16
JOIN_ALGORITHM_PREFERENCE 93
JOIN_PREFERENCE 55
MAIN_CACHE_MEMORY_MB 10
MAX_HASH_ROWS 93
MAX_STATEMENT_COUNT 25
NOEXEC 89
OS_FILE_CACHE_BUFFERING 15
OS_FILE_CACHE_BUFFERING_TEMPDB 15
ORDER BY clause 87
OS_FILE_CACHE_BUFFERING 15
OS_FILE_CACHE_BUFFERING_TEMPDB 15

output options
buffer cache monitor 67
output options, monitor 76
-bufalloc 72
-cache 68
-cache_by_type 70
-contention 73
-debug 76
-file_suffix 70
-interval 75
-jo 71
-summary 67
-threads 74

overhead
buffer cache 8

P

page size
block size 11
data compression 11
determining 11
memory, saving 11
reducing memory 11

paging
managing 5
monitoring on UNIX 81
monitoring on Windows 80

partitioned table 55
partitions
definition 17

Performance
monitoring and tuning 33

performance
balancing I/O 17
choosing correct index type 39
consideration 3
database procedure profiles 34
definition 3
designing for 3
dynamic monitor 56
monitoring 66
multi-user 26
restricting queries with iqgovern 21
subqueries 88

performance monitor
server level 56
physical memory
  buffer cache 9
planning
  queries 89
  query plans 91
PREFETCH_BUFFER_LIMIT 26
prefetched cache pages 26
prefetched rows
  controlling 27
Primary Keys 46
procedure profile
  ISQL 38
    sa_procedure_profile 37, 38
    sa_procedure_profile_summary 38
procedure profiling
  procedures 34
    summary of procedures 36
    viewing data in Interactive SQL 36
procedures, system
  sp_iqcolumnuse 95
  sp_iqindexuse 95
  sp_iqtableuse 95
  sp_iqunusedcolumn 95
  sp_iqunusedindex 95
  sp_iqunusedtable 95
  sp_iqworkmon 95
process threading model 16
processes
  growth 13
Proper Data Type Sizing 47
pushdown join 55

Q
queries 93
  caching methods 88
  condition hints 94
  controlling 93
  delete operations 96
  evaluation options 89
  HG delete operations 96
  joins 93
  limiting by row 23
  optimization, delete options 96
  optimizer simplications 93
  optimizing 39, 93
  ORDER BY, enhancing 87
  planning 89
  query plans 91
  query priority 93
query processing 92
query tree 91
  restricting concurrent 21
  subquery performance 88
  TEXT delete operations 98
  time limits 92
  WD delete operations 97
  workload monitoring 95
queries,
  Optimizing queries 87
  structuring 87
query execution
  distributed 28
query optimization options
  AGGREGATION_ALGORITHM_PREFERENCE 93
  DEFAULT_HAVING_SELECTIVITY 93
  DEFAULT_LIKE_MATCH_SELECTIVITY 93
  DEFAULT_LIKE_RANGE_SELECTIVITY 93
  EARLY_PREDICATE_EXECUTION 93
  IN_SUBQUERY_PREFERENCE 93
  INDEX_PREFERENCE 93
  JOIN_ALGORITHM_PREFERENCE 93
  MAX_HASH_ROWS 93
query plans 89
  evaluation options 89
  generating without executing 89
  graphical 91
  using 91
query plans, options
  INDEX_ADVISOR 89
  NOEXEC 89
  QUERY_DETAIL 89
  QUERY_PLAN 89
  QUERY_PLAN_AFTER_RUN 89
  QUERY_PLAN_AS_HTML 89
  QUERY_PLAN_AS_HTML_DIRECTORY 89
  QUERY_PLAN_TEXT_ACCESS 89
  QUERY_PLAN_TEXT_CACHING 89
  QUERY_TIMING 89
query processing
  controlling 92, 94
  monitoring 95
  priority 93
query server
  balancing loads 28
query tree 91

R

raw devices
effect on performance 17
raw partitions
file systems 8
memory use 8
RAWDETECT
disk striping option 19
request statistics 62
resource management
buffer cache 7
resource use
improving 27
indexing 39
load balancing 28
Loading with UNION ALL 54
multiplex disk space 27
network performance 30
resource use options 21
forcing non-scrolling cursors 24
limiting cursors 25
limiting dbspace use 22
limiting queries by row 23
limiting statements 25
prefetched rows 27
prefetching cache pages 26
restricting concurrent queries 21
setting available CPUs 22
typical usage 26
resources
multiplex 28
response time 3

S

sa_procedure_profile 37
sequential disk I/O 19
servers
monitoring performance 56
SET OPTION 16
Simple Index Selection Criteria 41
sp_iqcolumnuse 95
sp_iqindexuse 95
sp_iqtableuse 95
sp_iqunusedcolumn 95
sp_iqunusedindex 95
sp_iqunusedtable 95
sp_iqworkmon 95
start up options 12
-c 12
-ch 12
-cl 12
-gm 12
-gn 12
-iqgovern 12
-iqmt 12
statements
limiting statements 25
statistics
dynamic 56
store I/O statistics 63
stored procedures
performance monitoring 33
viewing profiling data 34
strategic file locations 19
structuring queries 87
subqueries
flattening 88
improving performance 88
subquery flattening 88
subquery performance 88
SUBQUERY_CACHING_PREFERENCE 88
SUBQUERY_FLATTENING_PERCENT 88
SUBQUERY_FLATTENING_PREFERENCE 88
swap files
effect on performance 5
swapping
disk space requirement 5
monitoring 6
sweeper threads 78
SWEEPER_THREADS_PERCENT 78
system procedures
sp_iqcolumnuse 95
sp_iqindexuse 95
sp_iqtableuse 95
sp_iqunusedcolumn 95
sp_iqunusedindex 95
sp_iqunusedtable 95
sp_iqworkmon 95
system resources
connections 12
managing 5
memory 5
performance considerations 3
resource use options 21
startup options 12
system triggers
  viewing profiling data 34

tables
  collapsing 39
  joining 39
TEMP_CACHE_MEMORY_MB 10
Temporary Store
  buffer cache size 10
Temporary Tables 52
thrashing, buffer manager
  actions to take 79
  HASH_PINNABLE_CACHE_PERCENT 79
  HASH_THRASHING_PERCENT 79
thread stacks
  memory 8
thread statistics 60
threads
  buffer caches 78
  monitoring 74
throughput 3
transaction log
  about 20
  truncating 20
transaction statistics 63
transaction status
  monitoring 56
Tuning
  performance 33

U
UNION ALL
  loading with 54
  rules 55
  view performance 56
  views 55
Unsigned Data Types 49
usage
  typical 26
USER_RESOURCE_RESERVATION 26
user-supplied conditions
  for queries 94

V
viewing procedure profiling information in
  Interactive SQL 36
virtual memory
  fragmentation 13
vmstat command
  monitoring buffer caches on UNIX 81

W
WASH_AREA_BUFFERS_PERCENT 78
WD delete operations 97
When and Where to use Indexes 40
wired memory 13
workload monitoring 95