SYBASE®

An SAP® Company

Performance and Tuning Guide

Sybase IQ 15.3

Contents

Managing System Resources

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

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

The largest performance gains can be realized by establishing a good design and by choosing the correct indexing strategy.

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

See also

- *Optimize Memory Use* on page 2
- *The Process Threading Model* on page 13
- *Balancing I/O* on page 15
- *Options for Tuning Resource Use* on page 21
- *Other Ways to Improve Resource Use* on page 27
- *Indexing Tips* on page 29
- *Managing Database Size and Structure* on page 31
- *Use UNION ALL Views for Faster Loads* on page 32
- *Network Performance* on page 34

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.

See also

- *Utilities to Monitor Swapping* on page 3
- *Server Memory* on page 3
- *Manage Buffer Caches* on page 4
- *Determine the Sizes of the Buffer Caches* on page 4
- *Set the Buffer Cache Sizes* on page 7
- *Specify the Page Size* on page 7
- *Optimize for Large Numbers of Users* on page 9
- *Platform-Specific Memory Options* on page 11

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 2
- *Server Memory* on page 3
- *Manage Buffer Caches* on page 4
- *Determine the Sizes of the Buffer Caches* on page 4
- *Set the Buffer Cache Sizes* on page 7
- *Specify the Page Size* on page 7
- *Optimize for Large Numbers of Users* on page 9
- *Platform-Specific Memory Options* on page 11

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.

Warning! Killing processes on UNIX systems may result in semaphores or shared memory being left behind instead of being cleaned up automatically. The correct way to shut down a Sybase IQ server on UNIX is the **stop_iq** utility,.

See also

- *Paging Increases Available Memory* on page 2
- *Utilities to Monitor Swapping* on page 3

- *Manage Buffer Caches* on page 4
- *Determine the Sizes of the Buffer Caches* on page 4
- *Set the Buffer Cache Sizes* on page 7
- *Specify the Page Size* on page 7
- *Optimize for Large Numbers of Users* on page 9
- *Platform-Specific Memory Options* on page 11

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 2
- *Utilities to Monitor Swapping* on page 3
- *Server Memory* on page 3
- *Determine the Sizes of the Buffer Caches* on page 4
- *Set the Buffer Cache Sizes* on page 7
- *Specify the Page Size* on page 7
- *Optimize for Large Numbers of Users* on page 9
- *Platform-Specific Memory Options* on page 11

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

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 5
- *Main and Temp Buffer Caches* on page 6

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 * number of CPUs * 20 * (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 **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, **sp_iqcheckdb** may require very little or a large amount of virtual memory. To limit the amount of memory needed, use the **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 s**p_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 5
- *Main and Temp Buffer Caches* on page 6

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 5
- *Memory Overhead* on page 5

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.

Buffer Cache Size Settings

Use these options to set buffer cache sizes.

Table 1. Settings that change buffer cache sizes

Method	When to use it	How long the setting is effective
-iqmc and **-iqtc** server switches	Recommended method. Sets cache sizes when the database and server are not running. Allows cache sizes >4GB. Especially useful for 64-bit platforms, or if cache size database options are set larger than your system can accommodate.	From the time the server is started until it is stopped Restart the server to change buffer cache sizes. The **-iqmc** and **-iqtc** server start-up options only remain in effect while the server is running, so you need to include them every time you restart the server.

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 swaps data in and out of memory in units of pages. 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.

Block Size

All I/O occurs in units of *blocks*. The size of these blocks is set when you create a Sybase IQ database; you cannot change it without recreating the database. By default, the IQ page size determines the I/O transfer block size. For example, the default IQ page size of 128KB results in a default block size of 8192 bytes. In general, Sybase IQ uses this ratio of default block size to page size, but it considers other factors also.

The default block size should result in an optimal balance of I/O transfer rate and disk space usage for most systems. It does favor saving space over performance, however. If the default block size does not work well for you, you can set it to any power of two between 4096 and 32,768, subject to the constraints that there can be no fewer than two and no more than 16 blocks in a page. You may want to set the block size explicitly in certain cases:

- For a raw disk installation that uses a disk array, larger blocks may give better performance at the expense of disk space.
- For a file system installation, to optimize performance over disk space, the IQ block size should be greater than or equal to the operating system's native block size, if there is one. You may get better I/O rates if your IQ block size matches your file system’s block size.

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, 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 2
- *Utilities to Monitor Swapping* on page 3
- *Server Memory* on page 3
- *Manage Buffer Caches* on page 4
- *Determine the Sizes of the Buffer Caches* on page 4
- *Set the Buffer Cache Sizes* on page 7
- *Optimize for Large Numbers of Users* on page 9
- *Platform-Specific Memory Options* on page 11

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.

Relative Priorities of New and Existing Connections

If Sybase IQ is very busy handling already connected users, it may be slow to respond to new connection requests. In extreme cases (such as test scripts that fire off hundreds of connections in a loop while the server is busy with inserts) new connections may time out their connection request. In this situation, the server may appear to be down when it is merely very busy. A user getting this behavior should try to connect again and should consider increasing connection time out parameters.

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

Assigns a priority to each query waiting in the **-iqgovern** queue.

-iqgovern *#_ ACTIVE_ queries_to_support*

-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.

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**numCPU* + 10. With a large number of connected users, you may find that setting this option to 2**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 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

For this many users	On these platforms	Set -c to this minimum value or higher
up to 1000	64-bit only	64MB
up to 200	64-bit	48MB (**start_iq** default for 64-bit); larger numbers of users may benefit from 64MB
up to 200	32-bit	32MB (**start_iq** default for 32-bit)

-cl and -ch

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

-cl *minimum cache ize* **-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:

```
-cl 128M
-ch 256M
```

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 (`.cfg`) 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

On all 64-bit platforms, the total amount of usable memory is limited only by the virtual memory of the system. On 32-bit platforms, some restrictions apply.

Available Memory on 32-bit Systems

On 32-bit platforms see the following table.

Table 3. Total available memory on 32-bit platforms

Platform	*Total memory available*
RedHat Linux 2.1	About 1.7GB available to Sybase IQ
RedHat Linux 3.0	About 2.7GB available to Sybase IQ
Windows 2000/2003/XP[a]	2.75GB available to Sybase IQ

[a]You need Windows 2000 Advanced Server or Datacenter Server, Windows Server 2003 Standard, Enterprise or Datacenter Edition, or Windows XP Professional to get this much memory, and you must set the /3GB switch. Without the switch, the limit is 2GB. This amount is the total memory available to the process. Total size of buffer caches must not exceed 2GB on Windows servers, even with the /3GB setting. For details, see the *Installation and Configuration Guide for Windows*.

Due to the virtual memory usage pattern within the Sybase IQ server, virtual memory fragmentation could cause excessive process growth on Windows platforms. To reduce the likelihood of this situation, Sybase IQ supports the use of Microsoft's low-fragmentation heap (LFH) on Windows XP and Windows Server 2003.

Wired Memory Pool

On HP and Sun 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 Sun.) 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 Sun 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.

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:

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

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

```
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.

This direct I/O performance option is available on Sun 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 Sun 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 IO 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.
- 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 13

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 12

The Process Threading Model

Sybase IQ uses operating system kernel threads for best performance.

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.

See also

- *Performance Considerations* on page 1
- *Optimize Memory Use* on page 2
- *Balancing I/O* on page 15
- *Options for Tuning Resource Use* on page 21
- *Other Ways to Improve Resource Use* on page 27
- *Indexing Tips* on page 29
- *Managing Database Size and Structure* on page 31
- *Use UNION ALL Views for Faster Loads* on page 32
- *Network Performance* on page 34

Balancing I/O

Balancing I/O on your system. It explains how to use disk striping and how to locate files on separate disks to gain better performance. Controlling the size of the message log file is also discussed.

Raw I/O (on UNIX Operating Systems)

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

Most UNIX file systems divide disks into fixed size partitions. 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

- *Using Disk Striping* on page 15
- *Internal Striping* on page 17
- *Using Multiple Files* on page 17
- *Strategic File Locations* on page 18
- *Working Space for Inserting, Deleting, and Synchronizing* on page 20
- *Setting Reserved Space Options* on page 21

Using Disk Striping

Disk striping is a generic method of spreading data from a single file across multiple disk drives. Striped disks achieve significant performance gains over single disks.

Disk striping combines one or more physical disks (or disk partitions) into a single logical disk. Striped disks split I/O transfers across the component physical devices, performing them in parallel.

Disk striping lets you locate blocks on different disks. The first block is located on the first drive. The second block is located on the second drive, and so on. When all the drives have been used, the process cycles back and uses additional blocks on the drives. The net effect of disk striping is the random distribution of data across multiple disk drives. Random operations against files stored on striped disks tend to keep all of the drives in the striped set equally busy, thereby maximizing the total number of disk operations per second. This is a very effective technique in a database environment.

Setting Up Disk Striping on UNIX

UNIX systems offering striped disks provide utilities for configuring physical disks into striped devices. See your UNIX or storage management system documentation for details.

Setting Up Disk Striping on Windows

On Windows, use hardware disk striping via an appropriate SCSI-2 disk controller. If your machine does not support hardware striping, but you have multiple disks available for your databases, you can use Windows striping to spread disk I/O across multiple disks. Set up Windows striping using the Disk Management.

Recommendations for Disk Striping

- Spread individual disks in a striped file system across several disk controllers. Do not saturate a disk controller with too many disks. See your hardware documentation for more information.
- Do not put disks on the same controller as slower devices, such as tape drives or CD-ROMs. This slows down the disk controller.
- Allocate 4 disks per server CPU in the stripe.
- The individual disks must be identical devices. This means they must be the same size, have the same format, and often be the same brand. If the layouts differ, the size of the smallest one is often used and other disk space is wasted. Also, the speed of the slowest disk is often used.
- Avoid using file striping disks for any other purpose. For example, do not use a file striped disk as a swap partition.
- Never use the disk containing the root file system as part of a striped device.
- Use raw partitions for maximum performance.

Note: For the best results when loading data, dump the data to a flat file located on a striped disk and then read the data into Sybase IQ with the **LOAD TABLE** command.

See also

- *Raw I/O (on UNIX Operating Systems)* on page 15
- *Internal Striping* on page 17
- *Using Multiple Files* on page 17
- *Strategic File Locations* on page 18
- *Working Space for Inserting, Deleting, and Synchronizing* on page 20
- *Setting Reserved Space Options* on page 21

Internal Striping

Disk striping lets you take advantage of multiple disk spindles at once, 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, you should 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 is

```
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 15
- *Using Disk Striping* on page 15
- *Using Multiple Files* on page 17
- *Strategic File Locations* on page 18
- *Working Space for Inserting, Deleting, and Synchronizing* on page 20
- *Setting Reserved Space Options* on page 21

Using Multiple Files

Using multiple files in a dbspace allows your Sybase IQ and temporary data to be distributed across multiple operating system files or partitions. Multiple files improve throughput and reduce average latency for the dbspace.

When to Add Files

When possible, allocate all files when you create a dbspace to ensure even data distribution.

If you add files later, use the **ALTER DBSPACE** command to add the additional files to the dbspace. Sybase IQ stripes new data across both old and new dbspaces. Striping may even out, or it may remain unbalanced, depending on the type of updates you have. The number of pages that are "turned over" due to versioning has a major impact on whether striping is rebalanced.

See also

- *Raw I/O (on UNIX Operating Systems)* on page 15
- *Using Disk Striping* on page 15
- *Internal Striping* on page 17
- *Strategic File Locations* on page 18
- *Working Space for Inserting, Deleting, and Synchronizing* on page 20
- *Setting Reserved Space Options* on page 21

Strategic File Locations

Provide additional storage resources to improve random and sequential 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 for proxy tables in other databases, such as Adaptive Server® Enterprise.
- Place the transaction log and message log on separate disks from the IQ store, catalog store, and temporary store, and from any proxy databases such Adaptive Server Enterprise.
- Place the database file, temporary dbspace, and transaction log file on the same physical machine as the database server.

Transaction Log

The transaction log file contains information that allows Sybase IQ to recover from a system failure. The transaction log is also required for auditing. The default file name extension for this file is `.log`.

To move or rename the transaction log file, use the Transaction Log utility (**dblog**). See Utility Guide > Database Administration Utilities > Transaction Log utility (dblog).

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.

Truncating the Transaction Log

Sybase IQ records in the transaction log the information necessary to recover from a system failure. Although the information logged is small for each committed transaction, the transaction log continues to grow in size. In systems with a high number of transactions that change data, over a period of time the log can grow to be very large.

When to truncate the log is really up to the DBA responsible for supporting the Sybase IQ systems, and depends on the growth profile of the log file and the operational procedures at the site.

The following table shows methods for truncating Sybase IQ transaction logs.

Table 4. Truncating transaction logs

If your database is …	Use this method …	For details, see …
Stopped	"The **–m** switch, which causes the transaction log to be truncated after each checkpoint for all databases	"Truncating the transaction log of a stopped database"
Running	The **dbbackup** command line utility with the **-xo** switch or the **-r** switch	Utility Guide > Database Administration Utilities > Backup utility (dbbackup)

See also

- *Message Log* on page 20
- *Truncating the Transaction Log Of a Stopped Database* on page 19
- *Truncating the Transaction Log Of a Stopped Database* on page 19

Truncating the Transaction Log Of a Stopped Database

Use the **–m** server start-up switch to truncate the database transaction log. Leaving the **–m** server start-up switch permanently set is *not* recommended. This switch should only be used to start Sybase IQ for a transaction log truncation. How this is done is up to the DBA, but the following procedure provides a suggestion.

1. Create a copy of the server switches `.cfg` file with a name identifying the file as the log truncation configuration setting, and edit this copy of the file to add the **–m** switch.
2. Start Sybase IQ with the configuration file containing the **–m** option. Note that no user access or transactions should be allowed at this time.
3. Shut down Sybase IQ and restart using the configuration file without the **–m** option set.

See also

- *Truncating the Transaction Log* on page 19
- *Message Log* on page 20
- *Truncating the Transaction Log* on page 19

Message Log

A message log file exists for each database. The default name of this file is `dbname.iqmsg`. The message log file is actually created when the database is started for the first time after creation of that database.

By default, 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 **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 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 1 > Overview of Sybase IQ System Administration*.

See also

- *Truncating the Transaction Log* on page 19
- *Truncating the Transaction Log Of a Stopped Database* on page 19

Working Space for Inserting, Deleting, and Synchronizing

When you insert or delete data, and when you synchronize join indexes, Sybase IQ needs some working space in the IQstore. This space is reclaimed for other purposes when the transaction that needs it commits.

Ordinarily, as long as you maintain a reasonable percentage of free space in your IQ store, you will have enough free space. However, for certain deletions, depending on the size of the data and its distribution among database pages, you may need a large amount of working space. In the case where you are deleting a major portion of your database, and the data is distributed sparsely across many pages, you could temporarily double the size of your database.

Long transactions, such as performing a row by row update on a large table in a single transaction, can consume a large amount of main dbspace. Versioning information is saved for each update until the transaction is committed.

See also

- *Raw I/O (on UNIX Operating Systems)* on page 15
- *Using Disk Striping* on page 15

- *Internal Striping* on page 17
- *Using Multiple Files* on page 17
- *Strategic File Locations* on page 18
- *Setting Reserved Space Options* on page 21

Setting Reserved Space Options

Two database options, **MAIN_RESERVED_DBSPACE_MB** and **TEMP_RESERVED_DBSPACE_MB**, control the amount of space Sybase IQ reserves for certain operations.

For more information see "IQ main store and IQ temporary store space management" in *System Administration Guide: Volume 1 > Working with Database Objects*.

See also

- *Raw I/O (on UNIX Operating Systems)* on page 15
- *Using Disk Striping* on page 15
- *Internal Striping* on page 17
- *Using Multiple Files* on page 17
- *Strategic File Locations* on page 18
- *Working Space for Inserting, Deleting, and Synchronizing* on page 20

Options for Tuning Resource Use

Tune your resources to create faster queries

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.

By specifying the **-iqgovern** switch, you can help 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 24
- *Limiting the Number of Statements* on page 25

- *Prefetching Cache Pages* on page 25
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 26

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
- 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 24
- *Limiting the Number of Statements* on page 25
- *Prefetching Cache Pages* on page 25
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 26

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 24
- *Limiting the Number of Statements* on page 25
- *Prefetching Cache Pages* on page 25
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 26

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 24
- *Limiting the Number of Statements* on page 25
- *Prefetching Cache Pages* on page 25
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 26

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 24
- *Limiting the Number of Statements* on page 25
- *Prefetching Cache Pages* on page 25
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 26

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 25
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 26

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 24
- *Prefetching Cache Pages* on page 25
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 26

Prefetching Cache Pages

Set the **PREFETCH_BUFFER_LIMIT** and **BT_PREFETCH_MAX_MISS** options to control prefetch memory behavior.

The **PREFETCH_BUFFER_LIMIT** option defines the number of cache pages available to Sybase IQ for use in prefetching (the read ahead of database pages). This option has a default value of 0. Set this option only if advised to do so by Sybase Technical Support.

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 24
- *Limiting the Number of Statements* on page 25
- *Optimizing for Typical Usage* on page 26
- *Controlling the Number of Prefetched Rows* on page 26

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 24
- *Limiting the Number of Statements* on page 25
- *Prefetching Cache Pages* on page 25
- *Controlling the Number of Prefetched Rows* on page 26

Controlling the Number of Prefetched Rows

Set the PrefetchRows and PrefetchBuffer parameters to improve performance on cursors under certain conditions.

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 24
- *Limiting the Number of Statements* on page 25
- *Prefetching Cache Pages* on page 25
- *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.

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 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

- *Load Balancing Among Query Servers* on page 28
- *Restricting Database Access* on page 28

- *Disk Caching* on page 29

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.

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*.

See also

- *Managing Disk Space in Multiplex Databases* on page 27
- *Restricting Database Access* on page 28
- *Disk Caching* on page 29

Restricting Database Access

Schedule updates when demand is low for better query performance.

For better query performance, set the database to read-only, if possible, or schedule significant updates for low usage hours. Sybase IQ allows multiple query users to read from a table while you are inserting or deleting from that table. However, performance can degrade during concurrent updates to the database.

See also

- *Managing Disk Space in Multiplex Databases* on page 27
- *Load Balancing Among Query Servers* on page 28
- *Disk Caching* on page 29

Disk Caching

Keep large amounts of memory in active use to balance the demand for real memory against the need for data from disk.

Disk cache is memory used by the operating system to store copies of disk blocks temporarily. All file system based disk reads and writes usually pass through a disk cache. From an application's standpoint, all reads and writes involving disk caches are equivalent to actual disk operations.

Operating systems use two different methods to allocate memory to disk cache: fixed and dynamic. A preset amount of memory is used in a fixed allocation; usually a 10–15 percent memory allocation is set aside. The operating system usually manages this workspace using a LRU (least recently used) algorithm. For a dynamic allocation, the operating system determines the disk cache allocation as it is running. The goal is to keep as much memory in active use as possible, balancing the demand for real memory against the need for data from disk.

See also

- *Managing Disk Space in Multiplex Databases* on page 27
- *Load Balancing Among Query Servers* on page 28
- *Restricting Database Access* on page 28

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.

Using Join Indexes

Users frequently need to see the data from more than one table at once. This data can be joined at query time, or in advance by creating a join index. Sometimes you can improve query performance by creating a join index for columns that are joined in a consistent way.

Because join indexes require substantial time and space to load, create them only for joins needed on a regular basis. Sybase IQ join indexes support one-to-many and one-to-one join relationships.

Allowing Enough Disk Space for Deletions

When you delete data rows, Sybase IQ creates a version page for each database page that contains any of the data being deleted. The versions are retained until the delete transaction commits. For this reason, you may need to add disk space when you delete data. See *"Overlapping versions and deletions"* on for details.

See also

- *Performance Considerations* on page 1
- *Optimize Memory Use* on page 2
- *The Process Threading Model* on page 13
- *Balancing I/O* on page 15
- *Options for Tuning Resource Use* on page 21
- *Other Ways to Improve Resource Use* on page 27
- *Managing Database Size and Structure* on page 31
- *Use UNION ALL Views for Faster Loads* on page 32
- *Network Performance* on page 34

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.

Data Quantity

To control the quantity of data stored in a given table, eliminate data rows you no longer need. If your database contains data that originated in a SQL Anywhere database, you may be able to eradicate unneeded data by simply replaying Anywhere deletions; command syntax is compatible. You can do the same with data from an Adaptive Server Enterprise database, because Sybase IQ provides Transact-SQL compatibility

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 default 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 default column index. These indexes can, however, use more space than needed when rows are deleted from a table.

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

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?
- Should you create join indexes to gain performance?

Use 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 **INSERT...SELECT** into a **UNION ALL** view. **UNION ALL** operators are not fully parallel in this release. Their use may limit query parallelism.

Optimizing Queries That Reference UNION ALL Views

To adjust performance for queries that reference **UNION ALL** views, set the `JOIN_PREFERENCE` option, which affects joins between **UNION ALL** views.

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

Sybase IQ includes patented optimizations for **UNION ALL** views, including:

- Split **GROUP BY** over **UNION ALL** view
- Push-down join into **UNION ALL** view

A **UNION** can be treated as a partitioned table only if it satisfies all of the following constraints:

- It contains only one or more **UNION ALL**.
- Each arm of the **UNION** has only one table in its **FROM** clause, and that table is a physical base table.
- No arm of the **UNION** has a **DISTINCT**, a **RANK**, an aggregate function, or a **GROUP BY** clause.
- Each item in the **SELECT** clause within each arm of the **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 34

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 33

Network Performance

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

Improving Large Data Transfers

To decrease overall throughput and increase average response time:

- Perform large transfers during off-hour periods, if possible.
- Limit the number of concurrent queries during large transfers.
- Do not run queries and insertions concurrently during large transfers.
- Use stored procedures to reduce total traffic.
- Use row buffering to move large batches through the network.
- If large transfers are common, consider installing better network hardware that is suitable for such transfers. For example:
  - Token ring–responds better during heavy utilization periods than ethernet hardware.
  - Fiber optic–provides very high bandwidth, but is usually too expensive to use throughout the entire network.

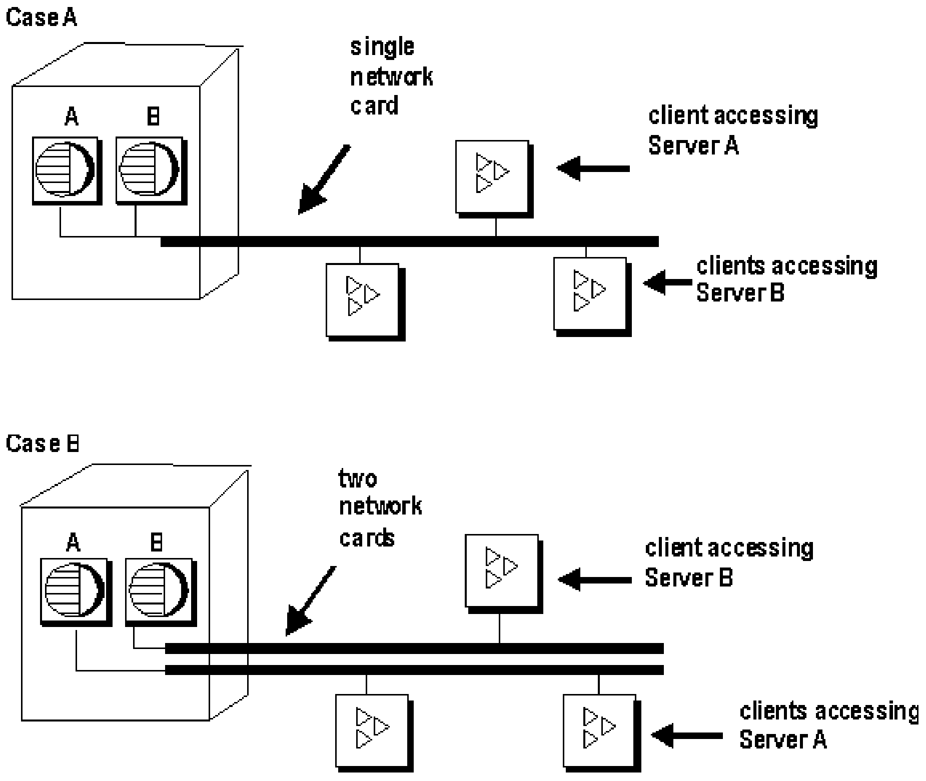
- Separate network–can be used to handle network traffic between the highest volume workstations and the server.

Isolate Heavy Network Users

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


Case A
A
B
single network card
client accessing Server A
clients accessing Server B
Case B
A
B
two network cards
client accessing Server B
clients accessing Server A

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.

See also

- *Performance Considerations* on page 1
- *Optimize Memory Use* on page 2
- *The Process Threading Model* on page 13
- *Balancing I/O* on page 15
- *Options for Tuning Resource Use* on page 21
- *Other Ways to Improve Resource Use* on page 27
- *Indexing Tips* on page 29
- *Managing Database Size and Structure* on page 31
- *Use UNION ALL Views for Faster Loads* on page 32

Monitoring and Tuning Performance

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

Viewing the Sybase IQ Environment

The first step in tuning Sybase IQ performance is to look at your environment.

Getting Information Using Stored Procedures

Several stored procedures display database information.

Table 5. Name statistics

Name	Description
sp_iqconnection	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 > System Procedures > System stored procedures > sp_iqconnection procedure*
sp_iqcontext	Tracks and displays, by connection, information about statements that are currently executing. See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqcontext procedure*
sp_iqcheckdb	Checks validity of the current database. Optionally corrects allocation problems for dbspaces or databases. See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqcheckdb procedure*
sp_iqdbstatistics	Reports results of the most recent **sp_iqcheckdb**. See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqdbstatistics procedure*
sp_iqdbsize	Displays the size of the current database. See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqdbsize procedure*

Name	Description
sp_iqspaceinfo	Displays space usage by each object in the database See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqspaceinfo procedure*
sp_iqstatus	Displays miscellaneous status information about the database. See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqstatus procedure*
sp_iqtablesize	Displays the number of blocks used by each object in the current database and the name of the dbspace in which the object is located. See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqtablesize procedure*

See *Reference: Building Blocks, Tables, and Procedures* for syntax details and examples of all Sybase IQ stored procedures.

Profiling Database Procedures

Procedure profiling tracks the execution times of procedures, triggers, and other system events. Use profiling in Sybase Cental 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 6. Database Profile Properties.

Property Name	Description
Name	Lists the name of the object.
Owner	Lists the owner of the object.
Table	Lists which table a trigger belongs to (this column only appears on the database Profile tab).
Event	Shows the type of trigger for system triggers. This can be Update or Delete.
Type	Lists the type of object, for example, a procedure.
# Exes.	lists the number times each object has been called.

Property Name	Description
#msecs.	Lists the total execution time for each object.

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 7. Object Profile Properties.

Property name	Description
Calls	Lists the number of times the object has been called.
Milliseconds	Lists the total execution time for each object.
Line	Lists the line number beside each line of the procedure.
Source	Displays the SQL procedure, line by line.

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
    - 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.

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.

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 8. sa_procedure_profile_summary statistics

Column name	Description
object_type	Identifies the object type: • P (Stored procedure) • F (Function) • T (Trigger) • E (Event) • S (System trigger)
object_name	Lists the name of the object.
executions	Lists the number times each object has been called.
owner_name	Lists the owner of the object.
table_name	Specifies which table to profile triggers.
executions	Lists the number of times the object has been called.

Column name	Description
Milliseconds	Identifies the time to execute the line, in milliseconds.
foreign_owner	Identifies the database user who owns the foreign table for a system trigger.
foreign_table	Identifies The name of the foreign table for a system trigger.

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 9. sa_procedure_profile statistics

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

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:

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

For other options, see SQL Anywhere Server - SQL Reference > System Procedures > sa_server_option system procedure.

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:

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

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

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 tree 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 43
- *Cache Statistics* on page 43
- *CPU Usage Statistics* on page 45
- *Thread Statistics* on page 46
- *Connection Statistics* on page 46
- *Request Statistics* on page 47
- *Transaction Statistics* on page 48
- *Store I/O Statistics* on page 49
- *DBspace Usage Statistics* on page 50
- *Network Statistics* on page 50

Memory Usage Statistics

Memory usage statistics show server memory statistics.

Table 10. Memory Usage

Name	Description	Monitored By Default?
Memory Allocated	Memory allocated by the IQ server in megabytes	Yes
Maximum Memory Allocated	Maximum memory allocated by the IQ server in megabytes	No

See also

- *Cache Statistics* on page 43
- *CPU Usage Statistics* on page 45
- *Thread Statistics* on page 46
- *Connection Statistics* on page 46
- *Request Statistics* on page 47
- *Transaction Statistics* on page 48
- *Store I/O Statistics* on page 49
- *DBspace Usage Statistics* on page 50
- *Network Statistics* on page 50
- *Monitoring Performance at the Server Level* on page 42

Cache Statistics

Cache statistics describe cache use.

Table 11. Cache Statistics

Name	Description	Monitored By Default?
Catalog Cache Hits	Number of catalog cache hits per second.	No
Temporary Cache Hits	Number of temporary cache hits per second.	No
Main Cache Hits	Number of main cache hits per second.	No
Catalog Cache Reads	Number of catalog cache page lookups per second.	Yes

Name	Description	Monitored By Default?
Temporary Cache Reads	Number of temporary cache page lookups per second.	No
Main Cache Reads	Number of main cache page lookups per second.	No
Catalog Cache Current Size	Current catalog cache size in megabytes.	No
Temporary Cache Current Size	Current temporary cache size in megabytes.	No
Main Cache Current Size	Current main cache size in megabytes.	No
Catalog Cache in Use Percentage	Percentage of catalog cache in use.	No
Temporary Cache in Use Percentage	Percentage of Temporary cache in use.	No
Main Cache in Use Percentage	Percentage of Main cache size in use.	No
Catalog Cache Pinned	Number of pinned catalog cache pages.	No
Temporary Cache Pinned	Number of pinned temporary cache pages.	No
Main Cache Pinned	Number of pinned main cache pages.	No
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

- *CPU Usage Statistics* on page 45

- *Thread Statistics* on page 46
- *Connection Statistics* on page 46
- *Request Statistics* on page 47
- *Transaction Statistics* on page 48
- *Store I/O Statistics* on page 49
- *DBspace Usage Statistics* on page 50
- *Network Statistics* on page 50
- *Monitoring Performance at the Server Level* on page 42

CPU Usage Statistics

CPU usage statistics show the percentage of CPU resources in use.

Table 12. CPU Usage

Name	Description	Monitored By Default?
CPU Usage	IQ process CPU usage percentage, including both system and user usage.	Yes
CPU System Usage	IQ process CPU system usage percentage.	No
CPU User Usage	IQ process CPU user usage percentage.	No

See also

- *Memory Usage Statistics* on page 43
- *Cache Statistics* on page 43
- *Thread Statistics* on page 46
- *Connection Statistics* on page 46
- *Request Statistics* on page 47
- *Transaction Statistics* on page 48
- *Store I/O Statistics* on page 49
- *DBspace Usage Statistics* on page 50
- *Network Statistics* on page 50
- *Monitoring Performance at the Server Level* on page 42

Thread Statistics

Thread statistics describe thread use.

Table 13. Thread Statistics

Name	Description	Monitored By Default?
IQ Threads in Use	Number of threads used by the IQ server	No
IQ Threads Available	Number of threads available in the IQ server	No
SA Threads in Use	Number of threads used by the SQL Anywhere engine.	No

See also

- *Memory Usage Statistics* on page 43
- *Cache Statistics* on page 43
- *CPU Usage Statistics* on page 45
- *Connection Statistics* on page 46
- *Request Statistics* on page 47
- *Transaction Statistics* on page 48
- *Store I/O Statistics* on page 49
- *DBspace Usage Statistics* on page 50
- *Network Statistics* on page 50
- *Monitoring Performance at the Server Level* on page 42

Connection Statistics

Connection statistics display connection activities.

Table 14. Connection Statistics

Name	Description	Monitored By Default?
Total Connections	Total number of connections including user and INC connections.	Yes
User Connections	Number of user connections.	No
INC Incoming Connections	Number of INC incoming connections	No

Name	Description	Monitored By Default?
INC Outgoing Connections	Number of INC outgoing connections	No
User Connections Per Minute	Number of user connections per minute	No
User Disconnections Per Minute	Number of user disconnections per minute	No

See also

- *Memory Usage Statistics* on page 43
- *Cache Statistics* on page 43
- *CPU Usage Statistics* on page 45
- *Thread Statistics* on page 46
- *Request Statistics* on page 47
- *Transaction Statistics* on page 48
- *Store I/O Statistics* on page 49
- *DBspace Usage Statistics* on page 50
- *Network Statistics* on page 50
- *Monitoring Performance at the Server Level* on page 42

Request Statistics

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

Table 15. Request Statistics

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

See also

- *Memory Usage Statistics* on page 43
- *Cache Statistics* on page 43
- *CPU Usage Statistics* on page 45
- *Thread Statistics* on page 46
- *Connection Statistics* on page 46
- *Transaction Statistics* on page 48
- *Store I/O Statistics* on page 49
- *DBspace Usage Statistics* on page 50
- *Network Statistics* on page 50
- *Monitoring Performance at the Server Level* on page 42

Transaction Statistics

Transaction statistics display transaction activity.

Table 16. Transaction Statistics

Name	Description	Monitored By Default?
Total Transaction Count	Total number of active transactions including user and INC transactions.	No
User Transaction Count	Number of active user transactions	No
INC Transaction Count	Number of active INC transactions	No
Active Load Table Statements	Number of active load table statements	No

See also

- *Memory Usage Statistics* on page 43
- *Cache Statistics* on page 43
- *CPU Usage Statistics* on page 45
- *Thread Statistics* on page 46
- *Connection Statistics* on page 46
- *Request Statistics* on page 47
- *Store I/O Statistics* on page 49
- *DBspace Usage Statistics* on page 50
- *Network Statistics* on page 50
- *Monitoring Performance at the Server Level* on page 42
- *Monitoring Performance at the Server Level* on page 42

Store I/O Statistics

Store I/O statistics describe disk reads and writes.

Table 17. Store I/O Statistics

Name	Description	Monitored By Default?
Catalog Store Disk Reads	Number of kilobytes per second that have been read from the catalog store.	No
Temporary Store Disk Reads	Number of kilobytes per second that have been read from the temporary store.	No
Main Store Disk Reads	Number of kilobytes per second that have been read from the main store.	No
Catalog Store Disk Writes	Number of kilobytes per second that have been written to the catalog store.	No
Temporary Store Disk Writes	Number of kilobytes per second that have been written to the temporary store.	No
Main Store Disk Writes	Number of kilobytes per second that have been written to the main store.	No

See also

- *Memory Usage Statistics* on page 43
- *Cache Statistics* on page 43
- *CPU Usage Statistics* on page 45
- *Thread Statistics* on page 46
- *Connection Statistics* on page 46
- *Request Statistics* on page 47
- *Transaction Statistics* on page 48
- *DBspace Usage Statistics* on page 50
- *Network Statistics* on page 50
- *Monitoring Performance at the Server Level* on page 42

DBspace Usage Statistics

DBspace usage statistics identify dbspace availability.

Table 18. DBSpace Usage

Name	Description	Monitored By Default?
DBSpace File Size in Use	DBSpace size in use. There is one such statistic per dbspace.	No
Percentage of DBSpace Size Available	Percentage of free space available for every dbspace file. There is one such statistic per dbspace per file.	No

See also

- *Memory Usage Statistics* on page 43
- *Cache Statistics* on page 43
- *CPU Usage Statistics* on page 45
- *Thread Statistics* on page 46
- *Connection Statistics* on page 46
- *Request Statistics* on page 47
- *Transaction Statistics* on page 48
- *Store I/O Statistics* on page 49
- *Network Statistics* on page 50
- *Monitoring Performance at the Server Level* on page 42

Network Statistics

Network statistics display network activity.

Table 19. Network Statistics

Name	Description	Monitored By Default?
Bytes Received	Number of bytes per second received during client/server communications.	Yes
Bytes Received Uncompressed	Number of bytes per second that would have been received during client/server communications if compression was disabled.	No

Name	Description	Monitored By Default?
Bytes Sent	Number of bytes per second sent during client/server communications.	Yes
Bytes Sent Uncompressed	Number of bytes per second that would have been sent during client/server communications if compression was disabled.	No
Free Communication Buffers	Number of available network communication buffers.	No
Total Communication Buffers	Total number of network communication buffers.	No

See also

- *Memory Usage Statistics* on page 43
- *Cache Statistics* on page 43
- *CPU Usage Statistics* on page 45
- *Thread Statistics* on page 46
- *Connection Statistics* on page 46
- *Request Statistics* on page 47
- *Transaction Statistics* on page 48
- *Store I/O Statistics* on page 49
- *DBspace Usage Statistics* on page 50
- *Monitoring Performance at the Server Level* on page 42

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.

See also

- *Viewing the Sybase IQ Environment* on page 37
- *Monitoring Performance Statistics* on page 42
- *Buffer Cache Structure* on page 64

- *Avoid Buffer Manager Thrashing* on page 65
- *System Utilities to Monitor CPU Use* on page 72
- *Buffer Cache Monitor Checklist* on page 68

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:

```
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.

See also

- *Output Options* on page 53
- *Checking Results While the Monitor Runs* on page 63
- *Stopping the Buffer Cache Monitor* on page 63

- *Examining and Saving Monitor Results* on page 64

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 20. -summary output fields

Output field	Description
Users	Number of users connected to the buffer cache
IO	Combined physical reads and writes by the buffer cache

The fields displayed are as described for the other options, plus the following:

See also

- *-cache* on page 53
- *-cache_by_type* on page 55
- *-file_suffix* on page 56
- *-io* on page 56
- *-bufalloc* on page 57
- *-contention* on page 58
- *-threads* on page 60
- *-interval* on page 61
- *-append* / *- truncate* on page 62
- *-debug* on page 62

-cache

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

Usage

```
monitor_options -cache
```

Output

Table 21. -cache output fields

Output field	Description
Finds	Find requests to the buffer cache. If the Finds value suddenly drops to zero and remains zero, the server is deadlocked. When the server has any activity, the Finds value is expected to be non-zero.
Creats	Requests to create a page within the database
Dests	Requests to destroy a page within the database
Dirty	Number of times the buffer was dirtied (modified)
HR%	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.
BWaits	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.
ReReads	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.
FMiss	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
Cloned	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.
Reads/Writes	Physical reads and writes performed by the buffer cache
PF/PFRead	Prefetch requests and reads done for prefetch.

Output field	Description
GDirty	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.
Pin%	Percentage of pages in the buffer cache in use and locked.
Dirty%	Percentage of buffer blocks that were modified. Try not to let this value exceed 85-90%; otherwise, *GDirty* will become greater than 0.

See also

- *-summary* on page 53
- *-cache_by_type* on page 55
- *-file_suffix* on page 56
- *-io* on page 56
- *-bufalloc* on page 57
- *-contention* on page 58
- *-threads* on page 60
- *-interval* on page 61
- *-append | - truncate* on page 62
- *-debug* on page 62

-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 53
- *-cache* on page 53
- *-file_suffix* on page 56
- *-io* on page 56
- *-bufalloc* on page 57
- *-contention* on page 58
- *-threads* on page 60
- *-interval* on page 61

- *-append* | *- truncate* on page 62
- *-debug* on page 62

-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 53
- *-cache* on page 53
- *-cache_by_type* on page 55
- *-io* on page 56
- *-bufalloc* on page 57
- *-contention* on page 58
- *-threads* on page 60
- *-interval* on page 61
- *-append* | *- truncate* on page 62
- *-debug* on page 62

-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 22. -io output fields

Output field	Description
Reads	Physical reads performed by the buffer cache
Lrd(KB)	Logical kilobytes read in (page size multiplied by the number of requests)
Prd(KB)	Physical kilobytes read in

Output field	Description
Rratio	Compression ratio of logical to physical data read in, a measure of the efficiency of the compression to disk for reads
Writes	Physical writes performed by the buffer cache
Lwrt(KB)	Logical kilobytes written
Pwrt(KB)	Physical kilobytes written
Wratio	Compression ratio of logical to physical data written

See also

- *-summary* on page 53
- *-cache* on page 53
- *-cache_by_type* on page 55
- *-file_suffix* on page 56
- *-bufalloc* on page 57
- *-contention* on page 58
- *-threads* on page 60
- *-interval* on page 61
- *-append | - truncate* on page 62
- *-debug* on page 62

-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 23. -bufalloc output fields

Output field	Description
OU	User_Resource_Reservation option setting (formerly Optimize_For_This_Many_Users)
AU	Current number of active users

Output field	Description
MaxBuf	Number buffers under control of the buffer allocator
Avail	Number of currently available buffers for pin quota allocation
AvPF	Number of currently available buffers for prefetch quota allocation
Slots	Number of currently registered objects using buffer cache quota
PinUser	Number of objects (for example, hash, sort, and B-tree objects) using pin quota
PFUsr	Number of objects using prefetch quota
Posted	Number of objects that are pre-planned users of quota
UnPost	Number of objects that are ad hoc quota users
Locks	Number of mutex locks taken on the buffer allocator
Waits	Number of times a thread had to wait for the lock

See also

- *-summary* on page 53
- *-cache* on page 53
- *-cache_by_type* on page 55
- *-file_suffix* on page 56
- *-io* on page 56
- *-contention* on page 58
- *-threads* on page 60
- *-interval* on page 61
- *-append | - truncate* on page 62
- *-debug* on page 62

-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 24. -contention output fields

Output field	Description
AU	Current number of active users
LRULks	Number times the LRU was locked (repeated for the temp cache)
woTO	Number times lock was granted without timeout (repeated for the temp cache)
Loops	Number times Sybase IQ retried before lock was granted (repeated for the temp cache)
TOs	Number of times Sybase IQ timed out and had to wait for the lock (repeated for the temp cache)
BWaits	Number of Busy Waits for a buffer in the cache (repeated for the temp cache)
IOLock	Number of times Sybase IQ locked the compressed I/O pool (repeated for the temp cache); can be ignored
IOWait	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
HTLock	Number of times Sybase IQ locked the block maps hash table (repeated for the temp cache)
HTWait	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
FLLock	Number of times Sybase IQ had to lock the free list (repeated for the temp cache)
FLWait	Number of times Sybase IQ had to wait for the lock on the free list (repeated for the temp cache)
MemLks	Number of times Sybase IQ took the memory manager (heap) lock
MemWts	Number of times Sybase IQ had to wait for the memory manager lock

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 53
- *-cache* on page 53
- *-cache_by_type* on page 55
- *-file_suffix* on page 56
- *-io* on page 56
- *-bufalloc* on page 57
- *-threads* on page 60
- *-interval* on page 61
- *-append | - truncate* on page 62
- *-debug* on page 62

-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 25. -thread output fields

Output field	Description
cpus	Number of CPUs Sybase IQ is using; this may be less than the number on the system
Limit	Maximum number of threads Sybase IQ can use
NTeams	Number of thread teams currently in use
MaxTms	Largest number of teams that has ever been in use
NThrds	Current number of existing threads
Resrvd	Number of threads reserved for system (connection) use
Free	Number of threads available for assignment. Monitor this value if it is very low, it indicates thread starvation
Locks	Number of locks taken on the thread manager
Waits	Number of times Sybase IQ had to wait for the lock on the thread manager

See also

- *-summary* on page 53
- *-cache* on page 53
- *-cache_by_type* on page 55
- *-file_suffix* on page 56
- *-io* on page 56
- *-bufalloc* on page 57
- *-contention* on page 58
- *-interval* on page 61
- *-append | - truncate* on page 62
- *-debug* on page 62

-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 53
- *-cache* on page 53
- *-cache_by_type* on page 55
- *-file_suffix* on page 56
- *-io* on page 56
- *-bufalloc* on page 57
- *-contention* on page 58
- *-threads* on page 60
- *-append | - truncate* on page 62
- *-debug* on page 62

-append | - truncate

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

Usage

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

See also

- *-summary* on page 53
- *-cache* on page 53
- *-cache_by_type* on page 55
- *-file_suffix* on page 56
- *-io* on page 56
- *-bufalloc* on page 57
- *-contention* on page 58
- *-threads* on page 60
- *-interval* on page 61
- *-debug* on page 62

-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 53
- *-cache* on page 53
- *-cache_by_type* on page 55
- *-file_suffix* on page 56
- *-io* on page 56

- *-bufalloc* on page 57
- *-contention* on page 58
- *-threads* on page 60
- *-interval* on page 61
- *-append | - truncate* on page 62

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
```

See also

- *Starting the Buffer Cache Monitor* on page 52
- *Output Options* on page 53
- *Stopping the Buffer Cache Monitor* on page 63
- *Examining and Saving Monitor Results* on page 64

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.

See also

- *Starting the Buffer Cache Monitor* on page 52
- *Output Options* on page 53
- *Checking Results While the Monitor Runs* on page 63
- *Examining and Saving Monitor Results* on page 64

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.

See also

- *Starting the Buffer Cache Monitor* on page 52
- *Output Options* on page 53
- *Checking Results While the Monitor Runs* on page 63
- *Stopping the Buffer Cache Monitor* on page 63

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.

Usually Sybase IQ is able to keep the Gdirty value at 0. If this value is greater than 0 for more than brief periods, you may need to adjust one of the database options that control the number of sweeper threads and the wash marker. See "SWEEPER_THREADS_PERCENT option" or "WASH_AREA_BUFFERS_PERCENT option" in *Reference: Statements and Options.*

See also

- *Viewing the Sybase IQ Environment* on page 37
- *Monitoring Performance Statistics* on page 42
- *Monitoring the Buffer Caches* on page 51
- *Avoid Buffer Manager Thrashing* on page 65
- *System Utilities to Monitor CPU Use* on page 72
- *Buffer Cache Monitor Checklist* on page 68

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 free memory 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 multicolumn 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.

See also

- *Manage Buffer Caches* on page 4

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 Systems* on page 67

Monitoring Paging on UNIX Systems

Use **vmstat** to monitor system activity such as paging.

The abbreviated command syntax for **vmstat** is:

```
vmstat interval count
```

The *interval* is the time between rows of output, and *count* is the number times a row of output is displayed. For more information about **vmstat** (including its options and field descriptions), see your operating system's documentation.

See also

- *Monitoring Paging on Windows Systems* on page 67

Buffer Cache Monitor Checklist

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

Table 26. Buffer cache monitor checklist

Statistic	Normal behavior	Behavior that needs adjusting	Recommended action
HR% (Cache hit rate)	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 > 0), the hit rate should gradually grow to above 90%.	Hit rate below 90% after prefetch is working. **Note:** Some objects do not do prefetching, so their hit rate may be low normally.	Try rebalancing the cache sizes of main versus temp by adjusting **-iqmc** and **-iqtc**. Also try increasing the number of prefetch threads by adjusting `PREFETCH_THREADS_PERCENT` option.
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.

Statistic	Normal behavior	Behavior that needs adjusting	Recommended action
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.
LRU Waits (LRUNum TimeOuts percentage in debug report)	20% or less	> 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	> 10%	Check for disk errors or I/O retries
FLWait (FLMutexWaits)	20% or lower	> 20%	Check the dbspace configuration: Is the database almost out of space? Is DISK_STRIPING ON? Does **sp_iqcheckdb** report fragmentation greater than 15%?
HTWait (BmapHTNumWaits) MemWts (MemNtimesWaited) (PFMgrCond VarWaits)	10% or lower	> 10%	Contact Sybase Technical Support.

Statistic	Normal behavior	Behavior that needs adjusting	Recommended action
CPU time (CPU Sys Seconds, CPU Total Seconds, in debug report)	CPU Sys Seconds < 20%	CPU Sys Seconds > 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.	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 <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.
InUse% (Buffers in use)	At or near 100% except during startup	Less than about 100%	The buffer cache may be too large. Try rebalancing the cache sizes of main versus temp by adjusting **-iqmc** and **-iqtc**.
Pin% (Pinned buffers)	< 90%	> 90 to 95%, indicating system is dangerously close to an Out of Buffers condition, which would cause transactions to roll back	Try rebalancing the cache sizes of main versus temp. If rebalancing buffer cache sizes is not possible, try reducing **-iqgovern** to limit the number of jobs running concurrently.

Statistic	Normal behavior	Behavior that needs adjusting	Recommended action
Free threads (ThrNumFree)	Free > Resrvd	If the number of free threads drops to the reserved count, the system may be thread starved.	Try one of the following: Increase the number of threads by setting **-iqmt**. Reduce thread-related options: `MAX_IQ_THREADS_ PER_CONNECTION`, `MAX_IQ_THREADS_ PER_TEAM`. Restrict query engine resource allocations by setting `USER_RESOURCE_ RESERVATION`. Limit the number of jobs by setting **-iqgovern**.
FlOutOfSpace (debug only)	0, indicating that the free list for this store is not full; unallocated pages are available	1, indicating that this store (main or temporary) is fully allocated	Add more dbspace to that store

See also

- *Viewing the Sybase IQ Environment* on page 37
- *Monitoring Performance Statistics* on page 42
- *Monitoring the Buffer Caches* on page 51
- *Buffer Cache Structure* on page 64
- *Avoid Buffer Manager Thrashing* on page 65
- *System Utilities to Monitor CPU Use* on page 72

System Utilities to Monitor CPU Use

OS-specific utilities are available to monitor CPU usage

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

See also

- *Viewing the Sybase IQ Environment* on page 37
- *Monitoring Performance Statistics* on page 42
- *Monitoring the Buffer Caches* on page 51
- *Buffer Cache Structure* on page 64
- *Avoid Buffer Manager Thrashing* on page 65
- *Buffer Cache Monitor Checklist* on page 68

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.
- Join indexes can often cause join queries to execute faster than ad hoc joins, at the expense of using more disk space and significantly increase load time. However, when a join query does not reference the largest table in a multi-table join index or the difference in row counts between the smaller and larger table is large, an ad hoc join usually outperforms the join index.
- You can improve performance by using an additional column to store frequently calculated results.

Note: Queries involving columns that have a significant number of `NULL` values run faster than in previous releases. The process of inserting or updating data in a table, however, may take longer (compared with previous releases) in cases where a significant number of NULL values are being inserted into the table.

Impact on Query Performance of GROUP BY Over a UNION ALL

Using a split **GROUP BY** method can reduce query processing time in some cases.

To improve load performance, very large tables are sometimes segmented into several small tables and accessed using a **UNION ALL** in a view. For certain very specific queries using such a view with a **GROUP BY**, the Sybase IQ optimizer is able to enhance performance by copying some **GROUP BY** operations into each arm of such a **UNION ALL**, performing the operations in parallel, then combining the results. This method, referred to as split **GROUP BY**, reduces the amount of data that is processed by the top level **GROUP BY**, and consequently reduces query processing time.

Only certain queries with a **GROUP BY** over a **UNION ALL** show a performance improvement. The following simple query, for example, benefits from the split **GROUP BY**:

```
CREATE VIEW vtable (v1 int, v2 char(4)) AS
SELECT a1, a2 FROM tableA
UNION ALL
SELECT b1, b2 FROM tableB;
SELECT COUNT(*), SUM(v1) FROM vtable GROUP BY v2;
```

When analyzing this query, the optimizer first performs COUNT(*) GROUP BY on `tableA` and **COUNT(*) GROUP BY** on `tableB`, then passes these results to the top level **GROUP BY**. The top level **GROUP BY** performs a **SUM** of the two **COUNT(*)** results, to produce the final query result. Note that the role of the top level **GROUP BY** changes: the aggregation used by the top level **GROUP BY** is **SUM** instead of **COUNT**.

See also

- *Enhancing ORDER BY Query Performance* on page 76
- *Improved Subquery Performance* on page 76
- *Using Caching Methods* on page 77

Examples of Split GROUP BY

See these examples to see how a split **GROUP BY** improves performance.

In this example, a large table named `tableA` is segmented into four smaller tables: `tabA1`, `tabA2`, `tabA3`, and `tabA4`. The view `unionTab` is created using the four smaller tables and **UNION ALL**:

```
CREATE VIEW unionTab (v1 int, v2 int, v3 int, v4 int) AS
SELECT a, b, c, d FROM tabA1
UNION ALL
SELECT a, b, c, d FROM tabA2
UNION ALL
SELECT a, b, c, d FROM tabA3
UNION ALL
SELECT a, b, c, d FROM tabA4;
```

The Sybase IQ optimizer splits the **GROUP BY** operation in the following queries and improves query performance:

```
SELECT v1, v2, SUM(v3), COUNT(*) FROM unionTab
GROUP BY v1, v2;

SELECT v3, SUM(v1*v2) FROM unionTab
GROUP BY v3;
```

See also

- *Restrictions on Split GROUP BY* on page 74

Restrictions on Split GROUP BY

Some restrictions apply to situations and queries that benefit from a split **GROUP BY**.

- The query may benefit from the split **GROUP BY**, if the query uses **UNION ALL**, rather than **UNION**. The following query uses **GROUP BY** with **UNION**, so it does *not* take advantage of the **GROUP BY** split:

  ```
  CREATE VIEW viewA (va1 int, va2 int, va3 int,
  va4 int) AS
  ```

```
SELECT b1, b2, b3, b4 FROM tableB
UNION
SELECT c1, c2, c3, c4 FROM tableC;

SELECT SUM(va1) FROM viewA GROUP BY va3;
```

- The query may benefit from the split **GROUP BY**, if an aggregation in the query does not contain DISTINCT. The following query uses **SUM DISTINCT**, so it does *not* take advantage of the split **GROUP BY**:

```
CREATE VIEW viewA (va1 int, va2 int, va3 int,
va4 int) AS
SELECT b1, b2, b3, b4 FROM tableB
UNION ALL
SELECT c1, c2, c3, c4 FROM tableC;

SELECT SUM(DISTINCT va1) FROM viewA GROUP BY va3;
```

- For the query to benefit from the split **GROUP BY**, you need enough memory in the temporary shared buffer cache to store the aggregation information and data used for processing the additional **GROUP BY** operators.

```
CREATE VIEW viewA (va1 int, va2 int, va3 int,
va4 int) AS
SELECT b1, b2, b3, b4 FROM tableB
UNION ALL
SELECT c1, c2, c3, c4 FROM tableC
UNION ALL
SELECT d1, d2, d3, d4 FROM tableD
UNION ALL
SELECT e1, e2, e3, e4 FROM tableE
UNION ALL
SELECT f1, f2, f3, f4 FROM tableF
UNION ALL
SELECT g1, g2, g3, g4 FROM tableG;

SELECT SUM(va1) FROM viewA GROUP BY va3;
```

  In this example, the Sybase IQ optimizer splits the **GROUP BY** and inserts six **GROUP BY** operators into the query plan. Consequently, the query requires more temporary cache to store aggregation information and data. If the system cannot allocate enough cache, the optimizer does not split the **GROUP BY**.You can use the `TEMP_CACHE_MEMORY_MB` database option to increase the size of the temporary cache, if memory is available.

- In order for the query to benefit from split GROUP BY, the `AGGREGATION_PREFERENCE` database option should be set to its default value of 0. This value allows the Sybase IQ optimizer to determine the best algorithm to apply to the **GROUP BY**. The query does *not* benefit from split **GROUP BY**, if the value of `AGGREGATION_PREFERENCE` forces the Sybase IQ optimizer to choose a sort algorithm to process the **GROUP BY**. The option `AGGREGATION_PREFERENCE` can be used to override the optimizer's choice of algorithm for processing the **GROUP BY** and should not be set to 1 or 2 in this case.

See also

- *Examples of Split GROUP BY* on page 74
- *Determine the Sizes of the Buffer Caches* on page 4

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:

```
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:

```
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:

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

See also

- *Improved Subquery Performance* on page 76
- *Using Caching Methods* on page 77
- *Impact on Query Performance of GROUP BY Over a UNION ALL* on page 73

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

- *Impact on Query Performance of GROUP BY Over a UNION ALL* on page 73
- *Using Caching Methods* on page 77
- *Enhancing ORDER BY Query Performance* on page 76

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

- *Improved Subquery Performance* on page 76
- *Enhancing ORDER BY Query Performance* on page 76
- *Impact on Query Performance of GROUP BY Over a UNION ALL* on page 73

Planning Queries

Generating a query plan can help you run more efficient queries.

If you have created the right indexes, the Sybase IQ query optimizer can usually execute queries in the most efficient way - sometimes even if you have not used the most effective syntax. Proper query design is still important, however. When you plan your queries carefully, you can have a major impact on the speed and appropriateness of results.

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*.

Note: For all database options that accept integer values, Sybase IQ truncates any decimal *option-value* setting to an integer value. For example, the value 3.8 is truncated to 3.

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_AS_HTML_DIRECTORY** – When **QUERY_PLAN_AS_HTML** is ON and a directory is specified with **QUERY_PLAN_AS_HTML_DIRECTORY**, Sybase IQ writes the HTML query plans in the specified directory.
- **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 79
- *Using Query Plans* on page 79

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 or a prejoin index set 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 77
- *Using Query Plans* on page 79

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 above. You can click on any node to navigate quickly through the plan.

Authorized users can display query plans in the Interactive SQL plan window. Users can also save and print query plans from **Interactive SQL** 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 77
- *The Query Tree* on page 79

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.

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 81
- *Setting Query Optimization Options* on page 82
- *Setting User-Supplied Condition Hints* on page 83
- *Monitoring Workloads* on page 83

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 81
- *Setting Query Optimization Options* on page 82
- *Setting User-Supplied Condition Hints* on page 83
- *Monitoring Workloads* on page 83

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.
- **EARLY_PREDICATE_EXECUTION** – Controls whether simple local predicates are executed before join optimization. Under most circumstances, it should not be changed.
- **IN_SUBQUERY_PREFERENCE** – Controls the choice of algorithms for processing IN subqueries. This option is designed primarily for internal use; do not use it unless you are an experienced database administrator.
- **INDEX_PREFERENCE** – Sets the index to use for query processing. The Sybase IQ optimizer normally chooses the best index available to process local **WHERE** clause predicates and other operations which can be done within an IQ index. This option is used to override the optimizer choice for testing purposes; under most circumstances it should not be changed.
- **JOIN_PREFERENCE** – Controls the choice of algorithms when processing joins. This option is designed primarily for internal use; do not use it unless you are an experienced database administrator.
- **JOIN_SIMPLIFICATION_THRESHOLD** – Controls the minimum number of tables being joined together before any join optimizer simplifications are applied. Normally you should not need to change this value.
- **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 81
- *Setting Query Priority* on page 81
- *Setting User-Supplied Condition Hints* on page 83
- *Monitoring Workloads* on page 83

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 81
- *Setting Query Priority* on page 81
- *Setting Query Optimization Options* on page 82
- *Monitoring Workloads* on page 83

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 81
- *Setting Query Priority* on page 81
- *Setting Query Optimization Options* on page 82
- *Setting User-Supplied Condition Hints* on page 83

Optimizing Delete Operations

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

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

Prior to Sybase IQ 12.6, the **HG** delete cost model considered only worst case I/O performance and therefore preferred large delete in most cases. The current 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** 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 86
- *TEXT Delete Operations* on page 87

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 84
- *TEXT Delete Operations* on page 87

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
- 3 = Mid 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 84
- *WD Delete Operations* on page 86

Tuning Servers on 32-bit Windows Systems

Refer to the tuning guidelines to optimize performance on Windows 32-bit systems.

General Performance Guidelines

Apply these general guidelines for loading and querying data.

Minimum Memory Requirements
The recommended minimum amount of memory (RAM) for running Sybase IQ under Windows is 512MB. For most applications 4GB is recommended for best performance. Set the /3GB and /PAE switches in the boot.ini to allow Sybase IQ to address as much memory as possible on Windows 32bit systems.

Preventing Memory Over Allocation
Excessive system page faulting results from overallocating the physical memory (RAM) of the machine. Excessive page faults severely degrade the performance of Sybase IQ. By carefully allocating Sybase IQ buffers, monitoring the virtual address space of the Sybase IQ process(es), and monitoring available physical memory, you can prevent memory overallocation. This section offers guidelines for monitoring Sybase IQ use of your machine's physical memory.

Monitoring Physical Memory
The amount of physical memory available to applications (Sybase IQ) is displayed under Physical Memory (K). If the Available value is consistently below 5000 it is possible the physical memory for the machine is overallocated. This is because at the 5000(K) mark, Windows produces page faults in order to maintain a minimum of 5MB of free memory.

To monitor physical memory, from Task Manager, select the Performance tab.

File Systems

The Windows file system supports compression at the file, directory and volume level. Check the compression options and disable Windows file system compression on all disks and volumes where you store Sybase IQ databases.

Sybase IQ provides built-in compression. Windows file system compression will be unable to reduce the database size further, but may add CPU overhead when performing reads or writes.

See also

- *Maximizing Network Applications Throughput* on page 90

Maximizing Network Applications Throughput

Enable the Network Services Server option “Maximize Throughput for Network Applications” to maximize throughput.

1. On the Control Panel, double click Network Connections.
2. Right-click Local Area Connection, choose Properties.
3. Choose File and Printer Sharing for Microsoft Network, then click Properties.
4. Under Optimization, choose Maximize data throughput for network applications.

Note: On some versions of Windows, you may need to install Microsoft Internet Information Services (IIS) to set server properties to maximize data throughput for network applications.

See also

- *File Systems* on page 89

Monitoring Performance

Use Sybase IQ Performance Monitor, Windows Task Manager, and the Windows Performance tool to monitor system performance.

Virtual Address Space

To avoid excessive system page faulting, the virtual address space should be less than the physical memory of the machine.

The virtual address space of a process is the total size of the process. The working set of a process is the amount of physical memory currently allocated to the process. To avoid excessive system page faulting, the virtual address space for the Sybase IQ process(es) should be less than the physical memory of the machine.

Due to the virtual memory usage pattern within the Sybase IQ server, virtual memory fragmentation could cause excessive process growth on Windows platforms. To reduce the likelihood of this situation, Sybase IQ supports the use of Microsoft's low-fragmentation heap (LFH) on Windows XP and Windows Server 2003.

Monitoring Virtual Address Space and Memory Usage

Change the options in Task Manager to monitor virtual address space and memory usage.

1. Right-click an empty space on the taskbar.
2. Choose Task Manager, then click the Processes tab.
3. Click View, choose Select Columns.
4. On the Select Columns dialog, choose these columns:

- Memory usage
- Memory usage delta
- Peak memory usage
- Page faults
- Page faults delta
- Virtual memory size

See also

- *Monitoring Page Faults* on page 91

Monitoring Page Faults

Use Windows Performance Monitor to track the number of hard page faults, which can indicate that the physical memory of the machine has been overallocated.

1. On the Control Panel, click Administrative Tools, then choose Performance.
2. Select the Sybase IQ process.
3. Select the counter **Page Faults/sec**.

   This counter includes both the soft and hard page faults. Hard page faults are the page faults resulting in disk I/O. Soft page faults in general are not a performance issue.
4. To determine the number of hard page faults, select the **LogicalDisk** object and the instance of where the file `pagefile.sys` is located (this should be on a separate volume from the Sybase IQ database).
5. Select the **Disk Transfers/sec** counter.

   This value when compared with the **Page Faults/sec** value will give an indication of the percentage of page faults that are hard page faults. Ideally there should be little to no I/O activity to the page file. In small memory configurations, however, paging is likely to occur.

   Sustained hard page fault rates above 20 per second indicate that the physical memory of the machine has been overallocated.

Note: For more information about using Performance Tools, click Help, then choose Help Topics in Performance.

See also

- *Monitoring Virtual Address Space and Memory Usage* on page 90

NTFS Cache

Use the NTFS cache to improve performance for inserts and queries.

NTFS can store significantly more data than the Sybase IQ buffer cache with the same amount of physical memory. This advantage can be most effectively be leveraged by reducing the size of the main cache, giving this memory to the NTFS which can use it more effectively. In IQ 15.x more use is made of temporary caches in load and query processing. As a result, using the NTFS cache is still a useful technique for the main cache, but may not provide the same advantages for the temporary cache.

The Sybase IQ buffer caches store Sybase IQ data (pages) in uncompressed form. As a result, a Sybase IQ buffer cache of 100MB can store 100MB worth of data. Conversely, the NTFS cache manages Sybase IQ data in its compressed form. Therefore, if the compression ratio were 2:1, 100MB of NTFS cache is potentially storing 200MB of Sybase IQ data. As a result, the NTFS cache is likely to sustain a higher cache hit rate which can lead to a reduction in I/O. The savings in I/O outweigh the computational overhead needed to decompress data as it moves from the NTFS cache to the Sybase IQ buffer caches.

See also

- *General Performance Guidelines* on page 89
- *Monitoring Performance* on page 90
- *Inserts and Queries* on page 92
- *Backup Operations* on page 93

Inserts and Queries

Well-tuned Sybase IQ insert operations exhibit certain characteristics. You can observe these characteristics from the Windows Task Manager and the Windows Performance tool.

I/O Operations

- Insert operations are generally CPU-bound. All CPUs within the system should be running at close to 100%, with 95% or higher of the CPU being executed in user mode. You can see this easily by clicking on the Performance tab of the Windows Task Manager with the View-Show Kernel Times option set.
- Physical memory should not be overallocated and in particular, the virtual address space for the Sybase IQ process should be less than the physical memory (RAM) for the machine.
- Hard page faults (I/O to the volume containing `pagefile.sys`) should be low and ideally close to 0 (zero).
- I/O operations to the IQ Store should be steady and within the I/O capacity of the disk subsystem.

Load Performance

Sybase IQ uses the Windows CreateFile option (for both creating and opening a file) that specifies a file is to be read for sequential access. This option is used on the files specified in the **LOAD TABLE** command. As a result, load performance is improved through read ahead and reduced NTFS Cache memory utilization.

Load performance can be further improved by setting the size of the main Sybase IQ buffer cache considerably smaller than the calculated recommended values. You can set the main Sybase IQ buffer cache as much as 50% smaller than the calculated recommended values.

Tuning for Queries
Reduce the size of the main buffer cache to improve query performance.

See also

- *Main and Temp Buffer Caches* on page 6
- *NTFS Cache* on page 91

Backup Operations

Set the **BLOCK FACTOR** close to the maximum block size to optimize backup operations.

Windows supports only fixed-length I/O devices. This means that each read or write to tape must be the same size as the one that preceded it and the one that follows. If any read/write operation exceeds the capacity of the hardware device, the operation fails. For backup and restore operations, this means that your backup (or restore) fails unless all of your writes (or reads) are the size the hardware is configured for.

The Sybase IQ defaults are designed to make your read and write operations as efficient as possible on each platform. However, if you override the default block size when you create a Sybase IQ database, you need to adjust the block factor when you back up that database.

For any backup or restore:

```
block size x block factor  I/O size
```

To adjust the block factor on a Windows system, you must know the maximum physical block size that can be handled by your tape device. This information usually is not documented by the drive manufacturer. To determine the value, typically 64KB, you need to write a small applet using WIN32 API calls. You must then use the block size of the database and the **BLOCK FACTOR** option of the **BACKUP** command to optimize backup performance. For complete syntax and usage, see the *Reference: Statements and Options*.

The closer to the maximum block size you can make each I/O operation, the better your backup performance will be. Use an integral **BLOCK FACTOR** that when multiplied by the block size yields as close to the drive's block size as possible.

Keep in mind that Sybase IQ adds some extra data to each block as it is written, for data integrity. So, if your database block size is 8192, and the maximum block size handled by the tape device is 128KB, you cannot use a block factor of 16, even though 8192 * 16 = 128KB. You have to account for the extra data added on each I/O operation by Sybase IQ and use a **BLOCK FACTOR** of 15. Note that 15 is the default block factor on Windows for the default database block size and the default IQ page size of 128KB.

See also

- *General Performance Guidelines* on page 89
- *Monitoring Performance* on page 90
- *NTFS Cache* on page 91
- *Inserts and Queries* on page 92

Index