SYBASE®
An SAP® Company

Performance and Tuning Guide

SAP Sybase IQ 16.0 SP01

DOCUMENT ID: DC00169-01-1601-01
LAST REVISED: April 2013
Copyright © 2013 by SAP AG or an SAP affiliate company. All rights reserved.
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.
Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors. National product specifications may vary.
These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.
SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries. Please see *http://www.sap.com/corporate-en/legal/copyright/index.epx#trademark* for additional trademark information and notices.

Contents

Audience

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

Performance Considerations

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

Response time 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 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.

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

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

Hardware Configuration

Hardware issues that impact SAP Sybase IQ performance.

Setting the Number of CPUs Available

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

Using the **-iqnumbercpus** switch is recommended only:

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

Additional Information

Administration: Database > Run Database Servers > Command Line Switches > Command Line Options for Performance > Memory Options > Number of CPUs Switch.

The Process Threading Model

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

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

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

A single, highly threaded process serves all SAP Sybase IQ users. The database server 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, SAP 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 SAP Sybase IQ threads. It is also possible that **-iqmt** is set too low for the number of connections.

SAP 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` command sets the number of threads available to a team of threads, enabling you to constrain the number of threads (and thereby the amount of system resources) allocated to a single operation.
- Use these options to control the amount of resources a particular operation consumes. For example, you can set this option before issuing an `INSERT`, `LOAD`, `BACKUP`, or `RESTORE` command.
- Setting this option requires the SET ANY PUBLIC OPTION system privilege.

Network Performance

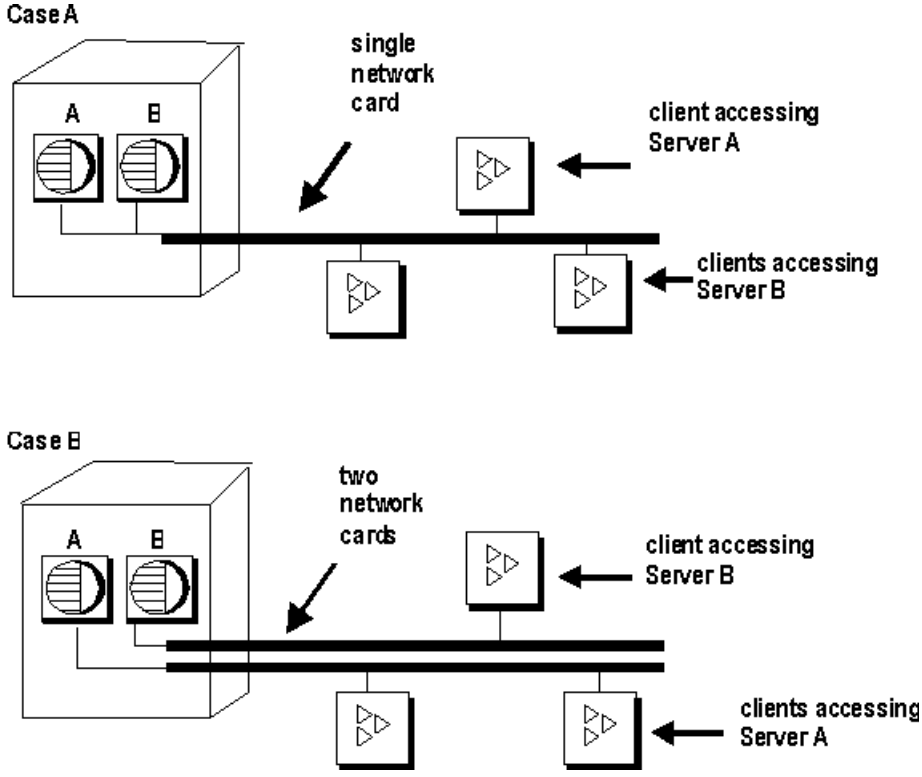
Minor changes in your environment can solve some network performance issues.

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

In case A (see the figure below) clients accessing two different database servers use one network card. That means that clients accessing Servers A and B have to compete over the network and past the network card. In the case B, clients accessing Server A use a different network card than clients accessing Server B.

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

Figure 1: Isolating heavy network users



Put Small Amounts of Data in Small Packets

If you send small amounts of data over the network, keep the default network packet size small (default is 512 bytes). The **-p** server start-up option lets you specify a maximum packet size. Your client application may also let you set the packet size.

Put Large Amounts of Data in Large Packets

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

Process at the Server Level

Filter as much data as possible at the server level.

Server Configuration

Understanding Memory

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

Server Memory

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

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.

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

See also

- *Required Memory* on page 9
- *Cache Memory* on page 11
- *Large Memory* on page 11
- *IQ Page Size* on page 12
- *Wired Memory* on page 13

Required Memory

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

Raw Partitions Versus File Systems

For UNIX-like operating 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, SAP 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 SAP 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 processing threads you use, the more memory needed. The **-iqmt** server switch controls the number of threads. 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. SAP Sybase 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 tables, their respective fields, and indexes before initiating any processing. Depending on the number of 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 tables, files, and indexes, so it uses as much virtual memory as **sp_iqcheckdb** uses when checking an entire database. It uses the temp buffer cache to keep track of blocks used.

See also

- *Server Memory* on page 9
- *Cache Memory* on page 11

- *Large Memory* on page 11
- *IQ Page Size* on page 12
- *Wired Memory* on page 13

Cache Memory

Allocate as much memory as possible to the IQ main and temporary buffer caches for optimal performance. Change the defaults to accommodate loads, queries, and applications.

Default cache size for the main and temporary buffer caches is 64MB each. Cache size is controlled with the **–iqmc** (main cache) and **–iqtc** (temporary cache) server startup options. These startup options only remain in effect while the server is running, so you need to include them every time you restart the server.

Large memory requirements are one third of the total available physical memory allocated to SAP Sybase IQ. To ensure adequate memory for the main and temporary IQ stores, set the **–iqlm**, **–iqmc**, and **–iqtc** startup parameters so that each parameter receives one third of all available physical memory.

See also

- *Server Memory* on page 9
- *Required Memory* on page 9
- *Large Memory* on page 11
- *IQ Page Size* on page 12
- *Wired Memory* on page 13

Large Memory

The **–iqlm** startup parameter specifies the maximum amount of large memory that SAP Sybase IQ can dynamically request from the operating system.

Some load operations may require more large memory than the 2GB default provides. If memory requirements exceed the default, use the **- iqlm** startup option to increase the memory that SAP Sybase IQ can dynamically request from the OS. Set **–iqlm** as a switch as part of the command or configuration file that starts the server.

Large Memory Allocation

As a general rule, large memory requirements represent one third of the total available physical memory allocated to SAP Sybase IQ. To ensure adequate memory for the main and temporary IQ stores, set the **–iqlm**, **–iqtc**, and **–iqmc** startup parameters so that each parameter receives one third of all available physical memory allocated to SAP Sybase IQ.

In most cases, you should allocate 80% of total physical memory to SAP Sybase IQ to prevent SAP Sybase IQ processes from being swapped out. Adjust actual memory allocation to accommodate other processes running on the same system. For example, on a machine with 32 cores and 128GB of total available physical memory, you would allocate 100GB

(approximately 80% of the 128GB total) to SAP Sybase IQ processes. Following the general rule, you would set the **–iqlm**, **–iqtc**, and **–iqmc** parameters to 33GB each.

See also

- *Server Memory* on page 9
- *Required Memory* on page 9
- *Cache Memory* on page 11
- *IQ Page Size* on page 12
- *Wired Memory* on page 13

IQ Page Size

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

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

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

Saving Memory

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

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

See also

- *Server Memory* on page 9
- *Required Memory* on page 9
- *Cache Memory* on page 11
- *Large Memory* on page 11
- *Wired Memory* on page 13

Wired Memory

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

Wired Memory Pool

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

Wired memory may improve SAP Sybase IQ performance when other applications are running on the same machine at the same time. Dedicating wired memory to SAP Sybase IQ, however, makes it unavailable to other applications on the machine.

To create a pool of wired memory on these UNIX platforms only, specify the **-iqwmem** command-line switch, indicating the number of MB of wired memory. (You must be user **root** to set **-iqwmem**, except on Solaris.) On 64-bit platforms, the only upper limit on **-iqwmem** is the physical memory on the machine.

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

```
-iqwmem 10000
```

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

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

Impact of Other Applications and Databases

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

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

Troubleshooting HP Memory Issues

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

See also

- *Server Memory* on page 9
- *Required Memory* on page 9
- *Cache Memory* on page 11
- *Large Memory* on page 11
- *IQ Page Size* on page 12

Tuning Options

Tuning options that provide faster query execution.

Optimizing for Typical Usage

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

SAP 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 SAP Sybase IQ thinks is currently using the product and hence allocate memory from the temporary cache more sparingly.

Set this option only if required. Contact Technical Support with details if you need to set this option.

See also

- *Optimizing for Large Numbers of Users* on page 15
- *Restricting Concurrent Queries* on page 16
- *Limiting Query Temp Space* on page 17
- *Limiting Queries by Rows Returned* on page 18
- *Forcing Cursors to be Non-Scrolling* on page 18
- *Limiting the Number of Cursors* on page 19
- *Limiting the Number of Statements* on page 20
- *Prefetching Cache Pages* on page 20
- *Controlling the Number of Prefetched Rows* on page 21
- *Controlling File System Buffering* on page 22
- *Optimizing the Cache Partitions* on page 23

Optimizing for Large Numbers of Users

Adjust the startup parameters for large numbers of users.

Table 1. Sever Startup Options

Parameter	Description
-gm	Sets the default number of connections. Usage: **-gm** *#_connections_to_support* Although this represents the total number of connections the server will support, not all connections will be active at any one time.
-iqgovern	Places a ceiling on the maximum number of queries to execute at once. If more users than the **-iqgovern** limit have submitted queries, new queries will be queued until one of the active queries is finished. Usage: **-iqgovern** *#_ ACTIVE_ queries_to_support* The optimal value for **-iqgovern** depends on the nature of your queries, number of CPUs, and size of the 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. Usage: **-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. Usage: **-c** *catalog_store_cache_size* The catalog cache size is highly dependent on schema size and the number of objects. The catalog store buffer cache is also the general memory pool for the catalog store. To specify in MB, use the form **-c** *nM*, for example, **-c** *64M*. For up to 1000 users, set **-c** to 16MB or higher. For up to 200 users, set **-c** to 48MB (default).

Parameter	Description
-cl and -ch	Set upper (**-ch**) and lower (**-cl**) limits for the catalog store cache size. *-cl minimum cache size* **-ch** *maximum cache size*. If the standard catalog cache size is too small, set **-cl** and **-ch** parameters. 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.
-iqmt	Sets the number of processing threads. If **-iqmt** is set too low for the **-gm** setting, then thread starvation can occur.

Note: 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 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

See also

- *Optimizing for Typical Usage* on page 14
- *Restricting Concurrent Queries* on page 16
- *Limiting Query Temp Space* on page 17
- *Limiting Queries by Rows Returned* on page 18
- *Forcing Cursors to be Non-Scrolling* on page 18
- *Limiting the Number of Cursors* on page 19
- *Limiting the Number of Statements* on page 20
- *Prefetching Cache Pages* on page 20
- *Controlling the Number of Prefetched Rows* on page 21
- *Controlling File System Buffering* on page 22
- *Optimizing the Cache Partitions* on page 23

Restricting Concurrent Queries

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

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

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

See also

- *Optimizing for Typical Usage* on page 14
- *Optimizing for Large Numbers of Users* on page 15
- *Limiting Query Temp Space* on page 17
- *Limiting Queries by Rows Returned* on page 18
- *Forcing Cursors to be Non-Scrolling* on page 18
- *Limiting the Number of Cursors* on page 19
- *Limiting the Number of Statements* on page 20
- *Prefetching Cache Pages* on page 20
- *Controlling the Number of Prefetched Rows* on page 21
- *Controlling File System Buffering* on page 22
- *Optimizing the Cache Partitions* on page 23

Limiting Query Temp Space

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.

SAP Sybase IQ estimates the temporary space needed to resolve the query. If the estimate exceeds the current `QUERY_TEMP_SPACE_LIMIT` setting, SAP 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

- *Optimizing for Typical Usage* on page 14
- *Optimizing for Large Numbers of Users* on page 15
- *Restricting Concurrent Queries* on page 16
- *Limiting Queries by Rows Returned* on page 18
- *Forcing Cursors to be Non-Scrolling* on page 18
- *Limiting the Number of Cursors* on page 19
- *Limiting the Number of Statements* on page 20

- *Prefetching Cache Pages* on page 20
- *Controlling the Number of Prefetched Rows* on page 21
- *Controlling File System Buffering* on page 22
- *Optimizing the Cache Partitions* on page 23

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
```

Set this option only to reject queries that consume vast resources.

See also

- *Optimizing for Typical Usage* on page 14
- *Optimizing for Large Numbers of Users* on page 15
- *Restricting Concurrent Queries* on page 16
- *Limiting Query Temp Space* on page 17
- *Limiting Queries by Rows Returned* on page 18
- *Forcing Cursors to be Non-Scrolling* on page 18
- *Limiting the Number of Cursors* on page 19
- *Limiting the Number of Statements* on page 20
- *Prefetching Cache Pages* on page 20
- *Controlling the Number of Prefetched Rows* on page 21
- *Controlling File System Buffering* on page 22
- *Optimizing the Cache Partitions* on page 23

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, SAP 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 SAP 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

- *Optimizing for Typical Usage* on page 14
- *Optimizing for Large Numbers of Users* on page 15
- *Restricting Concurrent Queries* on page 16
- *Limiting Query Temp Space* on page 17
- *Limiting Queries by Rows Returned* on page 18
- *Forcing Cursors to be Non-Scrolling* on page 18
- *Limiting the Number of Cursors* on page 19
- *Limiting the Number of Statements* on page 20
- *Prefetching Cache Pages* on page 20
- *Controlling the Number of Prefetched Rows* on page 21
- *Controlling File System Buffering* on page 22
- *Optimizing the Cache Partitions* on page 23

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

- *Optimizing for Typical Usage* on page 14
- *Optimizing for Large Numbers of Users* on page 15
- *Restricting Concurrent Queries* on page 16
- *Limiting Query Temp Space* on page 17
- *Limiting Queries by Rows Returned* on page 18
- *Forcing Cursors to be Non-Scrolling* on page 18

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

Prefetching Cache Pages

Set the BT_PREFETCH_MAX_MISS option to control prefetch memory behavior.

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

See also

Controlling the Number of Prefetched Rows

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

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

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

See also

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 role. Shut down the database and restart it for the change to take effect.

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

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

Note:

- SAP 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 SAP 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 buffer cache pages, leading to excess operating system paging activity and reduced performance. Use raw devices for databases on Solaris whenever possible.
- Windows can bias the paging algorithms to favor applications at the expense of the file system. This bias is recommended for SAP Sybase IQ performance.

See also

- *Optimizing for Typical Usage* on page 14
- *Optimizing for Large Numbers of Users* on page 15
- *Restricting Concurrent Queries* on page 16
- *Limiting Query Temp Space* on page 17
- *Limiting Queries by Rows Returned* on page 18

Optimizing the Cache Partitions

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

SAP 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.

As buffers approach the Least Recently Used (LRU) end of the cache, they pass a wash marker. SAP 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 SAP Sybase IQ processing threads, called sweeper threads, sweeps (writes) out the oldest buffers.

When SAP 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 SAP Sybase IQ had to write it out before using it.

Usually SAP 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.

Additional Information

- *Reference: Statements and Options > Database Options > Alphabetical List of Options > CACHE_PARTITIONS Option*
- *Reference: Statements and Options > Database Options > Alphabetical List of Options > SWEEPER_THREADS_PERCENT Option*
- *Reference: Statements and Options > Database Options > Alphabetical List of Options > WASH_AREA_BUFFERS_PERCENT Option*

See also

- *Optimizing for Typical Usage* on page 14
- *Optimizing for Large Numbers of Users* on page 15
- *Restricting Concurrent Queries* on page 16
- *Limiting Query Temp Space* on page 17
- *Limiting Queries by Rows Returned* on page 18

Balancing Input/Output

Use disk striping, random and sequential file disk access to balance Input/Output (I/O).

Raw Devices

On UNIX-like operating systems, you can create a database or dbspace on a raw device or a file system file.

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

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

See also

Disk Striping

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

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

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

By default, SAP Sybase IQ internally stripes pages across all files within a dbspace , so additional striping at the software or hardware level are not needed for performance. Of

course, additional striping may be necessary as part of implementing storage redundancy for the database, for example if RAID-5 is used.

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

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

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

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

See also

- *Raw Devices* on page 24
- *Internal Striping* on page 25
- *Random and Sequential File Access* on page 26
- *Transaction and Message Logs* on page 27

Internal Striping

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

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

To change the default striping when creating a dbspace:

```
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.

Changing the value of `DEFAULT_DISK_STRIPING` affects 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 Devices* on page 24
- *Disk Striping* on page 24
- *Random and Sequential File Access* on page 26
- *Transaction and Message Logs* on page 27

Random and Sequential File Access

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 SAP 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:

- 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 SAP Sybase IQ database I/O from I/O in other databases or other I/O intensive application.
- Place the database file, temporary dbspace, and transaction log file on the same physical machine as the database server.

See also

- *Raw Devices* on page 24
- *Disk Striping* on page 24
- *Internal Striping* on page 25
- *Transaction and Message Logs* on page 27

Transaction and Message Logs

Manage the size of the transaction and message logs to conserve disk space.

The transaction log file contains recovery and auditing information. Place the transaction log on a separate device or partition from the database itself to avoid database file fragmentation and to protect against media failure.

The transaction log can consume a large amount of disk space over time. Truncate the transaction log periodically to conserve disk space.

To truncate the log:

1. Shut down the server.
2. Start the server with the **–m** parameter as part of the **start_iq** command or `.cfg` file.
3. Shut down and restart the server without the **–m** parameter.

Do not leave the **–m** switch permanently set. When **–m** is set, there is no protection against media failure on the device that contains the database files. Remove the **–m** from the `.cfg` after you restart the server. To move or rename the transaction log file, use the transaction log utility (**dblog**).

Warning! The SAP 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.

Message Log

SAP Sybase IQ logs all messages in the message log file, including error, status, and insert notification messages. Limit the size of this file to conserve disk space.

At some sites the message log file tends to grow rapidly. To limit the size of this file:

- Set a maximum file size and archive the log files when the active message log is full
- Increase `NOTIFY_MODULUS` database option setting
- Use the `NOTIFY` parameter to turn off notification messages in `LOAD TABLE`. `INSERT`, and `CREATE INDEX` statements
- Use **-iqmsgsz** switch to limit the size of the message log

Additional Information

- *Utility Guide > start_iq Database Server Startup Utility > start_iq Server Options > -iqmsgsz iqsrv16 Server Option*
- *Utility Guide > start_iq Database Server Startup Utility > start_iq Server Options > -m iqsrv16 Server Option*

- *Reference: Statements and Options > Database Options > Alphabetical List of Options > NOTIFY_MODULUS Option*
- *Reference: Statements and Options > SQL Statements > CREATE INDEX Statement*
- *Reference: Statements and Options > SQL Statements > INSERT Statement*
- *Reference: Statements and Options > SQL Statements > LOAD Statement*

.

See also

- *Raw Devices* on page 24
- *Disk Striping* on page 24
- *Internal Striping* on page 25
- *Random and Sequential File Access* on page 26

Monitoring Performance

Tools you can use to determine whether your system is making optimal use of available resources.

Database Profiling Procedures

Stored procedures that return database usage statistics.

Table 2. Database Profiling Procedures

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 SAP Sybase IQ, connection status, database version status, and so on. Usage: `sp_iqconnection [ connhandle ]` 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. Usage: `sp_iqcontext [ connhandle ]` See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqcontext procedure*

Name	Description
sp_iqcheckdb	Checks validity of the current database. Optionally corrects allocation problems for dbspaces or databases. Usage: `sp_iqcheckdb 'mode target [ … ] [ resources resource-percent ]'` See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqcheckdb procedure*
sp_iqdbstatistics	Reports results of the most recent **sp_iqcheckdb**. Usage: `sp_iqdbstatistics` See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqdbstatistics procedure*
sp_iqdbsize	Displays the size of the current database. Usage: `sp_iqdbsize([ main ] )` See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqdbsize procedure*
sp_iqspaceinfo	Displays space usage by each object in the database. Usage: `sp_iqspaceinfo [‘main	[table table-name	index index-name] [...] ‘]` See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqspaceinfo procedure*
sp_iqstatus	Displays miscellaneous status information about the database. Usage: `sp_iqstatus` 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. Usage: `sp_iqtablesize ( table_owner.table_name )` See *Building Blocks, Tables, and Procedures > System Procedures > System stored procedures > sp_iqtablesize procedure*

See also

- *Event Profiling Procedures* on page 30
- *Key Performance Indicators* on page 30
- *Buffer Cache Performance* on page 32

Event Profiling Procedures

Event profiling procedures return performance statistics for stored procedures, functions, events, and triggers.

Table 3. Event Profiling Procedures

Name	Description
sa_server_option	Sets database profiling options in Interactive SQL. Usage: `CALL sa_server_option ( 'procedureprofiling', 'ON')`

sa_procedure_profile	Returns execution times for each line in database procedures, functions, events, or triggers. Usage:
	`sa_procedure_profile`
	`[ filename`
	`[, save_to_file ] ] )`

sa_procedure_profile_summary	Summarizes execution times for all procedures, functions, events, or triggers. Usage:
	`sa_procedure_profile_summary`
	`[ filename`
	`[, save_to_file ] ] )`

Additional Information

- *SQL Anywhere Server - SQL Reference > System procedures > Alphabetical list of system procedures > sa_server_option system procedure*.
- *SQL Anywhere Server - SQL Reference > System procedures > Alphabetical list of system procedures > sa_procedure_profile system procedure*.
- *SQL Anywhere Server - SQL Reference > System procedures > Alphabetical list of system procedures > sa_procedure_profile_summary system procedure*.

See also

- *Database Profiling Procedures* on page 28
- *Key Performance Indicators* on page 30
- *Buffer Cache Performance* on page 32

Key Performance Indicators

Set up a Statistics Collection in Sybase Control Center to monitor key performance indicators (KPI) on the server. KPI values are grouped into collections and appear in SCC monitors.

Key performance areas include SAP Sybase IQ servers, multiplex servers, and logical servers

SAP Sybase IQ Servers Statisitics
Various server status and usage statistics.

Table 4. SAP Sybase IQ Server Statisitics

Key Performance Area	Usage Statistics
SAP Sybase IQ Availability Statistics	Resource state, CPU usage, memory allocation, cache use, and active connections.
Overview Statistics	Server status, CPU usage, memory allocation, and current active connections.
Connection Statistics	Active, available, resumed, rolled back, and suspended user and internode connections. Also displays average number of connections and disconnections per minute.
DBSpace and DBSpace File Statistics	Dbspace and dbspace file size and available percentage.
Store Input and Output Statistics	Store Input and Output Statistics identify store reads and writes per second.
Cache Statistics	Main, catalog, and temporary cache statistics.
Operations and Request Statistics	Average, minimum, maximum, and total waiting, active, and total operations.
Network Statistics	Network statistics display network activity.
Transaction Statistics	Transaction details currently on the server.

Multiplex and Node-Related Statisitics
Multiplex and node-related statistics for multiplex servers.

Table 5. Multiplex and Node-Related Statisitics

Key Performance Area	
Multiplex Availability	Availability statistics for each multiplex node.
Multiplex Status	Status of the multiplex.
Multiplex Node Properties	Role, status, and failover state of each multiplex node.
Multiplex Link Availability	Internode communication status between a secondary node and the coordinator.

Logical Server Statistics
Logical server and node-related statistics for logical servers.

Table 6. Logical Server Statistics

Key Performance Area	Usage Statistics
Logical Server Availability	Logical server status.
Logical Server Engine Statistics	CPU usage, connection, and connection statistics.
Logical Server Connection	Average, minimum, maximum, and total number of available connections.
Logical Server Transaction	Average, minimum, maximum, and total transactions. Also displays average and minimum number of load transactions.
Logical Server Cache Statistics	Average, minimum, and maximum cache use statistics for the catalog, temporary, and main cache.
Logical Server Operations and Requests Statistics	Average, minimum, maximum, and total waiting and active operations.

Additional Information

For additional information, see the Sybase Control Center for SAP Sybase IQ online help in SCC or at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.

See also

- *Database Profiling Procedures* on page 28
- *Event Profiling Procedures* on page 30
- *Buffer Cache Performance* on page 32

Buffer Cache Performance

Buffer cache performance is a key factor in overall performance. The IQ UTILITIES Statement starts a cache monitor that collects buffer caches statistics. Use output from the cache monitor to fine-tune main and temp buffer cache memory allocation.

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

Table 7. 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%?

Statistic	Normal behavior	Behavior that needs adjusting	Recommended action
HTWait (BmapHTNumWaits) MemWts (MemNtimesWaited) (PFMgrCondVarWaits)	10% or lower	> 10%	Contact Sybase Technical Support.
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.

Statistic	Normal behavior	Behavior that needs adjusting	Recommended action
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.
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**.

Statistic	Normal behavior	Behavior that needs adjusting	Recommended action
FlOutOf-Space (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

Note: 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.

Additional Information

Reference: Statements and Options > SQL Statements > IQ UTILITIES Statement

See also

- *Database Profiling Procedures* on page 28
- *Event Profiling Procedures* on page 30
- *Key Performance Indicators* on page 30

Multiplex Performance

Adjust your system for maximum performance or better use of disk space.

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.

Managing Multiplex Disk Space

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

SAP 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. SAP 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.

Managing Logical Server Resources

Logical servers let 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.

Balancing Query Loads

Using the SAP Sybase 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.

Note: Third-party software is required. **VerifyServerName** simply allows this method to work.

Addtional Information

Administration: Database > Appendix: Connection and Communication Parameters Reference > Network Communications Parameters > VerifyServerName Communication Parameter [Verify]

Schema Design

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

Indexing

Indexing selection and solutions for SAP Sybase IQ.

See also

- *Join Column* on page 46
- *Primary Keys* on page 47
- *Foreign Keys* on page 47
- *Proper Data Type Sizing* on page 48
- *Null Values* on page 49
- *Unsigned Data Types* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52
- *Denormalizing for Performance* on page 53
- *UNION ALL Views for Faster Loads* on page 54
- *Hash Partitioning* on page 56

Indexing Tips

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

SAP 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

`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.

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 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.

Additional Information

Reference: Statements and Options > Database Options > Alphabetical List of Options > INDEX_ADVISOR Option

See also

- *When and Where to use Indexes* on page 42
- *Simple Index Selection Criteria* on page 43
- *HG Index Loads* on page 44
- *Multi-Column Indexes* on page 45

When and Where to use Indexes

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

Always use indexes on:

- Join columns (`HG` index regardless of cardinality)
- Searchable columns (`HG` or `LF` index based on cardinality)
- `DATE`, `TIME`, and `DATETIME`/`TIMESTAMP` columns (`DATE`, `TIME`, `DTTM`)

  The `DATE`, `TIME`, or `DATETIME`/`TIMESTAMP` column should also have an `LF` or `HG` index depending on data cardinality.
- If you are uncertain whether the column will be used heavily, place an `LF` or `HG` index on the column. Workload Management can subsequently be enabled to monitor the use of indexes.

- Use PRIMARY KEY, UNIQUE CONSTRAINT, or UNIQUE HG indexes where appropriate, as they provide SAP Sybase IQ with additional information about the unique data in the indexed column(s).
- A column with an HNG or CMP index should have a corresponding LF or HG index
- Indexes are not needed on columns whose data is ONLY returned to the client (projected)

See also

- *Indexing Tips* on page 41
- *Simple Index Selection Criteria* on page 43
- *HG Index Loads* on page 44
- *Multi-Column Indexes* on page 45

Simple Index Selection Criteria

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

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

- Is the cardinality greater than 1500-2000?
  If the answer is yes, place an HG index on this column. If not, place an LF index on the column.
- Does the column contain DATE, TIME, DATETIME, or TIMESTAMP data?
  If the answer is yes, place a DATE, TIME, or DTTM index on this column. You should also place an LF or HG on the column.
- Will the column be used in range searches or aggregations?
  If the answer is yes, place an HNG index on the column. You should also place an LF or HG should be on the column. If the aggregation contains more than just the column, an HNG may not be appropriate. In most cases an HNG index is not needed as the LF or HG indexes have more than enough capability to perform the aggregations. This does not apply to DATE, TIME, or DATETIME types.
- Will this column be used for word searching?
  If the answer is yes, place a WD index on the column. An LF or HG index is not necessary and would consume significant space.
- Will this column be used for full text searching?
  If the answer is yes, place a TEXT index on the column. An LF or HG is not necessary and would consume significant space.
- Will two columns in the same table be compared to each other (A = B, A < B, A > B, A <= B, A > = B)?
  If the answer is yes, place a CMP index on the two columns.
- Will this column, or set of columns, be used in GROUP BY or ORDER BY statements?

If the answer is yes, place an HG index on the column, or columns in the GROUP BY or ORDER BY statement. Each column should also have a corresponding HG or LF index.

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

See also

- *Indexing Tips* on page 41
- *When and Where to use Indexes* on page 42
- *HG Index Loads* on page 44
- *Multi-Column Indexes* on page 45

HG Index Loads

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

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

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

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

but have more of an impact to data loads. In these situations, it is important to understand why the load or delete took longer and what can be done about it.

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

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

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

See also

- *Indexing Tips* on page 41
- *When and Where to use Indexes* on page 42
- *Simple Index Selection Criteria* on page 43
- *Multi-Column Indexes* on page 45

Multi-Column Indexes

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

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

Some items to keep in mind about the `HG` indexes:

- `HG` inserts are the most expensive

- Try to guarantee that inserts will happen at the end of the index

Place generally incrementing data, like a transaction date or batch number (sequential data), at the beginning of the index list. Something that will try to guarantee a sequential key.

See also

- *Indexing Tips* on page 41
- *When and Where to use Indexes* on page 42
- *Simple Index Selection Criteria* on page 43
- *HG Index Loads* on page 44

Join Column

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

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

See also

- *Indexing* on page 41
- *Primary Keys* on page 47
- *Foreign Keys* on page 47
- *Proper Data Type Sizing* on page 48
- *Null Values* on page 49
- *Unsigned Data Types* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52
- *Denormalizing for Performance* on page 53
- *UNION ALL Views for Faster Loads* on page 54
- *Hash Partitioning* on page 56

Primary Keys

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

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

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Foreign Keys* on page 47
- *Proper Data Type Sizing* on page 48
- *Null Values* on page 49
- *Unsigned Data Types* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52
- *Denormalizing for Performance* on page 53
- *UNION ALL Views for Faster Loads* on page 54
- *Hash Partitioning* on page 56

Foreign Keys

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

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Primary Keys* on page 47

Proper Data Type Sizing

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

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

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

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

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Primary Keys* on page 47
- *Foreign Keys* on page 47
- *Null Values* on page 49
- *Unsigned Data Types* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52

- *Denormalizing for Performance* on page 53
- *UNION ALL Views for Faster Loads* on page 54
- *Hash Partitioning* on page 56

Null Values

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

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

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

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Primary Keys* on page 47
- *Foreign Keys* on page 47
- *Proper Data Type Sizing* on page 48
- *Unsigned Data Types* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52
- *Denormalizing for Performance* on page 53
- *UNION ALL Views for Faster Loads* on page 54
- *Hash Partitioning* on page 56

Unsigned Data Types

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

Use unsigned data types when the sign of the data does not matter as all data will always be greater than or equal to zero. The lack of sign storage results in column comparisons that no

longer have to perform sign comparison. This increases performance and eliminates a step in the joining and searching of data, particularly for key columns.

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Primary Keys* on page 47
- *Foreign Keys* on page 47
- *Proper Data Type Sizing* on page 48
- *Null Values* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52
- *Denormalizing for Performance* on page 53
- *UNION ALL Views for Faster Loads* on page 54
- *Hash Partitioning* on page 56

LONG VARCHAR and LONG VARBINARY

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

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

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

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Primary Keys* on page 47
- *Foreign Keys* on page 47

- *Proper Data Type Sizing* on page 48
- *Null Values* on page 49
- *Unsigned Data Types* on page 49
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52
- *Denormalizing for Performance* on page 53
- *UNION ALL Views for Faster Loads* on page 54
- *Hash Partitioning* on page 56

Large Object Storage

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

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

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Primary Keys* on page 47
- *Foreign Keys* on page 47
- *Proper Data Type Sizing* on page 48
- *Null Values* on page 49
- *Unsigned Data Types* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Temporary Tables* on page 52
- *Denormalizing for Performance* on page 53

- *UNION ALL Views for Faster Loads* on page 54
- *Hash Partitioning* on page 56

Temporary Tables

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

There are three types of Temporary Tables:

- # tables

  ```
  CREATE TABLE #temp_table( col1 int )
  ```
- Local Temporary Tables

  ```
  DECLARE LOCAL TEMPORARY TABLE temp_table ( col1 int )
  ```

  Local Temporary Tables behave just like # tables
- Global Temporary Tables

  ```
  CREATE GLOBAL TEMPORARY TABLE temp_table ( col1 int )
  ```

  Global Temporary Table structure is static across connections and reboots

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

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Primary Keys* on page 47
- *Foreign Keys* on page 47
- *Proper Data Type Sizing* on page 48
- *Null Values* on page 49
- *Unsigned Data Types* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Denormalizing for Performance* on page 53
- *UNION ALL Views for Faster Loads* on page 54
- *Hash Partitioning* on page 56

Denormalizing for Performance

Denormalizing your database can improve performance, but there are risks and disadvantages.

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

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

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

Deciding to Denormalize

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

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

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Primary Keys* on page 47
- *Foreign Keys* on page 47
- *Proper Data Type Sizing* on page 48
- *Null Values* on page 49
- *Unsigned Data Types* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52
- *UNION ALL Views for Faster Loads* on page 54
- *Hash Partitioning* on page 56

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.

SAP 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.

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.

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.

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Primary Keys* on page 47
- *Foreign Keys* on page 47
- *Proper Data Type Sizing* on page 48
- *Null Values* on page 49
- *Unsigned Data Types* on page 49

- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52
- *Denormalizing for Performance* on page 53
- *Hash Partitioning* on page 56

Queries Referencing 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.

SAP Sybase IQ includes 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

- *UNION ALL View Performance* on page 55

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

- *Queries Referencing UNION ALL Views* on page 55

Hash Partitioning

Hash table partitioning distributes data to logical partitions for parallel execution, which can enhance join performance on large tables and distributed queries (PlexQ).

New join algorithms and aggregation algorithms can take advantage of hash partitioning by reducing the amount of intermediate storage and network transfer required as well as providing increased parallelism by leveraging the semantic division of the table rows into partitions. The improved cache behavior provided by data affinity and affinity based work allocation provide further scalability in the PlexQ environment.

Using Hash Partitioned Join or Hash Partitioned Aggregation Algorithms

To use the hash partitioned join or hash partitioned aggregation algorithms, it is critical that all the columns of the hash partitioning key for the table or tables, be used in the equi-join conditions or grouping expressions for those tables. Additional join conditions or grouping expressions may be used, but if a column that is a component of the hash partitioning key is not used in the query, the partitioned algorithms will not be eligible. In such a case, other non-partitioned algorithms remain eligible and would be chosen as for non-partitioned tables. Hash partitioning of a set of tables should cover the smallest common set of columns used by the application queries on those tables. Frequently a single column is a sufficient partitioning basis. It is essential that the data types of columns in joins between partitioned tables be identical.

Small tables benefit less from hash partitioning than large tables. Consider hash partitioning for tables of at least 1 billion rows.

Large Memory

Some load operations may require more large memory than the 2GB default provides. If the memory requirements exceed the default, use the **-iqlm** startup option to increase the memory that SAP Sybase IQ can dynamically request from the OS.

As a general rule, large memory requirements represent one third of the total available physical memory allocated to SAP Sybase IQ. To ensure adequate memory for the main and temporary IQ stores, set the **–iqlm**, **–iqtc**, and **–iqmc** startup parameters so that each parameter receives one third of all available physical memory allocated to SAP Sybase IQ.

In most cases, you should allocate 80% of total physical memory to SAP Sybase IQ to prevent SAP Sybase IQ processes from being swapped out. Adjust actual memory allocation to accommodate other processes running on the same system. For example, on a machine with 32 cores and 128GB of total available physical memory, you would allocate 100GB (approximately 80% of the 128GB total) to SAP Sybase IQ processes. Following the general rule, you would set the **–iqlm**, **–iqtc**, and **–iqmc** parameters to 33GB each.

Additional Information

- *Reference: Statements and Options > Database Options > Alphabetical List of Options > JOIN_PREFERENCE Option*
- *Reference: Statements and Options > Database Options > Alphabetical List of Options > AGGREGATION_PREFERENCE Option*
- *Utility Guide > start_iq Database Server Startup Utility > start_iq Server Options > –iqlm iqsrv16 Server Option*
- *Utility Guide > start_iq Database Server Startup Utility > start_iq Server Options > –iqmc iqsrv16 Server Option*
- *Utility Guide > start_iq Database Server Startup Utility > start_iq Server Options > –iqtc iqsrv16 Server Option*

See also

- *Indexing* on page 41
- *Join Column* on page 46
- *Primary Keys* on page 47
- *Foreign Keys* on page 47
- *Proper Data Type Sizing* on page 48
- *Null Values* on page 49
- *Unsigned Data Types* on page 49
- *LONG VARCHAR and LONG VARBINARY* on page 50
- *Large Object Storage* on page 51
- *Temporary Tables* on page 52
- *Denormalizing for Performance* on page 53
- *UNION ALL Views for Faster Loads* on page 54

Troubleshooting

Identify and solve common performance issues.

Isolating Performance Problems

Determine whether the problem is external or internal to SAP Sybase IQ.

External to SAP Sybase IQ

- Monitor the OS, hardware, and storage for any bottlenecks or issues
- Look for high CPU use, high CPU system time, low CPU user time, high wait time
- Look for I/O service times that are more than 10ms

Internal to SAP Sybase IQ

- Enable `INDEX_ADVISOR` and look for missing indexes
- Enable `QUERY_PLAN_AS_HTML` and the `INDEX_ADVISOR` if the issue is with a single query

See also

- *Diagnostic Tools* on page 59
- *Common Performance Issues* on page 60

Diagnostic Tools

Utilities on your operating system that monitor system activities.

Use these utilities to get statistics on the number of running processes, including 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.

OS	Utility	Description
UNIX	**top**, **topas**	Provides an ongoing look at processor activity in real time. **top** is available on Solaris, Linux, and HP-UX platforms. **topas** is available on AIX.

OS	Utility	Description
	ps	Reports process status.
	vmstat	Displays information about system processes, memory, paging, block IQ, traps, and CPU activity.
	iostat -x	Displays disk subsystem information.
	sar	Writes selected OS activity results to standard output.
Windows	**Task Manager**, **Resource Monitor**	Provide detailed information about computer performance and running applications, processes, CPU usage, and other system services.

Note: 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

- *Isolating Performance Problems* on page 59
- *Common Performance Issues* on page 60

Common Performance Issues

Solutions to common performance problems.

See also

- *Isolating Performance Problems* on page 59
- *Diagnostic Tools* on page 59

Paging and Disk Swapping

Good memory management avoids page swapping. To minimize use of operating system files, increase or reallocate physical memory.

Insufficient memory severely degrades performance. If this is the case, you need to find a way to make more memory available. The more memory you can allocate to SAP Sybase IQ, the better.

Because there is always a fixed limit to the amount of memory, the operating system may sometimes keep part of the data in memory and the rest on disk. Paging or swapping occurs

when the operating system must go out to disk and retrieve any data before a memory request can be satisfied. Good memory management avoids or minimizes 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. 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, SAP Sybase IQ uses buffer caches for all reads and writes to your databases.

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

See also

- *Index and Row Fragmentation* on page 61
- *Catalog File Growth* on page 62
- *Thrashing and Query Execution* on page 62

Index and Row Fragmentation

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

Run these stored procedures for information about fragmentation issues:

- **sp_iqrowdensity** reports row fragmentation at the `FP` index level.
- **sp_iqindexfragmentation** reports internal fragmentation within supplemental indexes. .

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

Additional Information

- *Reference: Building Blocks, Tables, and Procedures > System Procedures > sp_iqrowdensity procedure*.
- *Reference: Building Blocks, Tables, and Procedures > System Procedures > sp_iqindexfragmentation procedure*.

See also

- *Paging and Disk Swapping* on page 60
- *Catalog File Growth* on page 62
- *Thrashing and Query Execution* on page 62

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

See also

- *Paging and Disk Swapping* on page 60
- *Index and Row Fragmentation* on page 61
- *Thrashing and Query Execution* on page 62

Thrashing and Query Execution

Adjust the `HASH_THRASHING_PERCENT` option to limit thrashing during queries that involve 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 SAP Sybase IQ may now be rolled back when the default `HASH_THRASHING_PERCENT` limit is reached. SAP 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. 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 SAP Sybase IQ is not estimating one or more hash sizes for this statement correctly. 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`.

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'`. Examine the estimates in the query plan.

See also

- *Paging and Disk Swapping* on page 60
- *Index and Row Fragmentation* on page 61
- *Catalog File Growth* on page 62

Queries and Deletions

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

Structuring Queries

Improving query structures can make your queries run faster.

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

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 66
- *Using Caching Methods* on page 66

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. SAP 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.

See also

- *Enhancing ORDER BY Query Performance* on page 65
- *Using Caching Methods* on page 66

Using Caching Methods

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

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

See also

- *Enhancing ORDER BY Query Performance* on page 65
- *Improved Subquery Performance* on page 66

Generating Query Plans

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

Before it executes any query, the query optimizer creates a query execution plan. A query execution plan represents the set of steps the database server uses to access information in the database related to a statement. The execution plan for a statement can be saved and reviewed, regardless of whether it was just optimized, whether it bypassed the optimizer, or whether its plan was cached from previous executions. Although a query execution plan may not correspond exactly to the syntax used in the original statement, operations described in the execution plan are semantically equivalent to the original query.

Query plans generate an execution tree that consists of a series of nodes that represent a processing stage. The lowest nodes on the tree are leaf nodes. Each leaf node represents a table in the query. At the top of the plan is the root of the operator tree. Information flows up from the tables and through any operators representing joins, sorts, filters, stores, aggregation, and subqueries.

Load Execution Plans

Load execution plans detail the steps that the database engine uses to insert data into a table. Load plans use the same database and output options as query execution plans. The Data Flow Object (DFO) tree identifies the number of rows processed at each stage of the load. Different SQL statements may generate different DFO trees and the same statement may generate different trees for different kind of tables (un-partitioned, range partitioned, hash partitioned, hash-range partitioned, etc.).

Generating Query Plans

To generate a query plan, set the appropriate evaluation options, then execute the query. Text versions of the plan are written to the `.iqmsg` file. HTML versions can be displayed in the **Interactive SQL Plan Viewer** or in most Web browsers.

Note: Use query plans only to evaluate the efficiency of a particular query or load. Running SAP Sybase IQ with the QUERY_PLAN option set to ON can significantly impact performance, particularly as the volume of **INSERT...VALUE** statements increase.

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.
- 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.
- QUERY_PLAN_MIN_TIME – Specifies a threshold for query execution. The query plan is generated only if query execution time exceeds the threshold in microseconds. This can improve system performance by turning off query plan generation for queries with very short execution times.

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.

See also

- *Using Query Plans* on page 69

Using Query Plans

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

HTML versions can be displayed in the Interactive SQL **Plan Viewer**. HTML query plan, each node in the tree is hyper-linked to processing details. You can click on any node to navigate quickly through the plan.

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 graphical query plans in the Interactive SQL **Plan Viewer**. Text plans are not supported in the **Plan Viewer** 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.

Additional Information

- *Reference: Building Blocks, Tables, and Procedures > SQL Functions > GRAPHICAL_PLAN function [String]*
- *Reference: Building Blocks, Tables, and Procedures > SQL Functions > HTML_PLAN function [String]*
- *Reference: Statements and Options > Database Options > QUERY_PLAN_TEXT_ACCESS Option*
- *Reference: Statements and Options > Database Options > QUERY_PLAN_TEXT_CACHING Option*

See also

- *Query Evaluation Options* on page 67

Controlling Query Processing

Any user can set limits on the amount of time spent processing a particular query. Users with the SET ANY PUBLIC OPTION system privilege 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 , SAP Sybase IQ stops the query with an appropriate error.

Note: SAP 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 70
- *Setting Query Optimization Options* on page 71
- *Setting User-Supplied Condition Hints* on page 71
- *Monitoring Workloads* on page 72

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 users with the SET ANY SYSTEM OPTION system privilege to set an upper boundary on IQGOVERN_PRIORITY for a user or a role.
- 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 70
- *Setting Query Optimization Options* on page 71
- *Setting User-Supplied Condition Hints* on page 71
- *Monitoring Workloads* on page 72

Setting Query Optimization Options

Optimization options affect query processing speed.

- AGGREGATION_PREFERENCE – Controls the choice of algorithms for processing an aggregate (GROUP BY, DISTINCT, SET functions). This option is designed primarily for internal use; do not use it unless you are an experienced database administrator.
- DEFAULT_HAVING_SELECTIVITY_PPM – Sets the selectivity for all HAVING predicates in a query, overriding optimizer estimates for the number of rows that will be filtered by the HAVING clause.
- DEFAULT_LIKE_MATCH_SELECTIVITY_PPM – Sets the default selectivity for generic LIKE predicates, for example, LIKE '*string*%*string*' where % is a wildcard character. The optimizer relies on this option when other selectivity information is not available and the match string does not start with a set of constant characters followed by a single wildcard.
- DEFAULT_LIKE_RANGE_SELECTIVITY_PPM – Sets the default selectivity for leading constant LIKE predicates, of the form LIKE '*string*%' where the match string is a set of constant characters followed by a single wildcard character (%). The optimizer relies on this option when other selectivity information is not available.
- MAX_HASH_ROWS – Sets the maximum estimated number of rows the query optimizer will consider for a hash algorithm. The default is 2,500,000 rows. For example, if there is a join between two tables, and the estimated number of rows entering the join from both tables exceeds this option value, the optimizer will not consider a hash join. On systems with more than 50MB per user of TEMP_CACHE_MEMORY_MB, you may want to consider a higher value for this option.
- MAX_JOIN_ENUMERATION – Sets the maximum number of tables to be optimized for join order after optimizer simplifications have been applied. Normally you should not need to set this option.

See also

- *Setting Query Time Limits* on page 70
- *Setting Query Priority* on page 70
- *Setting User-Supplied Condition Hints* on page 71
- *Monitoring Workloads* on page 72

Setting User-Supplied Condition Hints

Selectivity hints help the optimizer choose an appropriate query strategy.

The SAP 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.

Additional Information

- *Reference: Building Blocks, Tables, and Procedures > SQL Language Elements > User-supplied Condition Hints*

See also

- *Setting Query Time Limits* on page 70
- *Setting Query Priority* on page 70
- *Setting Query Optimization Options* on page 71
- *Monitoring Workloads* on page 72

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.

SAP 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.

See also

- *Setting Query Time Limits* on page 70
- *Setting Query Priority* on page 70
- *Setting Query Optimization Options* on page 71
- *Setting User-Supplied Condition Hints* on page 71

Optimizing Delete Operations

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

HG Delete Operations

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

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

HG Delete Costing

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

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

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

Using HG Delete Performance Option

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

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

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

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

See also

- *WD Delete Operations* on page 74
- *TEXT Delete Operations* on page 75

WD Delete Operations

SAP 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 73
- *TEXT Delete Operations* on page 75

TEXT Delete Operations

SAP Sybase IQ chooses one of two algorithms to process delete operations on columns with a TEXT index.

- Small delete provides optimal performance when the rows deleted contain few distinct words, so that not many TEXT pages need to be visited. The TEXT small delete algorithm performs an ordered access to the TEXT. Worst case I/O is bounded by the number of index pages. Small delete incorporates the cost of sorting the words and record IDs in the records to delete.
- Large delete provides optimal performance when the rows deleted contain a large number of distinct words, and therefore need to visit a large number of "groups" in the index. The large delete scans the TEXT in order, until all rows are deleted. Worst case I/O is bounded by the number of index pages. Large delete is parallel, but parallelism is limited by the internal structure of the index and the distribution of groups from which to delete.

TEXT Delete Costing

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

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

Using TEXT Delete Performance Option

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

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

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

See also

- *HG Delete Operations* on page 73
- *WD Delete Operations* on page 74

Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W