SYBASE®

Performance and Tuning Guide

Sybase IQ

15.2

DOCUMENT ID: DC00169-01-1520-01

LAST REVISED: April 2010

Contents

About This Book ix

CHAPTER 1 **Selecting Data from Database Tables** **1**
- Prerequisites 2
- Viewing table information 2
- Ordering query results 4
- Selecting columns and rows 5
- Using search conditions 6
  - Comparing dates in queries 7
  - Compound search conditions in the WHERE clause 8
  - Pattern matching in search conditions 9
  - Matching rows by sound 10
  - Using shortcuts for search conditions 10
- Obtaining aggregate data 11
  - Using aggregate functions to obtain grouped data 12
  - Restricting groups 12
  - Improving subtotal calculation 14
- Obtaining analytical data 17
- Eliminating duplicate rows 18

CHAPTER 2 **Joining Tables** **21**
- Joining tables with the cross product 22
- Restricting a join 22
- How tables are related 24
  - Primary key identifiers key 24
  - Foreign keys for table relationships 24
- Join operators 25
  - Joining tables using key joins 25
  - Joining tables using natural joins 26
- Using ad hoc joins vs. join indexes 27
- Joins and data types 28
- Support for joins between stores or databases 29
- Querying remote and heterogeneous databases 30

Replacing joins with subqueries 31

CHAPTER 3 **Optimizing Queries and Deletions 35**
Tips for structuring queries 36
Impact on query performance of GROUP BY over a UNION ALL 36
Enhancing ORDER BY query performance 39
Enhanced parallelism within queries 39
Improved subquery performance 40
Using caching methods 40
Conditions that cause processing by SQL Anywhere 41
Planning queries 42
Query evaluation options 42
The query tree 44
Using query plans 44
Controlling query processing 46
Setting query time limits 46
Setting query priority 46
Setting query optimization options 47
Setting user-supplied condition hints 48
Monitoring workloads 49
Optimizing delete operations 50
HG delete operations 50
WD delete operations 51
TEXT delete operations 52

CHAPTER 4 **Managing System Resources 55**
Introduction to performance terms 56
Designing for performance 56
Overview of memory use 56
Paging increases available memory 57
Utilities to monitor swapping 58
Server memory 58
Managing buffer caches 59
Determining the sizes of the buffer caches 59
Setting buffer cache sizes 63
Specifying page size 64
Saving memory 66
Optimizing for large numbers of users 66
Platform-specific memory options 68
Other ways to get more memory 72
The process threading model 72
Insufficient threads error 72

Sybase IQ options for managing thread usage 73
Balancing I/O 73
Raw I/O (on UNIX operating systems) 74
Using disk striping 74
Internal striping 76
Using multiple files 77
Strategic file locations 77
Working space for inserting, deleting, and synchronizing 79
Setting reserved space options 80
Options for tuning resource use 80
Restricting concurrent queries 81
Setting the number of CPUS available 81
Limiting temporary dbspace use by a query 81
Limiting queries by rows returned 82
Forcing cursors to be non-scrolling 82
Limiting the number of cursors 83
Limiting the number of statements 83
Prefetching cache pages 83
Optimizing for typical usage 83
Controlling the number of prefetched rows 84
Other ways to improve resource use 84
Managing disk space in multiplex databases 84
Load balancing among query servers 85
Restricting database access 85
Disk caching 85
Indexing tips 86
Choosing the right index type 86
Using join indexes 87
Allowing enough disk space for deletions 87
Managing database size and structure 87
Managing the size of your database 88
Controlling index fragmentation 88
Minimizing catalog file growth 89
Denormalizing for performance 89
Denormalization has risks 89
Disadvantages of denormalization 90
Performance benefits of denormalization 90
Deciding to denormalize 90
Using UNION ALL views for faster loads 91
Optimizing queries that reference UNION ALL views 92
Managing UNION ALL view performance 93
Improved loading for large single (fact) tables 93
Network performance 94
Improving large data transfers 94

Isolate heavy network users 95
Put small amounts of data in small packets 95
Put large amounts of data in large packets 96
Process at the server level 96

CHAPTER 5 Monitoring and Tuning Performance 97
Viewing the Sybase IQ environment 98
Monitoring performance statistics.. 98
Getting information using stored procedures 106
Profiling database procedures... 107
Monitoring the buffer caches 115
Starting the buffer cache monitor 116
Checking results while the monitor runs............................... 122
Stopping the buffer cache monitor 122
Examining and saving monitor results................................. 122
Examples of monitor results 123
Buffer cache structure 127
Avoiding buffer manager thrashing 129
Monitoring paging on Windows systems 130
Monitoring paging on UNIX systems 131
Buffer cache monitor checklist 133
System utilities to monitor CPU use 136

CHAPTER 6 Tuning Servers on 32-bit Windows Systems 137
General performance guidelines 138
Maximizing throughput 138
Preventing memory over allocation 138
Monitoring physical memory.. 139
File systems 139
Monitoring performance 139
Monitoring virtual address space and working set 140
Monitoring page faults 141
Using the NTFS cache 141
Tuning inserts and queries 142
Characteristics of well-tuned insert operations...................... 142
Tuning for queries 143
Tuning backup operations 143

APPENDIX A Performance statistics 145
Field reference 146
Buffer manager... 147
Buffer pool 149

Prefetch manager 150
IQ store free list 150
Buffer allocator.. 151
Memory manager... 152
Thread manager 152
CPU time statistics.. 153
Transaction manager.. 153
Context Server statistics 153
Catalog, DB Log, and Repository statistics 154
Statistics 154
Page Types.. 160
Lock Statistics.. 160
Buffer Allocator/Prefetch Client Types................................... 161

Index.. 163

About This Book

Audience

This book provides performance and tuning recommendations for system and database administrators. This document assumes that you are familiar with relational database systems and Sybase® IQ. Use this guide in conjunction with other manuals in the documentation set.

Related Sybase IQ documents

The Sybase IQ 15.2 documentation set includes:

- *Release Bulletin* provides information about last-minute changes to the product and documentation.
- *Installation and Configuration Guide* provides platform-specific instructions on installing, migrating to a new version, and configuring Sybase IQ for a particular platform.
- *Advanced Security in Sybase IQ* covers the use of user encrypted columns within the Sybase IQ data repository. You need a separate license to install this product option.
- *Error Messages* lists Sybase IQ error messages referenced by Sybase error code, SQLCode, and SQLState, and SQL preprocessor errors and warnings.
- *IMSL Numerical Library User's Guide: Volume 2 of 2 C Stat Library* contains a concise description of the IMSL C Stat Library time series C functions. This book is only available to RAP – The Trading Edition™ Enterprise users.
- *Introduction to Sybase IQ* includes hands-on exercises for those unfamiliar with Sybase IQ or with the Sybase Central™ database management tool.
- *New Features Summary Sybase IQ 15.2* summarizes new features and behavior changes for the current version.
- *Performance and Tuning Guide* describes query optimization, design, and tuning issues for very large databases.
- *Quick Start* lists steps to build and query the demo database provided with Sybase IQ for validating the Sybase IQ software installation. Includes information on converting the demo database to multiplex.

- *Reference Manual* – Includes two reference guides to Sybase IQ:
    - *Reference: Building Blocks, Tables, and Procedures* describes SQL, stored procedures, data types, and system tables that Sybase IQ supports.
    - *Reference: Statements and Options* describes the SQL statements and options that Sybase IQ supports.
- *System Administration Guide* – Includes two volumes:
    - *System Administration Guide: Volume 1* describes startup, connections, database creation, population and indexing, versioning, collations, system backup and recovery, troubleshooting, and database repair.
    - *System Administration Guide: Volume 2* describes writing and running procedures and batches, programming with OLAP, accessing remote data, setting up IQ as an Open Server, scheduling and event handling, programming with XML, and debugging.
- *Time Series Guide* describes SQL functions used for time series forecasting and analysis. You need RAP – The Trading Edition™ Enterprise to use this product option.
- *Unstructured Data Analytics in Sybase IQ* explains storage and retrieval of unstructured data within the Sybase IQ data repository. You need a separate license to install this product option.
- *User-Defined Functions Guide* provides information about the user-defined functions, their parameters, and possible usage scenarios.
- *Using Sybase IQ Multiplex* tells how to use multiplex capability, designed to manage large query loads across multiple nodes.
- *Utility Guide* provides Sybase IQ utility program reference material, such as available syntax, parameters, and options.

Related SQL Anywhere documentation

Because Sybase IQ shares many components with SQL Anywhere Server, a component of the SQL Anywhere® package, Sybase IQ supports many of the same features as SQL Anywhere Server. The IQ documentation set refers you to SQL Anywhere documentation, where appropriate.

Documentation for SQL Anywhere includes:

- *SQL Anywhere Server – Database Administration* describes how to run, manage, and configure SQL Anywhere databases. It describes database connections, the database server, database files, backup procedures, security, high availability, and replication with Replication Server®, as well as administration utilities and options.
- *SQL Anywhere Server – Programming* describes how to build and deploy database applications using the C, C++, Java, PHP, Perl, Python, and .NET programming languages such as Visual Basic and Visual C#. This book also describes a variety of programming interfaces such as ADO.NET and ODBC.
- *SQL Anywhere Server – SQL Reference* provides reference information for system procedures, and the catalog (system tables and views). It also provides an explanation of the SQL Anywhere implementation of the SQL language (search conditions, syntax, data types, and functions).
- *SQL Anywhere Server – SQL Usage* describes how to design and create databases; how to import, export, and modify data; how to retrieve data; and how to build stored procedures and triggers.

You can also refer to the SQL Anywhere documentation in the SQL Anywhere 11.0.1 collection at Product Manuals at http://sybooks.sybase.com and in DocCommentXchange at http://dcx.sybase.com/dcx_home.php.

CHAPTER 1

Selecting Data from Database Tables

This chapter reviews basic query construction and recommends refinements to take advantage of product design. In this tutorial, you will look at table contents, order query results, select columns and rows, and use search conditions to refine queries.

For advanced query performance recommendations, see Chapter 3, "Optimizing Queries and Deletions."

Topic	Page
Prerequisites	2
Viewing table information	2
Ordering query results	4
Selecting columns and rows	5
Using search conditions	6
Obtaining aggregate data	11
Obtaining analytical data	17
Eliminating duplicate rows	18

Prerequisites

If you use a graphical front-end tool instead of Interactive SQL to query your database, the tool may allow you to view the SQL syntax it generates. For example, in InfoMaker, you can view SQL statements by choosing the SQL Syntax button on the Table painter bar.

This tutorial introduces the SELECT statement used to retrieve information from databases. SELECT statements are commonly called queries, because they ask the database server about information in a database.

Note The SELECT statement is a versatile command. SELECT statements can become highly complex in applications retrieving very specific information from large databases. This tutorial uses only simple SELECT statements: later tutorials describe more advanced queries. For more information about the full syntax of the select statement, see the SELECT statement in Chapter 1, “SQL Statements,” in *Reference: Statements and Options*.

Ideally, you should be running Sybase IQ software on your computer while you read and work through the tutorial lessons.

This tutorial assumes that you have already started Interactive SQL and connected to the sample database. If you have not already done so, see Chapter 2, “Using Interactive SQL (dbisql)” in the *Utility Guide*.

Viewing table information

In this section, you will look at the data in the Employees table.

The sample database you use in this tutorial is the same fictional company as in *Introduction to Sybase IQ*. The database contains information about employees, departments, sales orders, and so on. All the information is organized into tables.

Listing tables

In *Introduction to Sybase IQ*, you learned how to display a list of tables by opening the Tables folder in Sybase Central. You can also list user tables from interactive SQL using a system stored procedure, sp_iqtable. System stored procedures are system functions that are implemented as stored procedures in Sybase IQ.

In the SQL Statements window, type `sp_iqtable` to run the system stored procedure of the same name.

For complete details about this and other system stored procedures, see Chapter 7, “System Procedures” in *Reference: Building Blocks, Tables, and Procedures*.

Using the SELECT statement

In this lesson, you view one of the tables in the database. The command used will look at everything in a table called Employees.

Execute the command:

```
SELECT * FROM Employees
```

The asterisk is a short form for all the columns in the table.

The SELECT statement retrieves all the rows and columns of the Employees table, and the Interactive SQL Results window lists those that will fit:

EmployeeID	ManagerID	Surname	GivenName	DepartmentID	...
102	501	Whitney	Fran	100	...
105	501	Cobb	Matthew	100	...
129	902	Chin	Philip	200	...
148	1,293	Jordan	Julie	300	...
160	501	Breault	Robert	100	...
184	1,576	Espinoza	Melissa	400	...
191	703	Bertrand	Jeannette	500	...
195	902	Dill	Marc	200	...
207	1576	Francis	Jane	400	...

The Employees table contains a number of **rows** organized into **columns**. Each column has a name, such as Surname or EmployeeID. There is a row for each employee of the company, and each row has a value in each column. For example, the employee with EmployeeID 102 is Fran Whitney, whose manager is ManagerID 501.

You will also see some information in the Interactive SQL Messages window. This information is explained later.

Note Tablest in this document that display query results may only include some of the data returned by the query. Columns and rows with elliptical values indicate additional query results.

Case sensitivity

The Employees table name is shown starting with an uppercase E, even though the real table name is all lowercase. Sybase IQ databases can be created as case sensitive (the default) or case insensitive in their string comparisons, but are always case insensitive in their use of identifiers.

Note The examples in this book were created case insensitive, using the CREATE DATABASE qualifier CASE IGNORE. The default is CASE RESPECT, which gives better performance.

For information on creating databases, see Chapter 5, “Working with Database Objects,” in the *System Administration Guide: Volume 1*.

You can type select or Select instead of SELECT. Sybase IQ allows you to type keywords in uppercase, lowercase, or any combination of the two. In this manual, uppercase letters are generally used for SQL keywords.

Manipulation of the Interactive SQL environment and use of Interactive SQL is specific to the operating system.

For information on how to scroll through data and manipulate the Interactive SQL environment, see Chapter 2, “Using Interactive SQL (dbisql)” in *Utility Guide*.

Ordering query results

In this section, you will add an ORDER BY clause to the SELECT statement to display results in alphabetical or numerical order.

Unless otherwise requested, Sybase IQ displays the rows of a table in no particular order. Often it is useful to look at the rows in a table in a more meaningful sequence. For example, you might like to see employees in alphabetical order.

Listing employees in alphabetical order

The following example shows how adding an ORDER BY clause to the SELECT statement causes the results to be retrieved in alphabetical order.

```
SELECT * FROM Employees ORDER BY Surname
```

EmployeeID	ManagerID	Surname	GivenName	DepartmentID	...
1,751	1,576	Ahmed	Alex	400	...
1,013	703	Barker	Joseph	500	...
591	1,576	Barletta	Irene	400	...

EmployeeID	ManagerID	Surname	GivenName	DepartmentID	...
191	703	Bertrand	Jeannette	500	...
1,336	1,293	Bigelow	Janet	300	...
1,062	1,576	Blaikie	Barbara	400	...
750	703	Braun	Jane	500	...
160	501	Breault	Robert	100	...
1,191	1,576	Bucceri	Matthew	400	...

Notes

The order of the clauses is important. The ORDER BY clause must follow the FROM clause and the SELECT clause.

Note If you omit the FROM clause, or if all tables in the query are in the SYSTEM dbspace, the query is processed by SQL Anywhere instead of Sybase IQ and may behave differently, especially with respect to syntactic and semantic restrictions and the effects of option settings. See the SQL Anywhere documentation for rules that may apply to processing.

If you have a query that does not require a FROM clause, you can force the query to be processed by Sybase IQ by adding the clause `FROM iq_dummy`, where iq_dummy is a one-row, one-column table that you create in your database.

Selecting columns and rows

Often, you are interested in only columns in a table. For example, to make up birthday cards for employees you might want to see the Surname, DepartmentID and BirthDate columns.

Listing last name, department, and birth date of each employee

In this section, you will select each employee's birth date, last name, and department ID. Type the following:

```
SELECT Surname, DepartmentID, BirthDate
FROM Employees
```

Surname	DepartmentID	BirthDate
Whitney	100	1958-06-05
Cobb	100	1960-12-04
Chin	200	1966-10-30
Jordan	300	1951-12-13

Surname	DepartmentID	BirthDate
Breault	100	1947-05-13

Rearranging columns

The three columns appear in the order in which you typed them in the SELECT command. To rearrange the columns, simply change the order of the column names in the command. For example, to put the BirthDate column on the left, use the following command:

```
SELECT BirthDate, Surname, DepartmentID
FROM Employees
```

Ordering rows

You can order rows and look at only certain columns at the same time as follows:

```
SELECT BirthDate, Surname, DepartmentID
FROM Employees
ORDER BY Surname
```

The asterisk in

```
SELECT * FROM Employees
```

is a short form for all columns in the table.

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

Using search conditions

In this section you will learn procedures for comparing dates, using compound search conditions in the WHERE clause, pattern matching, and search condition shortcuts.

Sometimes you will not want to see information on all the employees in the Employees table. Adding a WHERE clause to the SELECT statement allows only some rows to be selected from a table.

For example, suppose you would like to look at employees with the first name of John.

❖ **Listing all employees named John:**

- Type the following:

```
SELECT *
FROM Employees
WHERE GivenName = 'John'
```

EmployeeID	ManagerID	Surname	GivenName	DepartmentID	...
318	1,576	Crow	John	400	...
862	501	Sheffield	John	100	...
1,483	1,293	Letiecq	John	300	...

Apostrophes and case-sensitivity

- The apostrophes (single quotes) around the name 'John' are required. They indicate that John is a character string. Quotation marks (double quotes) have a different meaning. Quotation marks can be used to make otherwise invalid strings valid for column names and other identifiers.
- The sample database is not case sensitive, so you would get the same results whether you searched for ' 'JOHN', 'john', or 'John'.

Again, you can combine what you have learned:

```
SELECT GivenName, Surname, BirthDate
FROM Employees
WHERE GivenName = 'John'
ORDER BY BirthDate
```

Notes

- How you order clauses is important. The FROM clause comes first, followed by the WHERE clause, and then the ORDER BY clause. If you type the clauses in a different order, you will get a syntax error.
- You do not need to split the statement into several lines. You can enter the statement into the SQL Statements window in any format. If you use more than the number of lines that fit on the screen, the text scrolls in the SQL Statements window.

Comparing dates in queries

Sometimes you will not know exactly what value you are looking for, or you would like to see a set of values. You can use comparisons in the WHERE clause to select a set of rows that satisfy the search condition.

Listing employees born before March 3, 1964

The following example shows the use of a date inequality search condition. Type the following:

```
SELECT Surname, BirthDate
```

```
FROM Employees
WHERE BirthDate < 'March 3, 1964'
```

Surname	**BirthDate**
Whitney	1958-06-05
Cobb	1960-12-04
Jordan	1951-12-13
Breault	1947-05-13
Espinoza	1939-12-14
Dill	1963-07-19
Francis	1954-09-12
Shishov	1949-04-22
...	...

Sybase IQ knows that the BirthDate column contains a date, and converts *'March 3, 1964'* to a date automatically.

Compound search conditions in the WHERE clause

So far, you have seen equal (=) and less than (<) as comparison operators. Sybase IQ also supports other comparison operators, such as greater than (>), greater than or equal (>=), less than or equal (<=), and not equal (<>).

These conditions can be combined using AND and OR to make more complicated search conditions.

Qualifying the list

To list all employees born before March 3, 1964, but exclude the employee named Whitney:

```
SELECT Surname, BirthDate
FROM Employees
WHERE BirthDate < '1964-3-3'
AND Surname <> 'Whitney'
```

Surname	**BirthDate**
Cobb	1960-12-04
Jordan	1951-12-13
Breault	1947-05-13
Espinoza	1939-12-14
Dill	1963-07-19
Francis	1954-09-12

Surname	**BirthDate**
Shishov	1949-04-22
...	...

Pattern matching in search conditions

Another useful way to look for things is to search for a pattern. In SQL, the word LIKE is used to search for patterns. The use of LIKE can be explained by example.

Listing specific employees

To list all employees whose surname begins with "br" type the following:

```
SELECT Surname, GivenName
FROM Employees
WHERE Surname LIKE 'br%'
```

Surname	**GivenName**
Breault	Robert
Braun	Jane

The % in the search condition indicates that any number of other characters may follow the letters BR.

Qualifying the surname search

To list all employees whose surname begins with BR, followed by zero or more letters and a T, followed by zero or more letters, type:

```
SELECT Surname, GivenName
FROM Employees
WHERE Surname LIKE 'BR%T%'
```

Surname	**GivenName**
Breault	Robert

The first % sign matches the string “eaul”, and the second % sign matches the empty string (no characters).

Another special character that can be used with LIKE is the _ (underscore) character, which matches exactly one character.

The pattern BR_U% matches all names starting with BR and having U as the fourth letter. In Braun, the _ matches the letter A, and the % matches N.

Matching rows by sound

With the SOUNDEX function, you can match rows by sound, as well as by spelling. For example, suppose a phone message was left for a name that sounded like “Ms. Brown”. Which employees in the company have names that sound like Brown?

Searching surnames by sound

To list employees with surnames that sound like Brown, type the following:

```
SELECT Surname, GivenName
FROM Employees
WHERE SOUNDEX( Surname ) = SOUNDEX( 'Brown' )
```

Surname	GivenName
Braun	Jane

Jane Braun is the only employee matching the search condition.

Using shortcuts for search conditions

Using the short form BETWEEN

SQL has two short forms for typing in search conditions. Use the first, BETWEEN, when you are looking for a range of values. For example, the following two example queries are equal:

```
SELECT Surname, BirthDate
FROM Employees
WHERE BirthDate BETWEEN '1964-1-1'
AND '1965-3-31'
```

```
SELECT Surname, BirthDate
FROM Employees
WHERE BirthDate >= '1964-1-1'
AND BirthDate <= '1965-3-31'
```

Using the short form IN

Use the second short form, IN, to look for one of a number of values. The command. The following two example queries are equal.

```
SELECT Surname, EmployeeID
FROM Employees
WHERE Surname IN ('Yeung','Bucceri','Charlton')
```

```
SELECT Surname, EmployeeID
FROM Employees
WHERE Surname = 'Yeung'
OR Surname = 'Bucceri'
OR Surname = 'Charlton'
```

Obtaining aggregate data

This section tells how to construct queries that give you aggregate information. Examples of aggregate information are:

- The total of all values in a column
- The number of entries in a column
- The average value of entries in a column

A first look at aggregate functions

Suppose you want to know how many employees are in the database. The following statement retrieves the number of rows in the employee table:

```
SELECT count( * )
FROM Employees
```

The result returned from this query is a table with only one column (with title count(*)) and one row, which contains the number of employees.

Count()
75

The following command is a slightly more complicated aggregate query:

```
SELECT   count( * ),
min( BirthDate ),
max( BirthDate )
FROM Employees
```

The result set from this query has three columns and only one row. The three columns contain the number of employees, the birth date of the oldest employee, and the birth date of the youngest employee.

Count()	Min(BirthDate)	Max(BirthDate)
75	'1936-01-02'	'1973-01-18'

COUNT, MIN, and MAX are called aggregate functions. Aggregate functions summarize entire tables from the database using the GROUP BY clause of the SELECT statement. In total, there are seven aggregate functions: AVG, COUNT, MAX, MIN, STDDEV, SUM, and VARIANCE. All of the functions have either the name of a column or an expression as a parameter. As you have seen, COUNT also has an asterisk as its parameter, which returns the number of rows in each group.

Using aggregate functions to obtain grouped data

In addition to providing information about an entire table, aggregate functions can be used on groups of rows.

Using an aggregate function on groups of rows

To list the number of orders for which each sales representative is responsible, type:

```
SELECT SalesRepresentative, count( * )
FROM SalesOrders
GROUP BY SalesRepresentative
```

SalesRepresentative	Count()
129	57
195	50
299	114
467	56
667	54
690	52
856	55
902	47
949	53
1,142	57
1,596	53

The results of this query consist of one row for each SalesRepresentative ID number, containing the SalesRepresentative ID, and the number of rows in the SalesOrders table with that ID number.

Whenever GROUP BY is used, the resulting table has one row for each different value found in the GROUP BY column or columns.

Restricting groups

You have already seen how to restrict rows in a query using the WHERE clause. You can restrict GROUP BY clauses by using the HAVING keyword.

Restricting GROUP BY clauses

To list all sales reps with more than 55 orders, type:

```
SELECT SalesRepresentative, count( * )
FROM  SalesOrders
GROUP BY SalesRepresentative
HAVING count( * ) > 55
```

SalesRepresentative	count(*)
129	57
299	114
467	56
1,142	57

Note GROUP BY must always appear before HAVING. In the same manner, WHERE must appear before GROUP BY.

Using WHERE and GROUP BY

To list all sales reps with more than 55 orders and an ID of more than 1000, type:

```
SELECT SalesRepresentative, count( * )
FROM SalesOrders
WHERE SalesRepresentative > 1000
GROUP BY SalesRepresentative
HAVING count( * ) > 55
```

The Sybase IQ query optimizer moves predicates from the HAVING clause to the WHERE clause, when doing so provides a performance gain. For example, if you specify:

```
GROUP BY SalesRepresentative
HAVING count( *) > 55
 AND SalesRepresentative > 1000
```

instead of the WHERE clause in the preceding example, the query optimizer moves the predicate to a WHERE clause.

Sybase IQ performs this optimization with simple conditions (nothing involving OR or IN). For this reason, when constructing queries with both a WHERE clause and a HAVING clause, you should be careful to put as many of the conditions as possible in the WHERE clause.

Improving subtotal calculation

If you have data that varies across dimensions such as date or place, you may need to determine how the data varies in each dimension. You can use the ROLLUP and CUBE operators to create multiple levels of subtotals and a grand total from a list of references to grouping columns. The subtotals "roll up" from the most detailed level to the grand total. For example, if you are analyzing sales data, you can compute an overall average and the average sales by year using the same query.

Using ROLLUP

To select total car sales by year, model and color:

```
SELECT Name, Size, Color, Sum(Quantity)
FROM Products
GROUP BY ROLLUP (Name, Size, Color);
```

Name	Size	Color	sum(Products.Quantity)
Baseball Cap	One size fits all	Black	112
Baseball Cap	One size fits all	White	12
Baseball Cap	One size fits all	(NULL)	124
Baseball Cap	(NULL)	(NULL)	124
Shorts	Medium	Black	80
Shorts	Medium	(NULL)	80
Shorts	(NULL)	(NULL)	80
Sweatshirt	Large	Blue	32
Sweatshirt	Large	Green	39
Sweatshirt	Large	(NULL)	71
Sweatshirt	(NULL)	(NULL)	71
Tee Shirt	Medium	Orange	54
Tee Shirt	Medium	(NULL)	54
Tee Shirt	One size fits all	Black	75
Tee Shirt	One size fits all	(NULL)	75
Tee Shirt	Small	White	28
Tee Shirt	Small	(NULL)	28
Tee Shirt	(NULL)	(NULL)	157
Visor	One size fits all	Black	28
Visor	One size fits all	White	36
Visor	One size fits all	(NULL)	64
Visor	(NULL)	(NULL)	64
(NULL)	(NULL)	(NULL)	496

When processing this query, Sybase IQ first groups the data by all three specified grouping expressions (year, model, color), then for all grouping expressions except the last one (color). In the fifth row, NULL indicates the ROLLUP value for the color column, in other words, the total number of sales of that model in all colors. 343 represents the total sales of all models and colors in 1990 and 314 is the total for 1991. The last row represents total sales on all years, all models and all colors.

ROLLUP requires an ordered list of grouping expressions as arguments. When listing groups that contain other groups, list the larger group first (such as state before city.)

You can use ROLLUP with the aggregate functions: AVG, COUNT, MAX, MIN, STDDEV, SUM, and VARIANCE. ROLLUP does not support COUNT DISTINCT and SUM DISTINCT, however.

Using CUBE

The following query uses data from the Employees table, including the state (geographic location), gender, education level, and income of people. You can use the CUBE extension of the GROUP BY clause, if you want to compute the average income in the entire census of state, gender, and education and compute the average income in all possible combinations of the columns state, gender, and education, while making only a single pass through the census data in the table census. For example, use the CUBE operator if you want to compute the average income of all females in all states, or compute the average income of all people in the census according to their education and geographic location.

When CUBE calculates a group, CUBE puts a NULL value in the columns whose group is calculated. The distinction is difficult between the type of group each row represents and whether the NULL is a NULL stored in the database or a NULL resulting from CUBE. The GROUPING function solves this problem by returning 1, if the designated column has been merged to a higher level group.

The following query illustrates the use of the GROUPING function with GROUP BY CUBE.

```
SELECT case grouping(state) WHEN 1 THEN 'ALL' ELSE state
   END AS c_state, case grouping(sex) WHEN 1 THEN 'ALL'
   ELSE sex end AS c_gender, case
grouping(DepartmentId)
   WHEN 1 THEN 'ALL' ELSE cast(DepartmentId as char(4))
end
   AS c_dept, COUNT(*), CAST(ROUND(AVG(salary),2) AS
   NUMERIC(18,2))AS AVERAGE
FROM employees WHERE state IN ('MA' , 'CA')
```

```
GROUP BY CUBE(state, sex, DepartmentId)
ORDER BY 1,2,3;
```

The results of this query are shown below. Note that the NULLs generated by CUBE to indicate a subtotal row are replaced with ALL in the subtotal rows, as specified in the query.

c_state	c_gender	c_dept	COUNT()	AVERAGE
ALL	ALL	200	3	52,200.00
ALL	ALL	ALL	3	52,200.00
ALL	F	200	2	58,650.00
ALL	F	ALL	2	58,650.00
ALL	M	200	1	39,300.00
ALL	M	ALL	1	39,300.00
CA	ALL	200	3	52,200.00
CA	ALL	ALL	3	52,200.00
CA	F	200	2	58,650.00
CA	F	ALL	2	58,650.00
CA	M	200	1	39,300.00
CA	M	ALL	1	39,300.00

Data warehouse administrators find ROLLUP and CUBE particularly useful for operations like:

- Subtotaling on a hierarchical dimension like geography or time, for example year/month/day or country/state/city
- Populating summary tables

ROLLUP and CUBE allow you to use one query to compute data using multiple levels of grouping, instead of a separate query for each level.

For more information on the ROLLUP and CUBE operators, see the SELECT statement in “SQL Statements” in *Reference: Statements and Options*.

Obtaining analytical data

This section tells how to construct queries that give you analytical information. There are two types of analytical functions: rank and inverse distribution. The rank analytical functions rank items in a group, compute distribution, and divide a result set into a number of groupings. The inverse distribution analytical functions return a k-th percentile value, which can be used to help establish a threshold acceptance value for a set of data.

The rank analytical functions are RANK, DENSE_RANK, PERCENT_RANK, and NTILE. The inverse distribution analytical functions are PERCENTILE_CONT and PERCENTILE_DISC.

Suppose you want to rank employee salaries. In the following example, the NTILE function divides employees into four groups based on the employee’s salary. Employees whose ntile ranking = 1 are in the top 25% salary range.

```
SELECT Name
   Salary,
   NTILE(4) OVER(ORDER BY salary DESC)as Ranking
FROM emp1;
```

Name	Salary	Ranking
Sandy	55,000	1
Peter	48,000	1
Lisa	38,000	1
Scott	29,000	1
Tim	29,000	2
Tom	28,000	2
Mike	28,000	2
Adam	25,000	3
Antonia	22,000	3
Jim	22,000	3
Anna	18,000	4
Jeff	18,000	4
Amy	18,000	4

NTILE is a analytical function that distributes or ranks query results into a specified number of buckets and assigns the bucket number to each row in the bucket. You can divide a result set into tenths (deciles), fourths (quartiles), and other numbers of groupings.

The rank analytical functions require an OVER (ORDER BY) clause. The ORDER BY clause specifies the parameter on which ranking is performed and the order in which the rows are sorted in each group. Note that this ORDER BY clause is used only within the OVER clause and is *not* an ORDER BY for the SELECT.

The OVER clause indicates that the function operates on a query result set. The result set is the rows that are returned after the FROM, WHERE, GROUP BY, and HAVING clauses have all been evaluated. The OVER clause defines the data set of the rows to include in the computation of the rank analytical function.

Similarly, the inverse distribution functions require a WITHIN GROUP (ORDER BY) clause. The ORDER BY specifies the expression on which the percentile function is performed and the order in which the rows are sorted in each group. This ORDER BY clause is used only within the WITHIN GROUP clause and is *not* an ORDER BY for the SELECT. The WITHIN GROUP clause distributes the query result into an ordered data set from which the function calculates a result.

For more details on the analytical functions, see “Analytical functions,” Chapter 4, “SQL Functions,” in *Reference: Building Blocks, Tables, and Procedures* For information on individual analytical functions, see the section for each function in the same chapter.

Eliminating duplicate rows

Result tables from SELECT statements can contain duplicate rows. You can use the DISTINCT keyword to eliminate the duplicates. For example, the following command returns many duplicate rows:

```
SELECT city, state FROM Employees
```

To list only unique combinations of city and state, use this command:

```
SELECT DISTINCT city, state FROM Employees
```

Note The ROLLUP and CUBE operators do not support the DISTINCT keyword.

This chapter provides an overview of single-table SELECT statements. For more information about single-table SELECT statements, see

- Chapter 5, “Working with Database Objects,” in the *System Administration Guide: Volume 1*
- Chapter 2, “SQL Language Elements,” in *Reference: Building Blocks, Tables, and Procedures*
- “SELECT statement” in Chapter 1, “SQL Statements,” in *Reference: Statements and Options*

Advanced uses of the SELECT statement are described in the next chapter.

CHAPTER 2 Joining Tables

This chapter explains how to look at information in more than one table and describes various types of joins. You will complete tutorial tasks on joining tables.

Topic	Page
Joining tables with the cross product	22
Restricting a join	22
How tables are related	24
Join operators	25
Using ad hoc joins vs. join indexes	27
Joins and data types	28
Support for joins between stores or databases	29
Querying remote and heterogeneous databases	30
Replacing joins with subqueries	31

Joining tables with the cross product

One of the tables in the sample database is FinancialData, which lists the financial data for the company. Each data record has a code column that identifies its department and whether it is an expense or revenue record.

You can get information from two tables at the same time by listing both tables, separated by a comma, in the FROM clause of a SELECT query.

Example

The following SELECT command lists all the data in the FinancialCodes and FinancialData tables:

```
SELECT *
FROM FinancialCodes, FinancialData
```

The results of this query, displayed in the Interactive SQL data window, match every row in the FinancialCodes table with every row in the FinancialData table.This join is called a full cross product, also known as a cartesian product. Each row consists of all columns from the FinancialCodes table followed by all columns from the FinancialData table.

The cross product join is a simple starting point for understanding joins, but it is not very useful in itself. Subsequent sections in this chapter tell how to construct more selective joins, which you can think of as applying restrictions to the cross product table.

Restricting a join

To make a cross product join useful, you normally want to include only rows that satisfy some condition in the result. That condition, called the join condition, compares one column from one table to one column in the other table, using a comparison operator, such as =, =>, <, and so on. Thus, you eliminate some of the rows from the cross-product result.

For example, to make the join in the preceding section useful, you could insist that the SalesRepresentative in the SalesOrders table be the same as the one in the Employees table in every row of the result. Then each row contains information about an order and the sales representative responsible for it.

Example 1

To do this, add a WHERE clause to the previous query to show the list of employees and their course registrations:

```
SELECT *
FROM SalesOrders, Employees
```

```
WHERE SalesOrders.SalesRepresentative =
Employees.EmployeeID
```

The table name is given as a prefix to identify the columns. Although not strictly required in this case, using the table name prefix clarifies the statement, and is required when two tables have a column with the same name. A table name used in this context is called a **qualifier**.

Example 2

The following query is a modified version that fetches only some of the columns and orders the results:

```
SELECT Employees.Surname, SalesOrders.id,
SalesOrders.OrderDate
FROM SalesOrders, Employees
WHERE SalesOrders.SalesRepresentative =
Employees.EmployeeID
ORDER BY Employees.Surname
```

If there are many tables in a SELECT command, you may need to type several qualifier names. You can reduce typing by using a correlation name.

Correlation names

A correlation name is an alias for a particular instance of a table. This alias is valid only within a single statement. Correlation names are created by putting a short form for a table name immediately after the table name, separated by the keyword AS. Then you *must* use the short form as a qualifier instead of the corresponding table name:

```
SELECT E.Surname, S.id, S.OrderDate
FROM SalesOrders AS S, Employees AS E
WHERE S.SalesRepresentative = E.EmployeeID
ORDER BY E.Surname
```

Here, two correlation names S and E are created for the SalesOrders and Employees tables.

Note A table name or correlation name is needed only to resolve ambiguity if two columns of different tables have the same name. If you have created a correlation name, you must use it instead of the full table name; however, if you have not created a correlation name, use the full table name.

How tables are related

To construct other types of joins, you must first understand how the information in one table is related to that in another.

The primary key for a table identifies each row in the table. Tables are related to each other using a foreign key.

This section shows how using primary and foreign keys together lets you construct queries from more than one table.

Primary key identifiers key

Every table in the iqdemo database has a primary key. A primary key is one or more columns that uniquely identify a row in the table. For example, an employee number uniquely identifies an employee - EmployeeID is the primary key of the Employees table.

The SalesOrderItems table is an example of a table with two columns that make up the primary key. The order ID by itself does not uniquely identify a row in the SalesOrderItems table because there can be several items in an order. Also, the LineID number does not uniquely identify a row in the SalesOrderItems table. Both the order ID name and LineID are required to uniquely identify a row in the SalesOrderItems table. Therefore, the primary key of the table is both columns taken together.

Foreign keys for table relationships

Several tables in the iqdemo database refer to other tables in the database. For example, in the SalesOrders table, the SalesRepresentative column indicates which employee is responsible for an order. Only enough information to uniquely identify an employee is kept in the SalesOrders table. The SalesRepresentative column in the SalesOrders table is a foreign key to the Employees table.

A foreign key is one or more columns that contain candidate key values from another table. (For more about candidate keys, see Chapter 1, “SQL Statements,” in the *System Administration Guide: Volume 1*.)

Join operators

Many common joins are between two tables related by a key. The most common join restricts foreign key values in one table to be equal to primary key values in another table. The example you have already seen restricts foreign key values in the SalesOrders table to be equal to the candidate key values in the Employees table.

```
SELECT Surname,
  EmployeeID,
  OrderDate
FROM SalesOrders, Employees
WHERE SalesOrders.SalesRepresentative =
Employees.EmployeeID
```

The query can be more simply expressed using a KEY JOIN, as described in the following section.

Joining tables using key joins

Key joins are an easy way to join tables related by a foreign key. The following example returns the same results as a query with a WHERE clause that equates the two employee ID number columns:

```
SELECT Surname,
  EmployeeID,
  OrderDate
FROM SalesOrders
KEY JOIN Employees
```

```
SELECT Surname,
  EmployeeID,
  OrderDate
FROM SalesOrders, Employees
WHERE SalesOrders.SalesRepresentative =
Employees.EmployeeID
```

The join operator (KEY JOIN) is just a short cut for typing the WHERE clause; the two queries are identical.

In the diagram of the iqdemo database, in *Introduction to Sybase IQ*, foreign keys are represented by lines between tables. Anywhere that two tables are joined by a line in the diagram, you can use the KEY JOIN operator. Remember that your application must enforce foreign keys in order to ensure expected results from queries based on key joins.

Joining two or more tables

Two or more tables can be joined using join operators. The following query uses four tables to list the total value of the orders placed by each customer. It connects the four tables customer, SalesOrders, SalesOrderItems and Products single foreign-key relationships between each pair of these tables.

```
SELECT CompanyName,
  CAST( SUM(SalesOrderItems.Quantity *
  Products.UnitPrice) AS INTEGER) AS Value
FROM Customers
KEY JOIN SalesOrders
KEY JOIN SalesOrderItems
KEY JOIN Products
GROUP BY CompanyName
```

CompanyName	Value
The Power Group	5,808
The Birds Loft	4,404
Sampson &Sons	6,660
Hats Etc.	2,736
Howard Co.	5,388
...	...

The CAST function used in this query converts the data type of an expression. In this example the sum that is returned as an integer is converted to a value.

Joining tables using natural joins

The NATURAL JOIN operator joins two tables based on common column names. In other words, Sybase IQ generates a WHERE clause that equates the common columns from each table.

Example

For example, for the following query:

```
SELECT Surname,
  DepartmentName
FROM Employees
NATURAL JOIN Departments
```

the database server looks at the two tables and determines that the only column name they have in common is DepartmentID. The following ON phrase is internally generated and used to perform the join:

```
FROM Employees JOIN Departments
...
ON Employees.DepartmentID = Departments.DepartmentID
```

Errors using NATURAL JOIN

This join operator can cause problems by equating columns you may not intend to be equated. For example, the following query generates unwanted results:

```
SELECT *
FROM SalesOrders
NATURAL JOIN Customers
```

The result of this query has no rows.

The database server internally generates the following ON phrase:

```
FROM SalesOrders JOIN Customers
   ON SalesOrders.ID = Customers.ID
```

The id column in the SalesOrders table is an ID number for the order. The id column in the customer table is an ID number for the customer. None of them matched. Of course, even if a match were found, it would be a meaningless one.

You should be careful using join operators. Always remember that the join operator just saves you from typing the WHERE clause for an unenforced foreign key or common column names. Be mindful of the WHERE clause, or you may create queries that give results other than what you intend.

Using ad hoc joins vs. join indexes

If you have defined join indexes on the join columns referenced in your query, Sybase IQ will usually use them to execute queries joining those tables. (For information about defining join indexes, see Chapter 6, “Using Sybase IQ Indexes,” in the *System Administration Guide: Volume 1*.)

Any join that does not use join indexes is known as an **ad hoc join**. If several tables are referenced by the query, and not all of them have join indexes defined, Sybase IQ will use the join indexes for those tables that have them in combination with an ad hoc join with the rest of the tables.

Because you cannot create join indexes for all possible joins, ad hoc joins may sometimes be necessary. Thanks to optimizations in Sybase IQ, you may find that queries perform as well or better without join indexes.

Keep these rules in mind when creating join indexes:

- All join indexes are created using full outer joins. A query using a join index can be an inner, left outer, or right outer join though.

A full outer join is one where *all* rows from both the left and right specified tables are included in the result, with NULL returned for any column with no matching value in the corresponding column.

- The only comparison operator that may be used in the join predicate ON clause is EQUALS.
- You can use the NATURAL keyword instead of an ON clause, but you can only specify one pair of tables.
- Join index columns must have identical data type, precision, and scale.
- Join indexes tend to perform best compared to ad-hoc joins when the tables involved have similar numbers of rows. Join indexes perform less well compared to ad-hoc joins when there is a very large difference between the larger and smaller table.

Joins and data types

Join columns require like data types for optimal performance. Sybase IQ allows you to make an ad hoc join on any data types for which an implicit conversion exists. Unless join column data types are identical, however, performance can suffer to varying degrees, depending on the data types and the size of the tables. For example, while you can join an INT to a BIGINT column, this join prevents certain types of optimizations. The Sybase IQ index advisor can identify mis-matched join data types that can impact performance in cases like this.

Join keys with smaller data types tend to offer better performance than keys with wider data types; join keys with integer data types tend to be faster than numeric or character data types.

Although these data types may offer better performance, choosing keys with matching data types usually provides more efficient joints that choosing keys with ‘fast’ data types that do not match. If the data types are not the same, Sybase IQ must internally convert one of the data types to make the columns comparable, which can decrease performance.

For tables of implicit data type conversions, see Chapter 7, “Moving Data In and Out of Databases” in *System Administration Guide: Volume 1*.

Support for joins between stores or databases

This section clarifies current support for joins between stores or between databases.

Joining tables within a Sybase IQ database

Any joins within a given Sybase IQ database are supported. This means that you can join any system or user tables in the Catalog Store with any tables in the IQ Store, in any order.

Joining Adaptive Server Enterprise and Sybase IQ tables

Joins of Sybase IQ tables with tables in an Adaptive Server Enterprise database are supported under the following conditions:

- The Sybase IQ database can be either the local database or the remote database.
- If a Sybase IQ table is to be used as a proxy table in ASE, the table name must be 30 characters or fewer.
- In order to join a local Adaptive Server Enterprise table with a remote Sybase IQ 12 table, the ASE version must be 11.9.2 or higher, and you must use the correct server class:
    - To connect from a front end of Adaptive Server Enterprise 12.5 or higher to a remote Sybase IQ 12.5 or higher, use the ASIQ server class, which was added in ASE 12.5.
    - To connect from a front end of Adaptive Server Enterprise 11.9.2 through 12.0 to a remote Sybase IQ 12.x (or SQL Anywhere 6.x or higher), you must use server class ASAnywhere.
- When you join a local Sybase IQ table with any remote table, the local table must appear first in the FROM clause, which means the local table is the outermost table in the join.

Joins between Sybase IQ and Adaptive Server Enterprise rely on Component Integration Services (CIS).

For more information on queries from Adaptive Server Enterprise databases to Sybase IQ, see *Component Integration Services Users's Guide* in the Adaptive Server Enterprise core documentation set.

For more information on queries from Sybase IQ to other databases, see “Querying remote and heterogeneous databases.”

Joining SQL Anywhere and Sybase IQ tables

The CHAR data type is incompatible between SQL Anywhere and Sybase IQ when the database is built with BLANK PADDING OFF. If you want to perform cross-database joins between SQL Anywhere and Sybase IQ tables using character data as the join key, use the CHAR data type with BLANK PADDING ON.

Note Sybase IQ CREATE DATABASE no longer supports BLANK PADDING OFF for new databases. This change has no effect on existing databases. You can test the state of existing databases using the BlankPadding database property:

```
select db_property ( ‘BlankPadding’ )
```

Sybase recommends that you change any existing columns affected by BLANK PADDING OFF, to ensure correct join results. Recreate join columns as CHAR data type, rather than VARCHAR. CHAR columns are always blank padded.

Querying remote and heterogeneous databases

This section summarizes how you use Sybase IQ with Component Integration Services (CIS). CIS allows you to query Adaptive Server Enterprise databases and remote databases or nonrelational data sources through Sybase IQ. CIS is installed as part of Sybase IQ.

Using CIS, you can access tables on remote servers as if the tables were local. CIS performs joins between tables in multiple remote, heterogeneous servers and transfers the contents of one table into a supported remote server.

To query a remote database or data source, you need to map its tables to local proxy tables. CIS presents proxy tables to a client application as if the data were stored locally. When you query the tables, CIS determines the actual server storage location.

❖ **To join remote databases:**

1. Create proxy tables, following the steps in the *System Administration Guide: Volume 2*.
2. Map the remote tables to the proxy tables.

3 Reference the proxy tables in your SELECT statement, using the proxy database name as the qualifying name for each remote table. For example:

```
SELECT a.c_custkey, b.o_orderkey
FROM proxy_iqdemo..cust2 a,
iqdemo..orders b
WHERE a.c_custkey = b.o_custkey
```

For more information, see Chapter 4, “Accessing Remote Data” and Chapter 5, “Server Classes for Remote Data Access” in *System Administration Guide: Volume 2*.

Replacing joins with subqueries

A join returns a result table constructed from data from multiple tables. You can also retrieve the same result table using a subquery. A subquery is simply a SELECT statement within another select statement. This is a useful tool in building more complex and informative queries.

For example, suppose you need a chronological list of orders and the company that placed them, but would like the company name instead of their customer ID. You can get this result using a join as follows:

Using a join

To list the order_id, OrderDate, and CompanyName for each order from the beginning of 1994, type:

```
SELECT  SalesOrders.ID,
SalesOrders.OrderDate,
Customers.CompanyName
FROM SalesOrders
KEY JOIN Customers
WHERE OrderDate > '1994/01/01'
ORDER BY OrderDate
```

ID	OrderDate	CompanyName
2131	2000-01-02	BoSox Club
2126	2000-01-03	Leisure Time
2065	2000-01-03	Bloomfields
2127	2000-01-06	Creative Customs Inc.
2135	2000-01-06	East Coast Traders
2129	2000-01-07	Hospital Gifts
2132	2000-01-08	The Pep Squad

ID	OrderDate	CompanyName
2136	2000-01-09	Divas Design
2133	2000-01-10	The Road Side Inn
2083	2000-01-13	Pollys Custom Design

Using an outer join

The join in previous sections of the tutorial is more fully called an **inner join**.

You specify an **outer join** explicitly. In this case, a GROUP BY clause is also required:

```
SELECT  CompanyName,
MAX( SalesOrders.ID ),State
FROM Customers
KEY LEFT OUTER JOIN SalesOrders
WHERE State = 'WA'
GROUP BY CompanyName, State
```

CompanyName	MAX(SalesOrders.ID)	State
Its a Hit!	(NULL)	WA
Custom Designs	2547	WA

Using a subquery

To list order items for products low in stock, type:

```
SELECT *
FROM SalesOrderItems
WHERE ProductID IN
 ( SELECT ID
FROM Products
WHERE Quantity < 20 )
ORDER BY ShipDate DESC
```

ID	LineID	ProductID	Quantity	ShipDate
2082	1	401	48	2001-07-09
2053	1	401	60	2001-06-30
2125	2	401	36	2001-06-28
2027	1	401	12	2001-06-17
2062	1	401	36	2001-06-17
2023	1	401	24	2001-06-09
2031	2	401	48	2001-06-02
2073	1	401	24	2001-06-02
2647	1	401	36	2001-05-26
2639	1	401	36	2001-05-19

The subquery in the statement is the phrase enclosed in parentheses:

```
(  SELECT ID
FROM Products
WHERE Quantity < 20 )
```

By using a subquery, the search can be carried out in just one query, instead of using one query to find the list of low-stock products and a second to find orders for those products.

The subquery makes a list of all values in the id column in the product table satisfying the WHERE clause search condition.

Remember the following notes about subqueries:

- Subqueries may also be useful in cases where you may have trouble constructing a join, such as queries that use the NOT EXISTS predicate.
- Subqueries can only return one column.
- Subqueries are allowed only as arguments of comparisons, IN, or EXISTS clauses.
- Subqueries cannot be used inside an outer join ON clause.

CHAPTER 3

Optimizing Queries and Deletions

This chapter offers query and deletion performance recommendations

Topic	**Page**
Conditions that cause processing by SQL Anywhere	36
Planning queries	42
Controlling query processing	46
Optimizing delete operations	50

Tips for structuring queries

The following list illustrates hints for improving query structure:

- In some cases, command statements that include subqueries can also be formulated as joins and may run faster.
- If you group on multiple columns in a GROUP BY clause, list the columns by descending order by number of unique values if you can. This will give you the best query performance.
- Join indexes can often cause join queries to execute faster than ad hoc joins, at the expense of using more disk space and significantly increase load time. However, when a join query does not reference the largest table in a multi-table join index or the difference in row counts between the smaller and larger table is large, an ad hoc join usually outperforms the join index.
- You can improve performance by using an additional column to store frequently calculated results.

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

Impact on query performance of GROUP BY over a UNION ALL

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

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

```
CREATE VIEW vtable (v1 int, v2 char(4)) AS
```

```
SELECT a1, a2 FROM tableA
UNION ALL
SELECT b1, b2 FROM tableB;

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

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

Restrictions on split GROUP BY

There are some restrictions on the situations and queries that benefit from the split GROUP BY.

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

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

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

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

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

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

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

  ```
  CREATE VIEW viewA (va1 int, va2 int, va3 int,
  va4 int) AS
  SELECT b1, b2, b3, b4 FROM tableB
  ```

```
UNION ALL
SELECT c1, c2, c3, c4 FROM tableC
UNION ALL
SELECT d1, d2, d3, d4 FROM tableD
UNION ALL
SELECT e1, e2, e3, e4 FROM tableE
UNION ALL
SELECT f1, f2, f3, f4 FROM tableF
UNION ALL
SELECT g1, g2, g3, g4 FROM tableG;

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

In this example, the Sybase IQ optimizer splits the GROUP BY and inserts six GROUP BY operators into the query plan. Consequently, the query requires more temporary cache to store aggregation information and data. If the system cannot allocate enough cache, the optimizer does not split the GROUP BY.

You can use the TEMP_CACHE_MEMORY_MB database option to increase the size of the temporary cache, if memory is available. For information on setting buffer cache sizes, see “Determining the sizes of the buffer caches” on page 59.

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

Examples of split GROUP BY

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

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

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

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

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

Enhancing ORDER BY query performance

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()
```

Enhanced parallelism within queries

Earlier versions of Sybase IQ had limited ability to utilize many CPUs while running a single query. Sybase IQ now significantly increases the types of operators that are automatically parallelized by the Sybase IQ query engine. This feature is enabled by default and requires no change in query syntax or specific tuning.

Improved subquery performance

Subquery flattening is an optimization technique in which the optimizer rewrites a query containing a subquery into a query that uses a join. Sybase IQ flattens many but not all subqueries. The new database options SUBQUERY_FLATTENING_PREFERENCE and SUBQUERY_FLATTENING_PERCENT control under what circumstances the optimizer chooses to use this optimization.

The FLATTEN_SUBQUERIES option has been deprecated in Sybase IQ 15.0.

Using caching methods

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. Sybase IQ allows users to choose caching methods for executing the correlated subquery using the SUBQUERY_CACHING_PREFERENCE option.

Conditions that cause processing by SQL Anywhere

Sybase IQ uses Component Integration Services (CIS) to query tables on remote servers. CIS allows Sybase IQ to process queries that are not directly supported by Sybase IQ semantics.

CIS processes queries that:

- Reference a user-defined function
- Include certain system functions
- Reference a Catalog Store table

For more information on differences between Sybase IQ and SQL Anywhere, see Appendix A, “Compatibility with Other Sybase Databases,” in *Reference: Building Blocks, Tables, and Procedures*.

Planning queries

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

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

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

Query evaluation options

The following options can help you evaluate the query plan. See *Reference: Statements and Options* for details.

- 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 Chapter 7, "System Procedures," in the *Reference: Building Blocks, Tables, and Procedures.*

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

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

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

- QUERY_DETAIL – When this option and either QUERY_PLAN or QUERY_PLAN_AS_HTML are both ON, Sybase IQ displays additional information about the query when producing its query plan. When QUERY_PLAN and QUERY_PLAN_AS_HTML are OFF, this option is ignored.
- QUERY_PLAN – When set ON (the default), Sybase IQ produces messages about queries. These include messages about using join indexes, about the join order, and about join algorithms for the queries.
- QUERY_PLAN_TEXT_ACCESS – When this option is turned ON, you can view, save, and print IQ query plans from the Interactive SQL client. When QUERY_PLAN_ACCESS_FROM_CLIENT is turned OFF, query plans are not cached, and other query plan-related database options have no affect on the query plan display from the Interactive SQL client. This option is OFF by default.

  See also "GRAPHICAL_PLAN function [String]" and "HTML_PLAN function [String]" in *Reference: Building Blocks, Tables, and Procedures*.
- QUERY_PLAN_AFTER_RUN – When set ON, the query plan is printed after the query has finished running. This allows the plan to include additional information, such as the actual number of rows passed on from each node of the query. In order for this option to work, QUERY_PLAN must be ON. This option is OFF by default.
- QUERY_PLAN_AS_HTML – Produces a graphical query plan in HTML format for viewing in a Web browser. Hyperlinks between nodes make the HTML format much easier to use than the text format in the *.iqmsg* file. Use the QUERY_NAME option to include the query name in the file name for the query plan. This option is OFF by default.
- QUERY_PLAN_AS_HTML_DIRECTORY – When QUERY_PLAN_AS_HTML is ON and a directory is specified with QUERY_PLAN_AS_HTML_DIRECTORY, Sybase IQ writes the HTML query plans in the specified directory.
- QUERY_PLAN_TEXT_CACHING – Gives users a mechanism to control resources for caching plans. With this option OFF (the default), the query plan is not cached for that user connection.

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

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

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

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

The query tree

The optimizer creates a query "tree" that represents the flow of data in the query. The query plan presents the query tree in text form in the *.iqmsg* file, and optionally in graphical form.

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

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

Using query plans

A good way to start using query plans is to set the QUERY_PLAN_AS_HTML option ON. This option places a graphical version of the query plan in the same directory as the *.iqmsg* file. You can view this file in most Web browsers.

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

Authorized users can display query plans in the Java-based Interactive SQL plan window. Users can also save and print query plans from Interactive SQL instead of accessing the *.iqmsg* file or query plan files on the server.

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

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

- Execute the query and open the plan window. Depending on the plan type you selected from the Plan option (Tools > Options > Plan), the appropriate plan displays in the plan window.

  The IQ query plan displays only if the GRAPHICAL_PLAN option is selected. Other plans return the error message, "Plan type is not supported."

- Enter the query in the SQL statements window and select from the menu SQL > Get Plan. Depending on the plan type you selected from the Plan option (Tools > Options > Plan), the appropriate plan displays in the plan window.

  The IQ query plan displays only if the GRAPHICAL_PLAN option is selected. Other plans return the error message, "Plan type is not supported."

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

To access query plans, use the SQL functions, GRAPHICAL_PLAN and HTML_PLAN, for the following queries: SELECT, UPDATE, DELETE, INSERT SELECT, and SELECT INTO.

To save query plans from Interactive SQL, use GRAPHICAL_PLAN or HTML_PLAN to retrieve the query plan and save the output to a file using the OUTPUT statement.

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

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

Controlling query processing

Any user can set limits on the amount of time spent processing a particular query. Users with DBA privileges can give certain users’ queries priority over others, or change processing algorithms to influence the speed of query processing. See *Reference: Statements and Options* for details on the options described in this section.

Setting query time limits

By setting the MAX_QUERY_TIME option, a user can disallow long queries. If a query takes longer to execute than desired, Sybase IQ stops the query with an appropriate error.

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

Setting query priority

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

The following options assign queries a processing priority by user.

- IQGOVERN_PRIORITY – Assigns a numeric priority (1, 2, or 3, with 1 being the highest) to queries waiting in the processing queue.

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

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

Setting query optimization options

The following options affect query processing speed:

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

- INDEX_PREFERENCE – Sets the index to use for query processing. The Sybase IQ optimizer normally chooses the best index available to process local WHERE clause predicates and other operations which can be done within an IQ index. This option is used to override the optimizer choice for testing purposes; under most circumstances it should not be changed.
- JOIN_PREFERENCE – Controls the choice of algorithms when processing joins. This option is designed primarily for internal use; do not use it unless you are an experienced database administrator.
- JOIN_SIMPLIFICATION_THRESHOLD – Controls the minimum number of tables being joined together before any join optimizer simplifications are applied. Normally you should not need to change this value.
- MAX_HASH_ROWS – Sets the maximum estimated number of rows the query optimizer will consider for a hash algorithm. The default is 1,250,000 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.

Setting user-supplied condition hints

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

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

For syntax, parameters, and examples, see "User-supplied condition hints," in Chapter 2, "SQL Language Elements," in *Reference: Building Blocks, Tables, and Procedures*.

Monitoring workloads

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

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

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

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

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

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

See also "INDEX_ADVISOR option" in *Reference: Statements and Options*.

Optimizing delete operations

Sybase IQ chooses the best of three possible algorithms to process delete operations on columns with HG and WD indexes.

HG delete operations

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

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

HG delete costing

Prior to Sybase IQ 12.6, the HG delete cost model considered only worst case I/O performance and therefore preferred large delete in most cases. The current cost model considers many factors including I/O costs, CPU costs, available resources, index metadata, parallelism, and predicates available from the query.

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

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

Using HG delete performance option

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

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

- 1 = Small delete
- 2 = Large delete
- 3 = Mid delete

For more information on the HG_DELETE_METHOD database option, see "HG_DELETE_METHOD option" in Chapter 2, "Database Options" of *Reference: Statements and Options*.

WD delete operations

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

- **Small delete** Small delete for WD 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** 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** Large delete for WD 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 Chapter 2, "Database Options" of *Reference: Statements and Options*.

TEXT delete operations

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

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

TEXT delete costing

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

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

Using TEXT delete performance option

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

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

For more information on the TEXT_DELETE_METHOD database option, see "TEXT_DELETE_METHOD option" in Chapter 2, "TEXT Indexes and Text Configuration Objects" of *Unstructured Data Analytics*.

CHAPTER 4 Managing System Resources

This chapter describes the way Sybase IQ uses memory, disk I/O, and CPUs, and the relationships among these factors. It also explains how the DBA can tune performance by adjusting resource usage.

The suggestions in this chapter are generic. You need to adjust them to suit your hardware and software configuration. Recommendations for each platform are in its *Installation and Configuration Guide*.

Topic	**Page**
Introduction to performance terms	56
Designing for performance	56
Overview of memory use	56
The process threading model	72
Balancing I/O	73
Options for tuning resource use	80
Other ways to improve resource use	84
Indexing tips	86
Managing database size and structure	87
Using UNION ALL views for faster loads	91
Network performance	94

Introduction to performance terms

Performance is the measure of efficiency of a computerized business application, or of multiple applications running in the same environment. It is usually measured in *response time* and *throughput*.

Response time is the time it takes for a single task to complete. It is affected by:

- 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.

Designing for performance

Most gains in performance derive from good database design, thorough query analysis, and appropriate indexing. The largest performance gains can be realized by establishing a good design and by choosing the correct indexing strategy.

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

For more information, see Chapter 3, “Optimizing Queries and Deletions.”

Overview of memory use

Sybase IQ uses memory for several purposes:

- Buffers for data read from disk to resolve queries
- Buffers for data read from disk when loading from flat files
- Overhead for managing connections, transactions, buffers, and database objects

The sections that follow explain how the operating system supports Sybase IQ use of memory, how Sybase IQ allocates memory for various purposes, how you can adjust the memory allocations for better performance, and what you may need to do to configure the operating system so that enough memory is available for Sybase IQ.

Paging increases available memory

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

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

The most frequently used operating system files are *swap files*. When memory is exhausted, the operating system swaps pages of memory to disk to make room for new data. When the pages that were swapped are called again, other pages are swapped, and the required memory pages are brought back. This is very time-consuming for users with high disk usage rates. In general, try to organize memory to avoid swapping and, thus, to minimize use of operating system files. See “Platform-specific memory options” on page 68 for information on configuring memory to minimize swapping.

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

Note Your swap space on disk must be at least large enough to accommodate all of your physical memory.

Utilities to monitor swapping

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

For examples of vmstat output, see “Monitoring paging on UNIX systems.”

Server memory

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

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

Managing memory for multiplexes

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 would 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.

Killing processes affects shared memory

Warning! Killing processes on UNIX systems may result in semaphores or shared memory being left behind instead of being cleaned up automatically. The correct way to shut down a Sybase IQ server on UNIX is the stop_iq utility, described in “Stopping the database server” in Chapter 2, “Running Sybase IQ,” in *System Administration Guide: Volume 1*. For information on using the ipcs and ipcrm to clean up after an abnormal exit, see Chapter 14, “Troubleshooting Hints” in *System Administration Guide: Volume 1*.

Managing buffer caches

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

Read the sections that follow for in-depth information on managing buffer caches:

- For information on how to calculate your memory requirements, see "Determining the sizes of the buffer caches."
- For information on how to set buffer cache sizes once you know what they should be, see "Setting buffer cache sizes."

Determining the sizes of the buffer caches

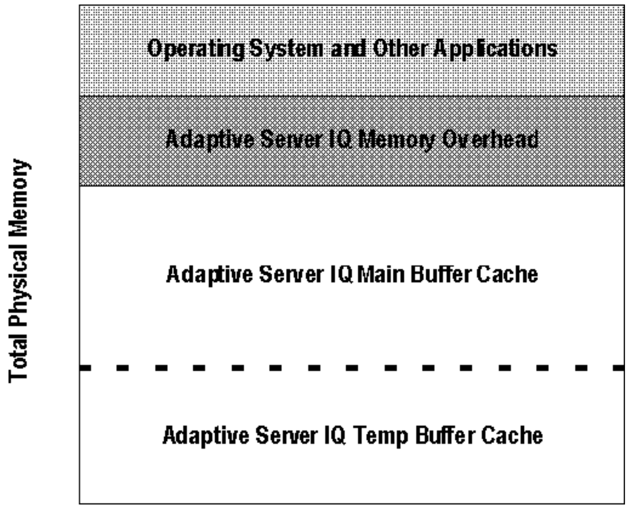
The buffer cache sizes you specify for the IQ store and temporary store will vary based on several factors. The default values (16MB for the main and 12MB for the temporary cache) are too low for most databases. The actual values required for your application depend on:

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

Read the next several sections for guidelines in determining the best settings for your site.

The following diagram shows the relationship between the buffer caches and other memory consumption.

Figure 4-1: Buffer caches in relation to physical memory


Operating System and Other Applications
Adaptive Server IQ Memory Overhead
Adaptive Server IQ Main Buffer Cache
Adaptive Server IQ Temp Buffer Cache
Total Physical Memory

The following sections describe each part in more detail and provide guidelines to help you determine how much memory each part requires.

Operating system and other applications

This amount of memory will vary for different platforms and how the system is used. For example, UNIX file systems do more file buffering than UNIX raw partitions, so the operating system has a higher memory requirement. Most operating systems will use a large percent of available memory for file system buffering. You should understand the buffering policies for your specific operating system to avoid over-allocating memory.

In addition, other applications that run in conjunction with Sybase IQ (such as query tools) have their own memory needs. See your application and operating system documentation for information on their memory requirements.

Sybase IQ memory overhead

After determining how much physical memory the operating system and other applications use, you can calculate how much of the remaining memory Sybase IQ requires to do its tasks. The factors that affect this overhead are described in the following sections.

Raw partitions versus file systems

For UNIX systems, databases using file systems rather than raw partitions may require another 30% of the remaining memory to handle file buffering by the operating system. On Windows, file system caching should be disabled by setting OS_FILE_CACHE_BUFFERING = 'OFF' (the default for new databases). For more information, see the *Installation and Configuration Guide* for your platform.

Multiuser database access

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

Memory for thread stacks

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

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

The total number of threads (-iqmt plus -gn) must not exceed the number allowed for your platform. For details, see Chapter 1, “Running the Database Server,” in the *Utility Guide*.

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. See "Increasing memory used during backup" in Chapter 12, "Data Backup, Recovery, and Archiving," in *System Administration Guide: Volume 1*.

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

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

Sybase IQ main and temp buffer caches

After determining how much overhead memory Sybase IQ needs, you must decide how to split what is left between your main Sybase IQ and temp buffer caches. The dashed line dividing the two areas in Figure 4-1 indicates that this split may change from one database to another based on several factors.

Unlike most other databases, the general guideline for Sybase IQ is a split of 40% for the main buffer cache and 60% for temp buffer cache. This guideline, however, is only a start. While some operations, such as queries with large sort-merge joins or inserts involving HG indexes, may require a temp buffer cache larger than main, other applications might have different needs.

Note These guidelines assume you have one active database on your system at a time (that is, any Sybase IQ users are accessing only one database). If you have more than one active database, you need to further split the remaining memory among the databases you expect to use.

Sybase strongly recommends that you start with the general guidelines presented here and watch the performance of Sybase IQ by using its monitor tool (described in “Monitoring the buffer caches” on page 115) and any specific tools described in the *Installation and Configuration Guide* for your platform.

Buffer caches and physical memory

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

In almost every case, the default temporary buffer cache size of 8MB is too low. For optimal performance, allocate as much memory as possible to the IQ main and temporary buffer caches. For example, if you have 4GB of physical memory on your machine available to Sybase IQ, you can split that amount between the main and temporary shared buffer caches.

Note On some UNIX platforms, you may need to set other server switches to make more memory available for buffer caches. See “Platform-specific memory options” on page 68 for more information.

Other considerations

Sybase IQ buffer cache sizes may differ from one database to the next based on use. For maximum performance, you need to change the settings between inserting, querying the database, and mixed use. In a mixed-use environment, however, it is not always feasible to require all users to exit the database so that you can reset buffer cache options. In those cases, you may need to favor either load or query performance.

The buffer cache and memory overhead guidelines also may differ between platforms. See your *Installation and Configuration Guide* for any other issues.

Setting buffer cache sizes

By default, Sybase IQ sets the size of the main and temporary buffer caches to 16MB and 12MB respectively. Most applications will require much higher values (limited by the total amount of physical memory). See the preceding sections to determine the right settings for your system.

Once you know what settings you need, use the options described in Table 4-1 to set buffer cache sizes.

Table 4-1: Settings that change buffer cache sizes

Method	When to use it	How long the setting is effective	For more information, see
-iqmc and -iqtc server switches	Recommended method. Sets cache sizes when the database and server are not running. Allows cache sizes >4GB. Especially useful for 64-bit platforms, or if cache size database options are set larger than your system can accommodate.	From the time the server is started until it is stopped You must restart the server to change buffer cache sizes. The -iqmc and -iqtc server start-up options only remain in effect while the server is running, so you need to include them every time you restart the server.	"Starting the database server" in Chapter 1, "Running the Database Server," in the *Utility Guide*.

Specifying page size

When you create a database, you set the Sybase IQ page size. This parameter, in conjunction with the size of the buffer cache, affects memory use and disk I/O throughput for that database.

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

Setting the page size

Sybase IQ swaps data in and out of memory in units of **pages**. When you create a database, you specify a separate page size for the catalog store and the IQ store. The temporary store has the same page size as the IQ store.

For Sybase IQ page size recommendations for the best performance, see "Choosing an IQ page size" in Chapter 5, "Working with Database Objects," *System Administration Guide: Volume 1*.

Because the catalog store accounts for only a tiny fraction of I/O, the 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. These factors are discussed in the sections that follow.

Block size

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

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

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

Table 4-2 shows the default block size for each IQ page size.

Table 4-2: Default block sizes

IQ page size (KB)	*Default block size (bytes)*
64	4096
128 (default for new databases)	8192
256	16384
512	32768

Data compression

Sybase IQ compresses all data when storing it on disk. Data compression both reduces disk space requirements and contributes to performance. The amount of compression is determined automatically, based on the IQ page size.

Saving memory

If your machine does not have enough memory, to save memory you can try the following adjustments.

Decrease buffer cache settings

You may be able to save memory by decreasing buffer cache sizes. Keep in mind that if you decrease the buffer caches too much, you could make your data loads or queries inefficient or incomplete due to insufficient buffers.

Optimizing for large numbers of users

Sybase IQ handles up to 200 user connections on 32-bit platforms and up to 1000 user connections on 64-bit platforms. To support the maximum number of users on 64-bit systems, you may need to adjust both operating system parameters and start_iq server parameters. For recommendations, see the *Installation and Configuration Guide* as well as the sections that follow.

Sybase IQ command line option changes for large numbers of users

The following start_iq switches affect operations with large numbers of users:

-gm *#_connections_to_support*

-iqgovern *#_ ACTIVE_ queries_to_support*

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

-c *catalog_store_cache_size*

-ch *size*

-cl *size*

-gm

This is the total number of connections the server will support. Statistically, some of these are expected to be connected and idle while others are connected and actively using the database.

-iqgovern

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

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

-gn

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

The catalog store buffer cache is also the general memory pool for the catalog store. To specify in MB, use the form -c nM, for example, -c 64M. Sybase recommends these values:

Table 4-3: Catalog buffer cache settings

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

In some cases the standard Catalog cache size may be too small, for example, to accommodate certain queries that need a lot of parsing. In these cases, you may find it helpful to set -cl and -ch. For example, on 32-bit platforms, try these settings

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

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

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

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

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

-iqmt	If -iqmt is set too low for the -gm setting, then thread starvation can occur. See “Starting the database server” in Chapter 1, “Running the Database Server,” in the *Utility Guide*.

Increasing Sybase IQ temporary space for large numbers of users

You may need to increase your temporary dbspace to accommodate more users.

Relative priorities of new and existing connections

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

Platform-specific memory options

On all platforms, Sybase IQ uses memory for four primary purposes:

- Main buffer cache
- Temporary buffer cache
- Sybase IQ memory overhead (including thread stacks)
- Load buffers

See Figure 4-1 on page 60 for a diagram of Sybase IQ memory use.

On all 64-bit platforms, the total amount of usable memory is effectively unlimited. The only limit is the virtual memory of the system.

For performance tuning hints on HP-UX systems, see the *Installation and Configuration Guide* for that platform.

On 32-bit platforms restrictions apply; see the following table for details.

Table 4-4: Total available memory on 32-bit platforms

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

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

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

For more performance tuning hints on Windows platforms, see Chapter 6, "Tuning Servers on 32-bit Windows Systems."

For UNIX systems only, Sybase IQ provides two command-line options that can help you manage memory.

Wired memory pool

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

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

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

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

```
    -iqwmem 10000
```

Warning! Use this switch only if you have enough memory to dedicate the amount you specify for this purpose. Otherwise, you can cause serious performance degradation.

Note For this version:

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

Impact of other applications and databases

Remember, the memory used for the server 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

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

Controlling file system buffering

On Sun Solaris UFS, Linux, Linux IBM, AIX, and Windows file systems only, you can control whether file system buffering is turned on or off. Turning off file system buffering saves a data copy from the file system buffer cache to the main IQ buffer cache. Usually, doing so reduces paging, and therefore improves performance. Be aware of one exception: If the IQ page size for the database is less than the block size of the file system (typically only in testing situations), turning off file system buffering may decrease performance, especially during multiuser operation.

File system buffering is turned off by default for newly created Sybase IQ databases.

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
```

Experiment with this option to determine the best setting for different conditions. You can only set this option for the PUBLIC group. You must shut down the database and restart it for the change to take effect.

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

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

Notes

- Sybase IQ does not support direct I/O on Linux kernel version 2.4. If you set the IQ_USE_DIRECTIO environment variable on Linux kernel version 2.4, the Sybase IQ server does not start. The error "`Error: Invalid Block I/O argument, maybe <`***`pathname`***`> is a directory, or it exceeds maximum file size limit for the platform, or trying to use Direct IO on unsupported OS`" is reported.
- Solaris does not have a kernel parameter to constrain the size of its file system buffer cache. Over time, the file system buffer cache grows and displaces the IQ buffer cache pages, leading to excess operating system paging activity and reduced Sybase IQ performance.
- Windows can bias the paging algorithms to favor applications at the expense of the file system. This bias is recommended for Sybase IQ performance. See Chapter 6, "Tuning Servers on 32-bit Windows Systems" for details.

See also "OS_FILE_CACHE_BUFFERING option" in Chapter 2, "Database Options" of *Reference: Statements and Options*.

Other ways to get more memory

In certain environments, you may be able to adjust other options to make more memory available to Sybase IQ.

Options for Java-enabled databases

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

The process threading model

Sybase IQ uses operating system kernel threads for best performance. Threads can be found at the user level and at the kernel level. Lightweight processes are underlying threads of control that are supported by the kernel. The operating system decides which lightweight processes (LWPs) should run on which processor and when. It has no knowledge about what the user threads are, but does know if they are waiting or able to run.

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

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

Insufficient threads error

When you do not have enough server threads to initiate the query you have issued, you get the 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 you issue it. If the condition persists, you may need to restart the server and specify more Sybase IQ threads, as described in the next section. It is also possible that -iqmt is set too low for the number of connections.

Sybase IQ options for managing thread usage

Sybase IQ offers the following options to help you manage thread usage.

- To set the maximum number of threads available for Sybase IQ use, set the server start-up option -iqmt. The default value is calculated from the number of connections and the number of CPUs and is usually adequate. See "Starting the database server" in Chapter 1, "Running the Database Server," in the *Utility Guide*.
- To set the stack size of the internal execution threads in the server, set the server start-up option -iqtss. 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. See "Starting the database server" in Chapter 1, "Running the Database Server," in the *Utility Guide*.
- To set the maximum number of threads a single user will use, issue the command SET OPTION MAX_IQ_THREADS_PER_CONNECTION. Some operations try to allocate and use a "team" of threads. To set then number of threads available to a "team", issue the command SET OPTION MAX_IQ_THREADS_PER_TEAM. These options can be used to control the amount of resources a particular operation consumes. For example, the DBA can set this option before issuing an INSERT ,LOAD , BACKUP, or RESTORE command.

Balancing I/O

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

Raw I/O (on UNIX operating systems)

Most UNIX file systems divide disks into fixed size partitions. *Partitions* are physical subsets of the disk that are accessed separately by the operating system. Disk partitions are typically accessed in two modes: file system mode (through the UFS file system) or raw mode. Raw mode (sometimes called character mode) does unbuffered I/O, generally making a data transfer to or from the device with every read or write system call. The UFS mode is a UNIX file system and a buffered I/O system which collects data in a buffer until it can transfer an entire buffer at a time.

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

For more information, see "Working with database objects" in Chapter 5, "Working with Database Objects" of the *System Administration Guide: Volume 1*.

Using disk striping

Disk striping is a generic method of spreading data from a single file across multiple disk drives. This method allows successive disk blocks to be located on striped disk drives. Striping combines one or more physical disks (or disk partitions) into a single logical disk. Striped disks split I/O transfers across the component physical devices, performing them in parallel. They achieve significant performance gains over single disks.

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

You can use disk striping either as provided by your operating system and hardware, or Sybase IQ internal disk striping.

Setting up disk striping on UNIX

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

Setting up disk striping on Windows

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

Recommendations for disk striping

Here are some general rules on disk striping:

- For maximum performance, the individual disks in a striped file system should be spread out across several disk controllers. But be careful not to saturate a disk controller with too many disks. Typically, most SCSI machines can handle 2–3 disks per controller. See your hardware documentation for more information.
- Do not put disks on the same controller as slower devices, such as tape drives or CD-ROMs. This slows down the disk controller.
- Allocate 4 disks per server CPU in the stripe.
- The individual disks must be identical devices. This means they must be the same size, have the same format, and often be the same brand. If the layouts differ, the size of the smallest one is often used and other disk space is wasted. Also, the speed of the slowest disk is often used.
- In general, disks used for file striping should not be used for any other purpose. For example, do not use a file striped disk as a swap partition.
- Never use the disk containing the root file system as part of a striped device.
- Use raw partitions for maximum performance.

In general, you should use disk striping whenever possible.

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

Internal striping

Sybase IQ stores its information in a series of dbspaces—files or raw partitions of a device—in blocks. Assuming that disk striping is in use, Sybase IQ spreads data across all dbspaces that have space available. This approach lets you take advantage of multiple disk spindles at once, and provides the speed of parallel disk writes.

Disk striping option

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

Turning disk striping on or off

The syntax you use to change the default striping when creating a dbspace is:

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

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

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

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

Using multiple files

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

You can add additional files to a dbspace with the ALTER DBSPACE command.

When to add files

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

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

Strategic file locations

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

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

To avoid disk bottlenecks, follow these suggestions:

- Keep random disk I/O away from sequential disk I/O.
- Isolate Sybase IQ database I/O from I/O for proxy tables in other databases, such as Adaptive Server Enterprise.
- Place the transaction log and message log on separate disks from the IQ store, catalog store, and temporary store, and from any proxy databases such Adaptive Server Enterprise.
- Place the database file, temporary dbspace, and transaction log file on the same physical machine as the database server.

The transaction log file

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

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

Warning! The Sybase IQ transaction log file is different from most relational database transaction log files. If for some reason you lose your database files, then you lose your database (unless it is the log file that is lost). However, if you have an appropriate backup, then you can reload the database.

Truncating the transaction log

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

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

Table 4-5 shows methods for truncating transaction logs in Sybase IQ.

Table 4-5: Truncating transaction logs

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

Truncating the transaction log of a stopped database

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

❖ **Truncating the transaction log of a stopped database**

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

The message log

A message log file exists for each database. The default name of this file is *dbname.iqmsg*, although you can specify a different name when you create the database. The message log file is actually created when the database is started for the first time after creation of that database.

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

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

For information on setting the maximum log file size, archiving message log files, and enabling message log wrapping, see "Message logging" in Chapter 1, "Overview of Sybase IQ System Administration" of the *System Administration Guide: Volume 1*.

Working space for inserting, deleting, and synchronizing

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

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

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

Setting reserved space options

Two database options, MAIN_RESERVED_DBSPACE_MB and TEMP_RESERVED_DBSPACE_MB, control the amount of space Sybase IQ reserves for certain operations. For more information see "IQ main store and IQ temporary store space management" in Chapter 5, "Working with Database Objects" of the *System Administration Guide: Volume 1*.

Options for tuning resource use

The number of concurrent users of a Sybase IQ database, the queries they run, and the processing threads and memory available to them, can have a dramatic impact on performance, memory use, and disk I/O. Sybase IQ provides several options for adjusting resource use to accommodate varying numbers of users and types of queries. These may be:

- SET OPTION command options that affect only the current database.
- Command-line options that affect an entire database server.
- Connection parameters that affect the current connection only.

For more information on all of these options, including parameters, when the options take effect, and whether you can set them for both a single connection and the PUBLIC group, see the *Reference: Statements and Options*.

For information specific to optimizing tables, see "Optimizing storage and query performance," in Chapter 5, "Working with Database Objects" of the *System Administration Guide: Volume 1*.

Restricting concurrent queries

The -iqgovern switch lets you specify the number of concurrent queries on a particular server. This is not the same as the number of connections, which is controlled by your license. By specifying the -iqgovern switch, you can help IQ optimize paging of buffer data out to disk, and avoid over committing memory. The default value of -iqgovern is (2 x the number of CPUs) + 10. You may need to experiment to find an ideal value. For sites with large numbers of active connections, try setting -iqgovern slightly lower.

Setting the number of CPUS available

The -iqnumbercpus switch on the Sybase IQ start-up command lets you specify the number of CPUs available. This switch is recommended only:

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

For details, see "Setting the number of CPUs," Chapter 2, "Running Sybase IQ," in the *System Administration Guide: Volume 1*.

Limiting temporary dbspace use by a query

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.

When you issue a query, Sybase IQ estimates the temporary space needed to resolve the query. If the total estimated temporary result space for sorts, hashes, and row stores exceeds the current QUERY_TEMP_SPACE_LIMIT setting, the query is rejected, and you receive a message such as:

```
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, the DBA can 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.

Limiting queries by rows returned

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

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

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

Forcing cursors to be non-scrolling

When you use scrolling cursors with no host variable declared, Sybase IQ creates a temporary store node where query results are buffered. This storage is separate from the temporary store buffer cache. The temporary store node enables efficient forward and backward scrolling when your application searches through a result set. However, if the query returns very large numbers (such as millions) of rows of output, and if your application performs mostly forward-scrolling operations, the memory requirements of the temporary store node may degrade query performance. To improve performance, eliminate the temporary store node by issuing the following command:

SET TEMPORARY OPTION FORCE_NO_SCROLL_CURSORS = ‘ON’

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

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

Limiting the number of cursors

The MAX_CURSOR_COUNT option specifies a resource governor to limit 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.

Limiting the number of statements

The MAX_STATEMENT_COUNT option specifies a resource governor to limit the maximum number of prepared statements that a connection can use at once.

Prefetching cache pages

The SET option PREFETCH_BUFFER_LIMIT defines the number of cache pages available to Sybase IQ for use in prefetching (the read ahead of database pages). This option has a default value of 0. Set this option only if advised to do so by Sybase Technical Support. For more information, see “PREFETCH_BUFFER_LIMIT option” in *Reference: Statements and Options*.

The SET option BT_PREFETCH_MAX_MISS 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. For more information, see “BT_PREFETCH_MAX_MISS option” in *Reference: Statements and Options*.

Optimizing for typical usage

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

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

Controlling the number of prefetched rows

Prefetching is used to improve 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.

Other ways to improve resource use

This section describes several ways to adjust your system for maximum performance or better use of disk space.

Managing disk space in multiplex databases

Sybase IQ cannot drop old versions of tables while any user on any server might be in a transaction that might need the old versions. Sybase IQ may therefore consume a very large amount of disk space when table updates and queries occur simultaneously in a multiplex database. The amount of space consumed depends on the nature of the data and indexes and the update rate.

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

Load balancing among query servers

You may be able to use the IQ Network Client to balance the query load among multiplex query servers. This method requires an intermediate system that is able to dispatch the client connection to a machine in a pool, depending on the workload of the machine.

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

For details on how to define an ODBC DSN for use in query server load balancing, see "VerifyServerName parameter [Verify]" in Chapter 4, "Connection and Communication Parameters" in the *System Administration Guide: Volume 1*.

Restricting database access

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

Disk caching

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

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

Indexing tips

The following sections give some tips for selecting and managing indexes. See Chapter 6, "Using Sybase IQ Indexes," in the *System Administration Guide: Volume 1* for more information on these topics.

Choosing the right index type

Choose the correct index type for your column data. 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.

The Sybase IQ query optimizer features an index advisor that generates messages when the optimizer would benefit from an additional index on one or more columns in your query. To activate the index advisor, set the INDEX_ADVISOR option ON. Messages print as part of a query plan or as a separate message in the message log (*.iqmsg*) if query plans are not enabled, and output is in OWNER.TABLE.COLUMN format. For details, see "INDEX_ADVISOR option," in Chapter 2, "Database Options," in *Reference: Statements and Options*.

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

```
SELECT c.name, SUM(l.price * (1 - l.discount))
```

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

Note that adding indexes increases storage requirements and load time and should only be done if there is a net benefit to query performance.

Using join indexes

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

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

Allowing enough disk space for deletions

When you delete data rows, Sybase IQ creates a version page for each database page that contains any of the data being deleted. The versions are retained until the delete transaction commits. For this reason, you may need to add disk space when you delete data. See “Overlapping versions and deletions” on page 394 for details.

Managing database size and structure

This section offers ideas on improving your database design and managing your data.

Managing the size of your database

The size of your database depends largely on the indexes you create, and the quantity of data you maintain. You achieve faster query processing by creating all of the indexes you need for the types of queries your users issue. However, if you find that some tables or indexes are not needed, you can drop them. By doing so, you free up disk space, increase the speed of loads and backups, and reduce the amount of archive storage you need for backups.

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

Controlling index fragmentation

Internal index fragmentation occurs when index pages are not being used to their maximum volume.

Row fragmentation can occur when rows are deleted. If you delete an entire page of rows, that page is freed, but if some rows on a page are unused, unused space remains on the disk.

DML operations (INSERT, UPDATE, DELETE) that act on tables cause index fragmentation. Two stored procedures report fragmentation:

- sp_iqrowdensity reports row fragmentation at the default index level. See "sp_iqrowdensity procedure," in Chapter 7, "System Procedures," in *Reference: Building Blocks, Tables, and Procedures*.
- sp_iqindexfragmentation reports internal fragmentation within supplemental indexes. See "sp_iqindexfragmentation procedure," in Chapter 7, "System Procedures," in *Reference: Building Blocks, Tables, and Procedures*.

The database administrator may create other indexes to supplement the default index on a column. These indexes can use more space than needed when rows are deleted from a table.

Neither procedure recommends further action. The database administrator must examine the information reported and determine whether to take further action, such as recreating, reorganizing, or rebuilding indexes.

Minimizing catalog file growth

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

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

Denormalizing for performance

Once you have created your database in normalized form, you may perform benchmarks and decide to intentionally back away from normalization to improve performance. Denormalizing:

- Can be done with tables or columns
- Assumes prior normalization
- Requires a knowledge of how the data is being used

Good reasons to denormalize are:

- All queries require access to the “full” set of joined data
- Computational complexity of derived columns require storage for selects

Denormalization has risks

Denormalization can be successfully performed only with thorough knowledge of the application and should be performed only if performance issues indicate that it is needed. One of the things to consider when you denormalize is the amount of effort it will then take to keep your data up-to-date with changes.

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.

Whatever form of denormalization you choose, it has the potential for data integrity problems which must be carefully documented and addressed in application design.

Disadvantages of denormalization

Denormalization has these disadvantages:

- Denormalization usually speeds retrieval but can slow updates. This is not a real concern in a DSS environment.
- Denormalization is always application-specific and needs to be re-evaluated if the application changes.
- Denormalization can increase the size of tables. This is not a problem in Sybase IQ, because you can optimize the storage of column data. For details, see the IQ UNIQUE column constraint in CREATE TABLE statement and "MAX_QUERY_TIME option" in *Reference: Statements and Options*.
- In some instances, denormalization simplifies coding; in others, it makes it more complex.

Performance benefits of denormalization

Denormalization can improve performance by:

- Minimizing the need for joins
- Precomputing aggregate values, that is, computing them at data modification time, rather than at select time
- Reducing the number of tables, in some cases

Deciding to denormalize

When deciding whether to denormalize, you need to analyze the data access requirements of the applications in your environment and their actual performance characteristics. Some of the issues to examine when considering denormalization include:

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

- Do any processes compute summaries?
- Should you create join indexes to gain performance?

Using UNION ALL views for faster loads

UNION ALL views can be used to improve load performance where it is too expensive to maintain secondary indexes for all rows in a table. Sybase IQ lets you split the data into several separate base tables (for example, by date). You load data into these smaller tables. You then join the tables back together into a logical whole by means of a UNION ALL view, which you can then query.

This strategy can improve load performance, but may negatively impact the performance of some types of queries. Most types of queries have roughly similar performance against a single base table or against a UNION ALL view over smaller base tables, as long as the view definition satisfies all the constraints described in "Optimizing queries that reference UNION ALL views" on page 92. However, some types of queries, especially those involving DISTINCT or involving joins with multiple join columns, may perform significantly slower against a UNION ALL view than against a single large base table. Before choosing to use this strategy, determine whether the improvements in load performance are worth the degradation in query performance for your application.

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

To create a UNION ALL view, choose a logical means of dividing a base table into separate physical tables. The most common division is by month.

For example, to create a view including all months for the first quarter, enter:

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

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

For syntax details, see UNION operation in the *Reference: Statements and Options*.

Note You cannot perform an INSERT...SELECT into a UNION ALL view. UNION ALL operators are not fully parallel in this release. Their use may limit query parallelism.

Optimizing queries that reference UNION ALL views

All partitions in a UNION ALL view must have a complete set of indexes defined for optimization to work.

Queries with DISTINCT will tend to run more slowly using a UNION ALL view than a base table.

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

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

Should you need to adjust performance for queries that reference UNION ALL views, you might want to set the JOIN_PREFERENCE database option, which affects joins between UNION ALL views. For details of these options, see Chapter 2, “Database Options,” in *Reference: Statements and Options.*

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 "SELECT statement" in *Reference: Statements and Options*.

Managing UNION ALL view performance

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 "SELECT statement" in *Reference: Statements and Options*.

Improved loading for large single (fact) tables

In order to meet the challenges of the exponential growth of information and the demand for real-time access to data, Sybase IQ has significantly enhanced the performance of loading High_Group (HG) indexes and Containment (also called WORD) (WD) indexes, while transactions are still allowed to access the tables being loaded. This load performance improvement for HG and WD indexes affects:

- INSERT...SELECT
- INSERT...LOCATION
- LOAD
- UPDATE
- CREATE INDEX

- Updatable cursors

The LOAD TABLE statement now performs parallel loads of HG and WD indexes, thus executing faster than in previous releases.

Network performance

The following sections offer suggestions for solving some network performance issues.

Improving large data transfers

Large data transfers simultaneously decrease overall throughput and increase the average response time. Here are some suggestions to improve performance during these transfers:

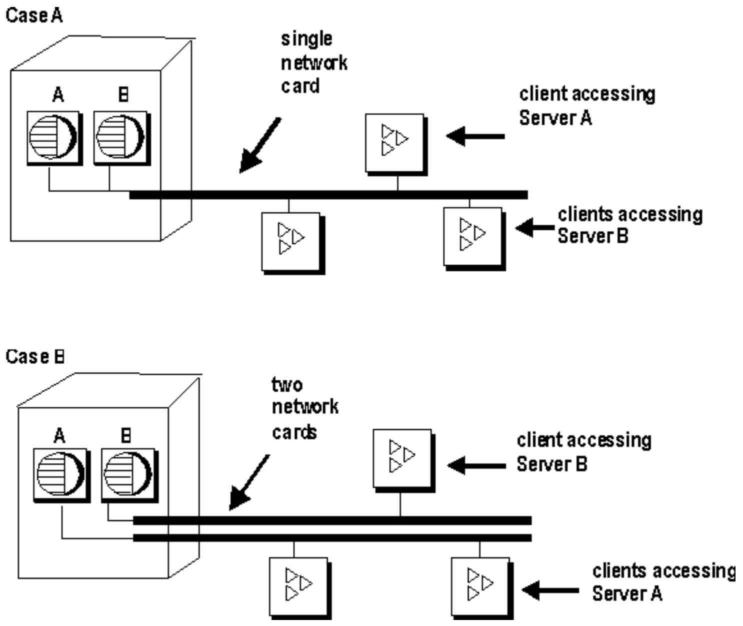
- Perform large transfers during off-hour periods, if possible.
- Limit the number of concurrent queries during large transfers.
- Do not run queries and insertions concurrently during large transfers.
- Use stored procedures to reduce total traffic.
- Use row buffering to move large batches through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

Isolate heavy network users

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

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

Figure 4-2: Isolating heavy network users


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

Put small amounts of data in small packets

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

Put large amounts of data in large packets

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

Process at the server level

Filter as much data as possible at the server level.

CHAPTER 5

Monitoring and Tuning Performance

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

Topic	Page
Viewing the Sybase IQ environment	98
Monitoring the buffer caches	115
Buffer cache structure	127
Avoiding buffer manager thrashing	129
Buffer cache monitor checklist	133
System utilities to monitor CPU use	136

Viewing the Sybase IQ environment

The first step in tuning Sybase IQ performance is to look at your environment. You have various options:

- Use system monitoring tools (each system and site has different tools in place).
- Use the dynamic performance monitor in Sybase Central. See “Monitoring performance statistics” on page 98. For multiplex performance monitoring, see *Using Sybase IQ Multiplex*.
- Use one of the stored procedures that displays information about Sybase IQ. See “Getting information using stored procedures” on page 106.
- Use procedure profiling to track execution times for stored procedures, functions, and events. See “Profiling database procedures” on page 107.
- Determine appropriateness of index types. See Chapter 6, “Using Sybase IQ Indexes” in *System Administration Guide: Volume 1* for more information about choosing index types.
- For on-screen information, look at your insert and delete notification messages. Chapter 7, “Moving Data In and Out of Databases” in *System Administration Guide: Volume 1* gives more information about these messages.
- Look at the Sybase IQ message file, called *dbname.iqmsg* by default.

Monitoring performance statistics

The Performance Monitor in Sybase Central displays a collection of statistics for one or more participating nodes. Statistics display in a dynamic chart in real time.

The performance monitor can be accessed at two different levels:

- Multiplex level – You can monitor only one statistic, across multiple servers.
- Server level – On a single server or a multiplex server, you can monitor up to ten statistics at a time.

This section describes server level access on a single server only. See *Using Sybase IQ Multiplex* to use multiplex level and server level for multiplex servers.

❖ **Monitoring performance at the server level**

- Click the server name in the Sybase Central tree view, and switch to the Performance Monitor tab.

Customizing statistics display

You can change the type or contents of the performance monitor graphs.

❖ **Changing statistics to monitor**

When you launch this dialog from the server-level performance monitor, you can select up to ten statistics to monitor at a time. (This dialog behaves differently for multiplex-level monitoring. See *Using Sybase IQ Multiplex*.)

1. Click the server and switch to the Performance Monitor tab.
2. In the Performance Monitor tab, right-click the chart area, choose Change Statistics from the shortcut menu.
3. In the Change Statistics dialog, choose the statistics you want to monitor.

❖ **Saving the chart as a bitmap**

You can save the chart as a .JPEG image file.

1. Right-click the Performance Monitor chart.
2. From the popup menu, select Save Chart As.
3. Specify a file name for the .JPEG file.
4. Click Save.

❖ **Printing the chart**

You can print the performance monitor chart.

1. Right-click the Performance Monitor chart.
2. From the popup menu, select Print Chart.
3. Review your print options and print the page to the desired printer.

❖ **Switching chart view**

You can choose between time-series, 2-D bar, and 3-D bar chart view.

1. Right-click the Performance Monitor chart.
2. From the menu, select Switch Chart View.

3. Select either Time Series Chart, Bar Chart 2-D Vertical, or Bar Chart 3-D Vertical.

❖ Customizing the chart

Change the chart settings and the chart refresh rate, also known as the monitor GUI heartbeat rate.

1. Right-click the performance monitor chart area.
2. Select Customize Chart.

   The Customize Chart dialog has these components:

   - Time Window – Appears only if the chart is a Time Series chart. Specify the period of time during which the data is displayed. The minimum value is 1 minute. The maximum value is 240 minutes (4 hours).
   - Chart Refresh Rate – Specify how often the data in the chart is refreshed, in seconds. The value also shows at the bottom of the performance monitor pane.
   - Real vs. Normalized Value – Real Value (the default) reflects actual data. Normalized Value scales chart data into a fixed range. This option is for display purposes only, so that statistics with different ranges display better in one chart.
   - Legend – Select to display or hide the legend.
3. To save any changes, click OK.

Categories of statistics

Statistics are grouped into the following categories:

- CPU usage statistics
- Memory usage statistics
- Cache statistics
- Thread statistics
- Connection statistics
- Request statistics
- Transaction statistics
- Store I/O statistics

- DBspace usage
- Network statistics

CPU usage statistics

Table 5-1: CPU Usage

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

Memory usage statistics

Table 5-2: Memory Usage

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

Cache statistics

Table 5-3: Cache Statistics

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

Name	Description	Monitored By Default?
Catalog Cache Dirty Pages Percentage	Percentage of catalog cache dirty pages.	No
Temporary Cache Dirty Pages Percentage	Percentage of temporary cache dirty pages.	No
Main Cache Dirty Pages Percentage	Percentage of main cache dirty pages.	No

Thread statistics

Table 5-4: Thread Statistics

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

Connection statistics

Table 5-5: Connection Statistics

Name	Description	Monitored By Default?
Total Connections	Total number of connections including user and INC connections.	Yes
User Connections	Number of user connections.	No
INC Incoming Connections	Number of INC incoming connections	No
INC Outgoing Connections	Number of INC outgoing connections	No
User Connections Per Minute	Number of user connections per minute	No
User Disconnections Per Minute	Number of user disconnections per minute	No

Request statistics

Table 5-6: Request Statistics

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

Transaction statistics

Table 5-7: Transaction Statistics

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

Store I/O statistics

Table 5-8: Store I/O Statistics

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

DBspace usage

Table 5-9: DBSpace Usage

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

Network statistics

Table 5-10: Network Statistics

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

Getting information using stored procedures

Sybase IQ offers several stored procedures that display information about your database:

- sp_iqconnection displays statistics about user connections and versions
- sp_iqcontext displays information about what statements are executing
- sp_iqcheckdb checks the validity of your current database
- sp_iqdbstatistics reports results of the most recent sp_iqcheckdb

- sp_iqdbsize gives the size of the current database
- sp_iqspaceinfo displays space usage by each object in the database
- sp_iqstatus displays miscellaneous status information about the database.
- sp_iqtablesize gives the size of the table you specify.
- sp_iqgroupsize lists the members of the specified group.

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

Profiling database procedures

Procedure profiling shows you how long it takes your stored procedures, functions, events, system triggers, and triggers to execute. You can also view the execution time for each line of a procedure. Using the database profiling information, you can determine which procedures can be fine-tuned to increase performance within your database.

When profiling is enabled, Sybase IQ monitors which stored procedures, functions, events, system triggers, and triggers are used, keeping track of how long it takes to execute them, and how many times each one is called.

Profiling information is stored in memory by the server and can be viewed in Sybase Central via the Profile tab or in Interactive SQL. Once profiling is enabled, the database gathers profiling information until you disable profiling or until the server is shut down.

For more information about obtaining profiling information in Interactive SQL, see "Viewing procedure profiling information in Interactive SQL" on page 113.

Enabling procedure profiling

Procedure profiling tracks the usage of procedures and triggers by all connections. You can enable profiling in either Sybase Central or Interactive SQL. You must have DBA authority to enable and use procedure profiling.

❖ **To enable profiling (Sybase Central)**

1. Connect to your database as a user with DBA authority.
2. Select the database in the left pane.
3. From the File menu, choose Properties.

The Database property sheet appears.

4. On the Profiling tab, select Enable Profiling on This Database.
5. Click OK to close the property sheet.

Note

You can also right click your database in Sybase Central to enable profiling. From the popup menu, choose Profiling > Start Profiling.

❖ **To enable profiling (SQL)**

1. Connect to your database as a user with DBA authority.
2. Call the sa_server_option stored procedure with the ON setting.

   For example, enter:

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

If necessary, you can see what procedures a specific user is using, without preventing other connections from using the database. This is useful if the connection already exists, or if multiple users connect with the same userid.

❖ **To filter procedure profiling by user**

1. Connect to the database as a user with DBA authority.
2. Call the following procedure:

   ```
   CALL sa_server_option
   ('ProfileFilterUser','userid')
   ```

   The value of *userid* is the name of the user being monitored.

Resetting procedure profiling

When you reset profiling, the database clears the old information and immediately starts collecting new information about procedures, functions, events, and triggers.

The following sections assume that you are already connected to your database as a user with DBA authority and that procedure profiling is enabled.

❖ **To reset profiling (Sybase Central)**

1. Select the database in the left pane.
2. From the File menu, choose Properties.

   The Database property sheet appears.

3. On the Profiling tab, click Reset Now.

4 Click OK to close the property sheet.

Note

You can also right click your database in Sybase Central to reset profiling. From the popup menu, click Profiling > Reset Profiling Information.

❖ **To reset profiling (SQL)**

- Call the sa_server_option stored procedure with the RESET setting.

  For example, enter:

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

Disabling procedure profiling

Once you are finished with the profiling information, you can either disable profiling or you can clear profiling. If you disable profiling, the database stops collecting profiling information and the information that it has collected to that point remains on the Profile tab in Sybase Central. If you clear profiling, the database turns profiling off and removes all the profiling data from the Profile tab in Sybase Central.

❖ **To disable profiling (Sybase Central)**

1 Select the database in the left pane.

2 From the File menu, choose Properties.

  The Database property sheet appears.

3 On the Profiling tab, clear the Enable Profiling on This Database option.

4 Click OK to close the property sheet.

Note

You can also right click your database in Sybase Central to disable profiling. From the popup menu, choose Profiling > Stop Profiling.

❖ **To disable profiling (SQL)**

- Call the sa_server_option stored procedure with the OFF setting.

  For example, enter:

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

❖ **To clear profiling (Sybase Central)**

1 Select the database in the left pane.

2 From the File menu, choose Properties.

The Database property sheet appears.

3 On the Profiling tab, click Clear Now.

You can only clear profiling if profiling is enabled.

4 Click OK to close the property sheet.

Note

You can also right click your database in Sybase Central to clear profiling. From the popup menu, select Profiling > Clear Profiling Information.

❖ **To clear profiling (SQL)**

- Call the sa_server_option stored procedure with the CLEAR setting.

  For example, enter:

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

Viewing procedure profiling information in Sybase Central

Procedure profiling provides you with different information depending whether you choose to look at information for your entire database, a specific type of object, or a particular procedure. The information can be displayed in the following ways:

- details for all profiled objects within the database
- details for all stored procedures and functions
- details for all events
- details for all triggers
- details for all system triggers
- details for individual profiled objects

You must be connected to your database and have profiling enabled to view profiling information.

When you view profiling information for your entire database, the following columns appear:

- **Name** Lists the name of the object.
- **Owner** Lists the owner of the object.
- **Table** Lists which table a trigger belongs to (this column only appears on the database Profile tab).

- **Event** Shows the type of trigger for system triggers. This can be Update or Delete.
- **Type** Lists the type of object, for example, a procedure.
- **# Exes.** Lists the number times each object has been called.
- **#msecs.** Lists the total execution time for each object.

These columns provide a summary of the profiling information for all of the procedures that have been executed within the database. One procedure can call other procedures, so there may be more items listed than those users call specifically.

❖ **To view summary profiling information for stored procedures and functions**

1 Select the Procedures & Functions folder in the left pane.

2 Click the Profile tab in the right pane.

Profiling information about all the stored procedures and functions within your database appears on the Profile tab.

❖ **To view summary profiling information for events**

1 Open the Events folder in the left pane.

A list of all the events in your database appears on the Events tab in the right pane.

2 Click the Profile tab in the right pane.

Profiling information about all of the events within your database appears on the Profile tab.

❖ **To view summary profiling information for triggers**

1 Open the Triggers folder in the left pane.

A list of all the triggers in your database appears on the Triggers tab.

2 Click the Profile tab in the right pane.

Profiling information about all of the triggers in your database appears on the Profile tab.

❖ **To view summary profiling information for system triggers**

1 Open the System Triggers folder in the left pane.

A list of all the triggers in your database appears on the System Triggers tab.

2 Click the Profile tab in the right pane.

Profiling information about all of the system triggers in your database appears on the Profile tab.

Viewing profiling information for a specific procedure

Sybase IQ provides procedure profiling information about individual stored procedures, functions, events, and triggers. Sybase Central displays different information about individual procedures than it does about all of the stored procedures, functions, events, or triggers within a database.

When you look at the profiling information for a specific procedure, the following columns appear:

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

The procedure is broken down line by line and you can examine it to see which lines have longer execution times and therefore might benefit from changes to improve the procedure's performance. You must be connected to the database, have profiling enabled, and have DBA authority to access procedure profiling information.

❖ To view the profiling information for a stored procedure or function

1 Expand the database in the left pane.

2 Select the Procedures and Functions folder in the left pane.

A list of all the stored procedures and functions within your database appears on the Procedures & Functions tab in the right pane.

3 Click the stored procedure or function you want to profile in the left pane.

4 Click the Profile tab in the right pane.

Profiling information about the specific stored procedure or function appears on the Profile tab in the right pane.

❖ To view profiling information for an event

1 Expand the database in the left pane.

2 Select the Events folder in the left pane.

A list of all the events within your database appears on the Events tab in the right pane.

3. Click the event you want to profile in the left pane.
4. Click the Profile tab in the right pane.

   Profiling information about the specific event appears on the Profile tab in the right pane.

❖ **To view profiling information for triggers**

1. Expand the database in the left pane.
2. Open the Triggers folder in the left pane.

   A list of all the triggers appears on the Triggers tab in the right pane.
3. Select the trigger you want to profile in the right pane.
4. Click the Profile tab in the right pane.

   Profiling information about the specific trigger appears on the Profile tab in the right pane.

❖ **To view profiling information for system triggers**

1. Expand the database in the left pane.
2. Open the System Triggers folder in the left pane.

   A list of all the system triggers appears on the System Triggers tab in the right pane.
3. Select the system trigger you want to profile in the right pane.
4. Click the Profile tab in the right pane.

   Profiling information about the specific system trigger appears on the Profile tab in the right pane.

Viewing procedure profiling information in Interactive SQL

You can use stored procedures to view procedure profiling information. The profiling information is the same whether you view it in Sybase Central or in Interactive SQL.

The sa_procedure_profile_summary stored procedure provides information about all of the procedures within the database. You can use this procedure to view the profiling data for stored procedures, functions, events, system triggers, and triggers within the same result set. The following parameters restrict the rows the procedure returns.

- **p_object_name** Specifies the name of an object to profile.
- **p_owner_name** Specifies the owner whose objects you want to profile.
- **p_table_name** Specifies table to profile triggers.
- **p_object_type** Specifies the type of object to profile. You can choose from the following five options. Choosing one of these values restricts the result set to only objects of the specified type.
  - **P** Stored procedure
  - **F** Function
  - **T** Trigger
  - **E** Event
  - **S** System trigger
- **p_ordering** Specifies the sort order of the result set.

Keep in mind that there may be more items listed than those called specifically by users because one procedure can call another procedure.

The following sections assume that you are already connected to your database as a user with DBA authority and that you have procedure profiling enabled.

❖ To view summary profiling information for all procedures

1. Execute the sa_procedure_profile_summary stored procedure.

   For example, enter:

   ```
   CALL sa_procedure_profile_summary
   ```

2. From the SQL menu, choose Execute.

   A result set with information about all of the procedures in your database appears on the Results pane.

For more information about the sa_procedure_profile_summary stored procedure, see *SQL Anywhere Server – SQL Reference.*

Viewing profiling information for a specific procedure in Interactive SQL

The sa_procedure_profile stored procedure provides information about individual lines within specific procedures. The result set includes the line number, execution time, and percentage of total execution time for lines within procedures. You can use the following parameters to restrict the rows the procedure returns:

- **p_object_name** Specifies the name of an object to profile.
- **p_owner_name** Specifies the owner whose objects you want to profile.
- **p_table_name** Specifies which table to profile triggers.

If you do not include any parameters in your query, the procedure returns profiling information for all the procedures that have been called.

❖ To view profiling information for specific lines within procedures

1. Execute the sa_procedure_profile stored procedure.

   For example, enter:

   ```
   CALL sa_procedure_profile
   ```

2. From the SQL menu, choose Execute.

   A result set with profiling information for individual procedure lines appears in the Results pane.

For more information about the sa_procedure_profile stored procedure, see *SQL Anywhere Server – SQL Reference*.

Monitoring the buffer caches

Sybase IQ provides a tool to monitor the performance of the buffer caches. This monitor collects statistics on the buffer cache, memory, and I/O functions taking place within Sybase IQ, and stores them in a log file.

Buffer cache performance is a key factor in overall performance of Sybase IQ. Using the information the monitor provides, you can fine-tune the amount of memory you allocate to the main and temp buffer caches. If one cache is performing 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 changes in performance.

Starting the buffer cache monitor

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

Use this syntax to start the monitor:

> **IQ UTILITIES { MAIN | PRIVATE }**
> **INTO** *dummy_table_name*
> **START MONITOR** '*monitor_options* […]'

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

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

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

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

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

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

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

'*monitor_options*' can include one or more of the following values:

- -summary displays summary information for both the main and temp buffer caches. If you do not specify any monitor options, you receive a summary report. The fields displayed are as described for the other options, plus the following:
    - *Users*: Number of users connected to the buffer cache
    - *IO*: Combined physical reads and writes by the buffer cache
- -cache displays activity in detail for the main or temp buffer cache. Critical fields are Finds, HR%, and BWaits. The fields displayed are:
    - *Finds*: Find requests to the buffer cache. If the Finds value suddenly drops to zero and remains zero, the server is deadlocked. When the server has any activity, the Finds value is expected to be non-zero.
    - *Creats*: Requests to create a page within the database
    - *Dests*: Requests to destroy a page within the database
    - *Dirty*: Number of times the buffer was dirtied (modified)
    - *HR%*: Hit rate, the percentage of above satisfied by the buffer cache without requesting any I/O. The higher the Hit Rate the better, usually 90% - 100% if you set the cache large enough. For a large query, Hit Rate may be low at first, but increase as prefetching starts to work.
    - *BWaits*: Find requests forced to wait for a busy page (page frame contention). Usually it is low, but is some special cases it may be high. For example, if identical queries are started at the same time, both need the same page, so the second request must wait for the first to get that page from disk.
    - *ReReads*: Approximate number of times the same portion of the store needed to be reread into the cache within the same transaction. Should always be low, but a high number is not important for Sybase IQ 12.4.2 and above.
    - *FMiss*: False misses, number of times the buffer cache needed multiple lookups to find a page in memory. This number should be 0 or very small. If the value is high, it is likely that a rollback occurred, and certain operations needed to be repeated
    - *Cloned*: Number of buffers that Sybase IQ needed to make a new version for a writer, while it had to retain the previous version for concurrent readers. A page only clones if other users are looking at that page.

    - *Reads/Writes*: Physical reads and writes performed by the buffer cache
    - *PF/PFRead*: Prefetch requests and reads done for prefetch.
    - *GDirty*: Number of times the LRU buffer was grabbed dirty and Sybase IQ had to write it out before using it. This value should not be greater than 0 for a long period. If it is, you may need to increase the number of sweeper threads or move the wash marker.
    - *Pin%*: Percentage of pages in the buffer cache in use and locked.
    - *Dirty%*: Percentage of buffer blocks that were modified. Try not to let this value exceed 85-90%; otherwise, GDirty will become greater than 0.

- -cache_by_type produces the same results as -cache, but broken down by IQ page type. (An exception is the Bwaits column, which shows a total only.) This format is most useful when you need to supply information to Sybase Technical Support.
- -file_suffix *suffix* creates a monitor output file named `<dbname>.<connid>-<main_or_temp>-<suffix>`. If you do not specify a suffix, it defaults to *iqmon*.
- -io displays main or temp (private) buffer cache I/O rates and compression ratios during the specified interval. These counters represent all activity for the server; the information is not broken out by device. The fields displayed are:
    - *Reads*: Physical reads performed by the buffer cache
    - *Lrd(KB)*: Logical kilobytes read in (page size multiplied by the number of requests)
    - *Prd(KB)*: Physical kilobytes read in
    - *Rratio*: Compression ratio of logical to physical data read in, a measure of the efficiency of the compression to disk for reads
    - *Writes*: Physical writes performed by the buffer cache
    - *Lwrt(KB)*: Logical kilobytes written
    - *Pwrt(KB)*: Physical kilobytes written
    - *Wratio*: Compression ratio of logical to physical data written

- -bufalloc displays information on the main or temp buffer allocator, which reserves space in the buffer cache for objects like sorts, hashes, and bitmaps.
    - *OU*: User_Resource_Reservation option setting (formerly Optimize_For_This_Many_Users)
    - *AU*: Current number of active users
    - *MaxBuf*: Number buffers under control of the buffer allocator
    - *Avail*: Number of currently available buffers for pin quota allocation
    - *AvPF*: Number of currently available buffers for prefetch quota allocation
    - *Slots*: Number of currently registered objects using buffer cache quota
    - *PinUser*: Number of objects (for example, hash, sort, and B-tree objects) using pin quota
    - *PFUsr*: Number of objects using prefetch quota
    - *Posted*: Number of objects that are pre-planned users of quota
    - *UnPost*: Number of objects that are ad hoc quota users
    - *Locks*: Number of mutex locks taken on the buffer allocator
    - *Waits*: Number of times a thread had to wait for the lock
- -contention displays many key buffer cache and memory manager locks. These lock and mutex counters show the activity within the buffer cache and heap memory and how quickly these locks were resolved. Watch the timeout numbers. If system time exceeds 20%, it indicates a problem.

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

- *AU*: Current number of active users
- *LRULks*: Number times the LRU was locked (repeated for the temp cache)
- *woTO*: Number times lock was granted without timeout (repeated for the temp cache)
- *Loops*: Number times Sybase IQ retried before lock was granted (repeated for the temp cache)

  - *TOs*: Number of times Sybase IQ timed out and had to wait for the lock (repeated for the temp cache)
  - *BWaits*: Number of “Busy Waits” for a buffer in the cache (repeated for the temp cache)
  - *IOLock*: Number of times Sybase IQ locked the compressed I/O pool (repeated for the temp cache); can be ignored
  - *IOWait*: Number of times Sybase IQ had to wait for the lock on the compressed I/O pool (repeated for the temp cache); can be ignored
  - *HTLock*: Number of times Sybase IQ locked the block maps hash table (repeated for the temp cache)
  - *HTWait*: Number of times Sybase IQ had to wait for the block maps hash table (repeated for the temp cache); HTLock and HTWait indicate how many block maps you are using
  - *FLLock*: Number of times Sybase IQ had to lock the free list (repeated for the temp cache)
  - *FLWait*: Number of times Sybase IQ had to wait for the lock on the free list (repeated for the temp cache)
  - *MemLks*: Number of times Sybase IQ took the memory manager (heap) lock
  - *MemWts*: Number of times Sybase IQ had to wait for the memory manager lock

- -threads displays counter used by the processing thread manager. Values are server-wide (i.e., it does not matter whether you select this option for main or private). They represent new events after the last page of the report.
  - *cpus*: Number of CPUs Sybase IQ is using; this may be less than the number on the system
  - *Limit*: Maximum number of threads Sybase IQ can use
  - *NTeams*: Number of thread teams currently in use
  - *MaxTms*: Largest number of teams that has ever been in use
  - *NThrds*: Current number of existing threads
  - *Resrvd*: Number of threads reserved for system (connection) use
  - *Free*: Number of threads available for assignment. Monitor this value— if it is very low, it indicates thread starvation

- *Locks*: Number of locks taken on the thread manager
- *Waits*: Number of times Sybase IQ had to wait for the lock on the thread manager

---

Note When an object or query needs work, Sybase IQ allocates a group of processing threads called a thread team. Useful options in adjusting thread use include database options MAX_IQ_THREADS_PER_CONNECTION and MAX_IQ_THREADS_PER_TEAM, and the server option -iqmt which specifies the number of threads Sybase IQ can use.

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

  The first display shows counters from the start of the server. Subsequent displays show the difference from the previous display.
- -append | - truncate Append to existing output file or truncate existing output file, respectively. Truncate is the default.
- -debug is used mainly to supply information to Sybase Technical Support. It displays all the information available to the performance monitor, whether or not there is a standard display mode that covers the same information. The top of the page is an array of statistics broken down by disk block type. This is followed by other buffer cache statistics, memory manager statistics, thread manager statistics, free list statistics, CPU utilization, and finally buffer allocator statistics. The buffer allocator statistics are then broken down by client type (hash, sort, and so on) and a histogram of the most recent buffer allocations is displayed. Note that memory allocations indicate how much is allocated after the last page of the report.

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

Checking results while the monitor runs

On UNIX systems, you can watch monitor output as queries are running.

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

```
iq utilities main into monitor_tab
start monitor “-cache -interval 2 -file_suffix iqmon”
```

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

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

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

Stopping the buffer cache monitor

The command you use to stop a monitor run is similar to the one you use to start it, except that you do not need to specify any options. Use this syntax to stop the Sybase IQ buffer cache monitor:

IQ UTILITIES { MAIN | PRIVATE }
INTO *dummy_table_name* **STOP MONITOR**

Note In order for certain option settings to take effect you must restart the database. If the monitor is running you need to shut it down so that the database can be restarted.

Examining and saving monitor results

The monitor stores results in an ordinary text file. This file defaults to:

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

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

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

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

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

Examples of monitor results

This section shows sample results using different monitor options.

The -summary option produces results like the following. Note that it shows both main and temp buffer cache statistics, no matter which you request in the IQ UTILITIES command:

```
 Sybase Adaptive Server IQ Performance Monitor

               ---------------------------------------------

                           Version 3.2


Options string for Main cache: "-summary -interval 5"

                                                          Summary

                                                   2004-07-16 13:53:24

Active|               Main Cache                |           Temp Cache

 Users|  Finds   HR%  Reads/Writes  GDirty  Pin% Dirty% InUse%|  Finds   HR%
Reads/Writes  GDirty  Pin% Dirty% InUse%

     0     286  99.3      2/34          0   0.0    1.6   26.2     608  99.7
2/47          0   0.0    3.6   20.0

     1    2621  99.4     16/155         0   5.6    8.7   81.7    4121  99.6
16/163         0  11.4   23.2   67.3

     1    2646  99.8      6/48          0   1.6   13.5  100.0    3388  99.8
6/70          1   4.1   40.9   94.5

     1    2684  99.9      7/78          0   5.6   14.3  100.0    3497  99.9
8/103         1  10.9   42.3   99.1

     1    1993  99.9     17/22          0   4.0   31.0  100.0    3342  98.7
```

```
122/149          0   8.2   41.4   91.4

     1    2479  99.9     32/110          0   5.6   13.5  100.0    3370  99.8
55/112          0  11.4   45.5   95.9

     1    3273 100.0      0/0            0   5.6   23.8  100.0    3951 100.0
0/108           1  13.6   49.1  100.0

     1    2512  99.9      2/0            0   1.6   31.0  100.0    3916  98.9
88/173          0   5.5   48.6  100.0

     1    1264  99.9     66/131          0   4.0   45.2  100.0    4317  98.9
378/305          0   6.4   40.0   77.3

     1    2122  99.8     30/125          0   5.6   12.7   99.2    3122  99.7
67/127          0  12.3   40.0   90.5

     1    3370 100.0      2/0            0   5.6   23.0  100.0    4034 100.0
2/98            2  13.2   46.4   98.2

     1    2981  99.9      2/0            0   5.6   31.7  100.0    3715  99.9
2/110           0  14.1   53.2  100.0

     1    3351  99.6     13/3            0   5.6   39.7  100.0    4131  99.7
13/123          0  14.1   57.7  100.0

     1    3286  99.6     13/13           0   5.6   40.5  100.0    4135  99.6
15/139          0  12.3   55.9   97.7

     1     296 100.0      0/0            0   1.6   41.3  100.0    3646  96.9
366/320          0   7.3   53.2  100.0

     1    1230  99.4     71/129          0   6.3   58.7  100.0    4221  98.9
390/297          0   9.5   59.1   91.8

     1    1900 100.0    125/279          0   4.0   50.0  100.0    4102 100.0
344/279          0   7.7   38.6   72.3


                Sybase Adaptive Server IQ Performance Monitor

                ---------------------------------------------

                               Shutting Down


     0     422  98.8     16/99           0   0.0    0.8   99.2     853  98.9
34/101          0   0.0    1.8   59.1
```

The -cache option produces results like the following, which are for the temp buffer cache.

```
Options string for Temp cache: "-cache -interval 10"
```

```
                                        Temp Shared Buffer Cache
                                          2001-02-18 17:43:55
    Finds Creats Dests Dirty   HR% BWaits ReReads FMiss Cloned Reads/    PF/
GDirty Pin% Dirty%
                                                                     Writes PFRead
Tm:   640     82    57    84  99.4      0       4     0      0    4/0    0/0
0  0.0   2.8
Tm:  1139    109    83   109 100.0      0       0     0      0    0/0    0/0
0  0.0   5.5
Tm:  6794    754   749   754 100.0      0       0     0      0    0/0    0/0
0  0.0   6.1
Tm: 10759   1646  1646  1646 100.0      0       0     0      0    0/0    0/0
0  0.0   6.1
```

The -io option produces results like the following, which are for the main buffer cache:

```
Options string for main cache: "-IO -interval 5"
                                   Main Buffer Cache
                                  2001-02-18 13:58:48
                Input                             Output
      Reads Lrd(KB) Prd(KB)     Rratio  Writes  Lwrt(KB)  Pwrt(KB)  Wratio
Mn:      10      40      34       1.18      14        56        23    2.43
Mn:       0       0       0       0.00      21        84        34    2.43
Mn:       0       0       0       0.00       7        28        11    2.43
Mn:       0       0       0       0.00      22        88        35    2.48
Mn:       0       0       0       0.00      63       252       100    2.51
Mn:       0       0       0       0.00      54       216        93    2.32
Mn:       0       0       0       0.00      64       256       101    2.52
Mn:       0       0       0       0.00      62       248        94    2.62
Mn:       0       0       0       0.00      73       292       110    2.65
Mn:       0       0       0       0.00     105       420       121    3.47
```

The -buffalloc option produces results like the following.

```
 Options string for Main cache: "-bufalloc -file_suffix bufalloc-iqmon -append
-interval 10"

                                        Buffer Allocation
2001-02-18 10:58:39

OU/AU MaxBuf  Avail AvPF  Slots PinUsr  PFUsr Posted UnPost Quota Locks  Waits
1/0    1592   1592    20      0      0      0      0      0     0     1      0

1/1    1592   1592    20      0      0      0      0      0     0     1      0
```

```
1/1    1592   1592    20      0      0      0      0      0      0      1      0
```

Note The actual -contention output shows Main Cache, Temp Cache, and Memory Manager on the same line. Because this format is very wide, each of these sets of columns is shown separately here.

The -contention results for the main cache are:

```
Options string for Main cache:
"-contention -file_suffix contention-iqmon -append -interval 10"
                       Contention
                       2001-02-18 10:57:03


                      Main Cache                                            |
AU |LRULks woTO  Loops    TOs BWaits IOLock IOWait HTLock HTWait FLLock FLWait

 0     66     0      0     0     0      1      0     5      0      4   0
 1   2958     0      0     0     0    160      0  1117      0      6   0
 1   1513     0      0     0     1    378      0     2      0      8   0
 1    370     0      0     0     0     94      0     2      0     10   0
 1    156     0      0     0     0     46      0     2      0     12   0
 1    885     0      0     0     0    248      0     2      0     14   0
 1   1223     0      0     0     0    332      1     2      0     16   0
 1    346     0      0     0     0     66      0     2      0     18   0
```

The -contention results for the temp cache are:

```
                             Temp Cache
|LRULks   woTO  Loops    TOs BWaits IOLock IOWait HTLock HTWait FLLock FLWait
70        0      0      0      0      1      0      4      0      5      0
466       0      0      0      0      2      0     15      0     12      0
963       0      0      0      0      2      0      8      0     20      1
1186      0      0      0      0      2      0      2      0     23      1
357       0      0      0      0      2      0      2      0     25      1
444       0      0      0      0      2      0      3      0     29      0
884       0      0      0      0      2      0      2      0     31      1
1573      0      0      0      0      2      0      5      0     37      1
```

The results for the memory manager are:

```
| Memory Mgr
MemLks MemWts
55483    13
5705      0
2048      0
```

```
186       4
2         0
137       0
22        0
203       3
```

The results of the -threads option look like the following:

```
 Options string for Main cache: "-threads -file_suffix threads-iqmon -append -
interval 10"


                                                    Threads

                                             2001-02-18 10:59:24



   CPUs   Limit  NTeams  MaxTms  NThrds   Resrvd  Free   Locks  Waits

   10     100       4      12     100      13      68     106    590

   10     100       6      12     100      12      63       4      6

   10     100       6      12     100      12      63       0      0

   10     100       7      12     100      12      62       1      1

   10     100       7      12     100      12      62       0      0

   10     100       7      12     100      12      58       1      5

   10     100       7      12     100      12      58       0      0
```

Buffer cache structure

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

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

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

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

Avoiding buffer manager thrashing

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

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

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

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

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

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

If you are experiencing dramatic performance problems, you should monitor paging to determine if thrashing is a problem. If so, then reset your buffer sizes as described in “Managing buffer caches”.

If you monitor paging and determine that thrashing is a problem, you can also limit the amount of thrashing during the execution of a statement which includes a query that involves hash algorithms. Adjusting the HASH_THRASHING_PERCENT database option controls the percentage of hard disk I/Os allowed before the statement is rolled back and an error is returned.

The default value of HASH_THRASHING_PERCENT is 10%. Increasing HASH_THRASHING_PERCENT permits more paging to disk before a rollback and decreasing HASH_THRASHING_PERCENT permits less paging before a rollback.

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

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

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

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

Monitoring paging on Windows systems

Windows provides the System Monitor to help you monitor paging. To access it, 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.

Monitoring paging on UNIX systems

UNIX provides a system command, vmstat, to help you monitor system activity such as paging. The abbreviated command syntax is:

vmstat *interval count*

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

```
>  vmstat 2 3
procs    memory            page             disk      faults      cpu
r b w swap    free   re mf pi po fr de sr s0 s1 sd  in    sy  cs us sy id

0 0 0 3312376 31840 0  8  0  0  0  0  0   0  0  0 297  201 472 82  4 14
0 0 0 3312376 31484 2  3  0  0  0  0  0   0  0  0 260  169 597 80  3 17
0 0 0 3312368 31116 0  8  0  0  0  0  0   0  0  0 205 1202 396 67  4 29
```

The above output shows a steady Sybase IQ querying state where the physical memory of the machine has not been overallocated. Little to no system page faulting is occurring. These next set of examples show vmstat output that indicates a problem. (The output shown omits some of the above fields to fit better on the page.)

```
procs     memory            page              faults       cpu
r b w swap    free    re   mf  pi po fr de sr  in   sy   cs us sy  id

0 0 0 217348  272784  0  148  11  3  9  0  2 251 1835  601  6  3  91
0 0 0 3487124 205572  0    5   0  0  0  0  0  86  131  133  0  1  99
0 0 0 3487124 205572  0    5   0  0  0  0  0  71  162  121  0  0 100
0 0 0 3483912 204500  0  425  36  0  0  0  0 169  642  355  2  2  96
0 0 0 3482740 203372  0   17   6  0  0  0  0 158  370  210  1  3  97
0 0 0 3482676 203300  0    4  10  0  0  0  0 160 1344  225  1  2  97
0 0 0 3343272 199964  1 2123  36  0  0  0  0 213  131  399  7  8  85
0 0 0 3343264 185096  0  194  84  0  0  0  0 283  796  732  1  6  93
0 0 0 3342988 183972  0   17  58  0  0  0  0 276 1051  746  2  4  94
0 0 0 3342860 183632  0  119 314  0  0  0  0 203 1660  529  3  4  94
0 0 0 3342748 182316  2  109 184  0  0  0  0 187  620  488  4  2  95
0 0 0 3342312 181104  2  147  96  0  0  0  0 115  256  260  9  2  89
0 0 0 3340748 179180  0  899  26  0  0  0  0 163  836  531  4  4  92
0 0 0 3328704 167224  0 2993   6  0  0  0  0  82 2195  222  4  7  89
```

The first line of the above output provides a summary of the system activity after the machine was started. The first three lines show that there is approximately 200MB of free physical memory and that the machine is idle. The fourth line corresponds to Sybase IQ starting up for the first time. Beginning at the eighth line, the amount of free memory starts to reduce rapidly. This corresponds to the Sybase IQ buffer caches being allocated and database pages being read in from disk (note that CPU usage has increased). At this time there is little user CPU time as no queries have begun.

```
procs     memory                 page              faults      cpu
r  b w swap    free   re mf   pi   po fr de sr in     sy  cs   us sy id

 7 0 0 3247636 58920  0 1880 1664 0  0  0  0  1131  442 1668 80 18  3
18 0 0 3246568 43732  0  709 1696 0  0  0  0  1084  223 1308 90 10  1
12 0 0 3246604 37004  0  358  656 0  0  0  0   600  236  722 95  5  0
15 0 0 3246628 32156  0  356 1606 0  0  0  0  1141  226 1317 91  9  0
19 0 0 3246612 26748  0  273 1248 0  0  0  0   950  394 1180 92  7  0
```

The above output is from slightly later when the query is underway. This is evident from the user mode CPU level (*us* field). The buffer cache is not yet full as page-in faults (*pi* field or KB paged in) are still occurring and the amount of free memory is still going down.

```
procs     memory            page                   faults        cpu
r  b w swap    free  re mf pi   po  fr   de sr  in   sy   cs   us sy id

21 0 0 3246608 22100 0 201 1600   0    0 0    0 1208 1257 1413 88 12  0
18 0 0 3246608 17196 0 370 1520   0  464 0  139  988  209 1155 91  8  0
11 0 0 3251116 16664 0 483 2064 138 2408 0  760 1315  218 1488 88 12  0
30 0 0 3251112 15764 0 475 2480 310 4450 0 1432 1498  199 1717 87 13  0
```

The above output is from even later. On the third line of the output it shows that the system has reached its threshold for the amount of free memory it can maintain. At this point, page-outs (*po* field or KB paged out) occur and the level of system mode CPU (*sy* field) increases accordingly. This situation results because physical memory is overallocated: the Sybase IQ buffer caches are too big for the machine. To resolve this problem, reduce the size of one or both of the buffer caches.

Buffer cache monitor checklist

The following table summarizes the most common items to look for in monitor results, and suggests actions you may need to take if behavior is outside the normal range. The Statistic column lists the name you see in the standard monitor reports; if this statistic appears differently in the debug report, the debug statistic is also listed.

Remember that for any monitor statistic, a temporary anomaly may occur while the system changes state, such as when a new query is starting.

Table 5-11: Buffer cache monitor checklist

Statistic	Normal behavior	Behavior that needs adjusting	Recommended action
HR% (Cache hit rate)	Above 90%. For individual internal data structures like garray, barray, bitmap (bm), hash object, sort object, variable-length btree (btreev), fixed-length btree (btreef), bit vector (bv), dbext, dbid, vdo, store, checkpoint block (ckpt), the hit rate should be above 90% while a query runs. It may be below 90% at first. Once prefetch starts working (PF or PrefetchReqs > 0), the hit rate should gradually grow to above 90%.	Hit rate below 90% after prefetch is working. **Note** Some objects do not do prefetching, so their hit rate may be low normally.	Try rebalancing the cache sizes of main versus temp by adjusting -iqmc and -iqtc. Also try increasing the number of prefetch threads by adjusting PREFETCH_THREADS_PERCENT option.
Gdirty (Grabbed Dirty)	0 in a system with a modest cache size (< 10GB).	GDirty > 0 **Note** Sweeper threads are activated only when the number of dirty pages reaches a certain percentage of the wash area. If GDirty/GrabbedDirty is above 0 and the I/O rate (Writes) is low, the system may simply be lightly loaded, and no action is necessary.	Adjust SWEEPER_THREADS_PERCENT option (default 10%) or WASH_AREA_BUFFERS_PERCENT option (default 20%) to increase the size of the wash area.

Statistic	Normal behavior	Behavior that needs adjusting	Recommended action
BWaits (Buffer Busy Waits)	0	Persistently > 0, indicating that multiple jobs are colliding over the same buffers.	If the I/O rate (Writes) is high, Busy Waits may be caused by cache thrashing. Check Hit Rate in the cache report to see if you need to rebalance main versus temp cache. If a batch job is starting a number of nearly identical queries at the same time, try staggering the start times.
LRU Waits (LRUNum TimeOuts percentage in debug report)	20% or less	> 20%, which indicates a serious contention problem.	Check the operating system patch level and other environment settings. This problem tends to be an O.S. issue.
IOWait (IONumWaits)	10% or lower	> 10%	Check for disk errors or I/O retries
FLWait (FLMutexWaits)	20% or lower	> 20%	Check the dbspace configuration: Is the database almost out of space? Is DISK_STRIPING ON? Does sp_iqcheckdb report fragmentation greater than 15%?
HTWait (BmapHTNumWaits) MemWts (MemNtimes Waited) (PFMgrCond VarWaits)	10% or lower	> 10%	Contact Sybase Technical Support.

Statistic	**Normal behavior**	**Behavior that needs adjusting**	**Recommended action**
CPU time (CPU Sys Seconds, CPU Total Seconds, in debug report)	CPU Sys Seconds < 20%	CPU Sys Seconds > 20% If CPU Total Seconds also reports LOW utilization, and there are enough jobs that the system is busy, the cache may be thrashing or parallelism may be lost.	Adjust -iqgovern to reduce allowed total number of concurrent queries. Check Hit Rate and I/O Rates in the cache report for cache thrashing. Also check if hash object is thrashing by looking at the hit rate of the has object in cache_by_type (or debug) report: is it <90% while the I/O rate (Writes) is high? Check query plans for attempted parallelism. Were enough threads available? Does the system have a very large number of CPUs? Strategies such as multiplex configuration may be necessary.
InUse% (Buffers in use)	At or near 100% except during startup	Less than about 100%	The buffer cache may be too large. Try rebalancing the cache sizes of main versus temp by adjusting -iqmc and -iqtc.
Pin% (Pinned buffers)	< 90%	> 90 to 95%, indicating system is dangerously close to an Out of Buffers condition, which would cause transactions to roll back	Try rebalancing the cache sizes of main versus temp. If rebalancing buffer cache sizes is not possible, try reducing -iqgovern to limit the number of jobs running concurrently.

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

System utilities to monitor CPU use

In addition to the performance monitor in Sybase Central, you can use operating system utilities to monitor CPU usage while using Sybase IQ.

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

CHAPTER 6

Tuning Servers on 32-bit Windows Systems

This chapter provides performance and tuning guidelines specific to running Sybase IQ on Windows systems. Use this chapter in conjunction with Chapter 4, “Managing System Resources.”

Topic	Page
General performance guidelines	138
Monitoring performance	139
Using the NTFS cache	141
Tuning inserts and queries	142
Tuning backup operations	143

General performance guidelines

The following are general guidelines that apply to both loading and querying data. The recommended minimum amount of memory (RAM) for running Sybase IQ under Windows is 512MB. For most applications 4GB is recommended for best performance. Set the /3GB and /PAE switches in the boot.ini to allow Sybase IQ to address as much memory as possible on Windows 32bit systems.

Maximizing throughput

If you are running on Windows, make sure the Network Services Server option “Maximize Throughput for Network Applications” is enabled.

❖ **Maximize Throughput for Network Applications**

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

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

Preventing memory over allocation

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

Monitoring physical memory

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

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

File systems

The Windows file system supports compression at the file, directory and volume level. *Check the compression options and disable Windows file system compression on all disks and volumes where you store* Sybase IQ *databases.* This is because Sybase IQ provides built-in compression. The file system compression will be unable to reduce the database size further, but may add CPU overhead when performing reads or writes.

Monitoring performance

Your primary tool for monitoring Sybase IQ performance is the Sybase IQ performance monitor, described in Chapter 5, “Monitoring and Tuning Performance.” However, you can also use Windows Task Manager and Windows Performance Tool to monitor system performance.

Task Manager

Windows Task Manager provides information about the applications and processes running on your computer. Task Manager also displays details about machine performance and resources.

- To open Task Manager, right-click an empty space on the taskbar, and choose Task Manager.
- For more information about using Windows Task Manager, click Help, then choose Task Manager Help Topics in Windows Task Manager.

Performance

Windows Performance tools include System Monitor and Performance Logs and Alerts. System Monitor displays real-time data about memory, disk, processor, and network activity. Performance Logs and Alerts lets you record performance data and set alerts for specific performance activities.

- To open the Windows Performance Tools, on the Control Panel, click Administrative Tools, then choose Performance.
- For more information about using Performance Tools, click Help, then choose Help Topics in Performance.

Monitoring virtual address space and working set

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

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

❖ Monitoring virtual address space and memory usage

1. Right-click an empty space on the taskbar, choose Task Manager, then click the Processes tab.
2. Click View, choose Select Columns.
3. On the Select Columns dialog, choose these columns:
   - Memory usage
   - Memory usage delta
   - Peak memory usage
   - Page faults
   - Page faults delta
   - Virtual memory size

Monitoring page faults

From the Windows Performance Monitor select the Sybase IQ process as described above. Select the counter Page Faults/sec. This counter includes both the “soft” and “hard” page faults. Hard page faults are the page faults resulting in disk I/O. Soft page faults in general are not a performance issue.

To determine the number of hard page faults, select the object LogicalDisk and the instance of where the file *pagefile.sys* is located (this should be on a separate volume from the Sybase IQ database). Select the counter Disk Transfers/sec. This value when compared with the Page Faults/sec value will give an indication of the percentage of page faults that are hard page faults. Ideally there should be little to no I/O activity to the page file. In small memory configurations, however, paging is likely to occur.

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

Using the NTFS cache

With the Network Services Server option “Maximize Throughput for Network Applications” on page 138 enabled, using the NTFS and its associated cache can improve Sybase IQ database performance for both inserts and queries. This is largely due to the NTFS being able to store significantly more data than the Sybase IQ buffer cache with the same amount of physical memory. This advantage can be most effectively be leveraged by reducing the size of the main cache, giving this memory to the NTFS which can use it more effectively. In IQ 15.x more use is made of temporary caches in load and query processing. As a result, using the NTFS cache is still a useful technique for the main cache, but may not provide the same advantages for the temporary cache.

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

Tuning inserts and queries

This section provides additional guidelines for tuning inserts and queries on Windows platforms.

Characteristics of well-tuned insert operations

A well-tuned Sybase IQ insert operation exhibits certain characteristics. You can observe these characteristics from the Windows Task Manager and Windows Performance Monitor.

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

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

Load performance can be further improved, sometimes significantly, by setting the size of the main Sybase IQ buffer cache considerably smaller than the calculated recommended values in “Sybase IQ main and temp buffer caches” on page 62. The reasons for this performance improvement are described in “Using the NTFS cache” on page 141. You can set the main Sybase IQ buffer cache as much as 50% smaller than the calculated recommended values.

Tuning for queries

You may also improve query performance by reducing the size of the main buffer cache as described in the previous paragraph. See Chapter 3, “Optimizing Queries and Deletions,” for details about query plans, structure, and options.

Tuning backup operations

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

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

For any backup or restore:

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

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

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

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

APPENDIX A Performance statistics

sp_iqsysmon is a stored procedure that can monitor multiple Sybase IQ components, including buffer cache management, memory, threads, locks, I/O functions, and CPU utilization. sp_iqsysmon runs in batch or file mode. Performance statistics are output by section.

This document defines sp_iqsysmon sections, statistics, and field names.

Topic	Page
Field reference	146
Statistics	154

Field reference

Table A-1 identifies sp_iqsysmon statistics by section; Table A-2 provides links to field name definitions by statistic type.

Table A-1: Section names

Buffer manager	Buffer pool
Prefetch manager	IQ store free list
Buffer allocator	Memory manager
Thread manager	CPU time statistics
Transaction manager	Context Server statistics
Catalog, DB Log, and Repository statistics	

Table A-2: Field name definitions

Statistics	Page Types
Lock Statistics	Buffer Allocator/Prefetch Client Types

Note See “System Stored Procedures” in *Reference: Building Blocks, Tables, and Procedures* for more information about sp_iqsysmon.

Buffer manager

Buffer manager manages the pool of in-memory buffers for the buffer cache. sp_iqsysmon generates main (Table A-3) and temporary (Table A-4) buffer cache statistics. Statistic names are sorted alphabetically.

Table A-3: Buffer manager main statistics

BlockmapMutexs NLocks	BlockmapMutexs NWaits
BlockmapRegEver	BlockmapRegisters
BlockmapUID	BlockmapUIDnallocs
BmapHTMaxEntries	BmapHTNClears
BmapHTNCollisn	BmapHTNEntries
BmapHTNFinds	BmapHTNHits
BmapHTNHits1	BmapHTNHits2
BmapHTNInserts	BmapHTNLChain
BmapHTNRehash	BmapHTNumLocks
BmapHTNumWaits	BufAllocAvailBufs
BufAllocAvailPF	BufAllocMaxBufs
BufHTNFoiledOps	BufHTNw2orMore
BusyWait	CacheTeamNum Asleep
CacheTeamTimes Woken	Creates
Destroys	Dirties
FalseMiss	Finds
GrabbedDirty	HitRate
Hits	IONumLocks
IONumWaits	LRUNumLocks
LRUNumSpinLoops	LRUNumSpinsWoTO
LRUNumTimeOuts	PReadBlks
PWriteKB	PrefetchInMem
PrefetchNotInMem	PrefetchReqs
PWriteBlks	PWriteKB
Reads	Realdirties
ReReads	UnOwnRR
Writes	

Table A-4: Buffer manager temporary statistics

BlockmapMutexs NLocks	BlockmapMutexs NWaits
BlockmapRegEver	BlockmapRegisters
BlockmapUID	BlockmapUIDnallocs

BmapHTMaxEntries	BmapHTNClears
BmapHTNCollisn	BmapHTNEntries
BmapHTNFinds	BmapHTNHits
BmapHTNHits1	BmapHTNHits2
BmapHTNInserts	BmapHTNLChain
BmapHTNRehash	BmapHTNumLocks
BmapHTNumWaits	Buffer Pool main
BufHTNEntries	BufHTNFoiledOps
BufHTNw2orMore	BusyWait
CacheTeamNum Asleep	CacheTeamTimes Woken
Cloned	Creates
Destroys	Dirties
Finds	FalseMiss
Flushes	GetPageFrame
GetPageFrameFailure	GotEmptyFrame
GrabbedDirty	HitRate
Hits	InUse
IONumLocks	IONumWaits
LRUNumLocks	LRUNumSpinLoops
LRUNumSpinsWoTO	LRUNumTimeOuts
MovedToMRU	MovedToWash
Pages	Pinned
PWriteBlks	PWriteKB
PrefetchInMem	PrefetchNotInMem
PrefetchReqs	PWriteBlks
PWriteKB	**Reads**
Realdirties	RemovedFromLRU
RemovedFromWash	RemovedInScanMode
ReReads	TimesSweepers Woken
UnOwnRR	Washed
washIntensity	WashMaxSize
washNActive Sweepers	washNBuffers
washNBuffers	washSignalThreshold
washTeamSize	Writes

Buffer pool

The buffer pool manages in-memory buffers for the buffer cache. sp_iqsysmon generates main (Table A-5) and temporary (Table A-6) buffer cache statistics. Statistic names are sorted alphabetically.

Table A-5: Buffer pool main statistics

Dirty	FlushedBufferCount
Flushes	GetPageFrame
GetPageFrameFailure	GotEmptyFrame
InUse	MovedToMRU
MovedToWash	Pages
Pinned	RemovedFromLRU
RemovedFromWash	RemovedInScanMode
TimesSweepers Woken	Washed
washIntensity	WashMaxSize
washNActive Sweepers	washNBuffers
washNDirtyBuffers	washSignalThreshold
washTeamSize	

Table A-6: Buffer pool temporary statistics

Dirty	FlushedBufferCount
Flushes	GetPageFrame
GetPageFrameFailure	GotEmptyFrame
InUse	MovedToMRU
MovedToWash	Pages
Pinned	RemovedFromLRU
RemovedFromWash	RemovedInScanMode
TimesSweepers Woken	Washed
washIntensity	WashMaxSize
washNActive Sweepers	washNBuffers
washNDirtyBuffers	washSignalThreshold
washTeamSize	

Prefetch manager

Prefetch manager manages asynchronous reads of prefetched pages. sp_iqsysmon generates main (Table A-7) and temporary (Table A-8) prefetch manager statistics. Statistic names are sorted alphabetically.

Table A-7: Prefetch manager main statistics

PFMgrCondVar	PFMgrNDropped
PFMgrNRead	PFMgrNReading
PFMgrNSubmitted	PFMgrNThreads
PFMgrNValid	

Table A-8: Prefetch manager temporary statistics

PFMgrCondVar	PFMgrNDropped
PFMgrNRead	PFMgrNReading
PFMgrNSubmitted	PFMgrNThreads
PFMgrNValid	

IQ store free list

The IQ store free list maps disk blocks in use by the database. sp_iqsysmon generates main (Table A-9) and temporary (Table A-10) free list statistics for the IQ store. Statistic names are sorted alphabetically.

Table A-9: IQ store main free list statistics

FLBitCount	FLIsOutOfSpace
FLMutexLocks	FLMutexWaits

Table A-10: IQ store temporary free list

FLBitCount	FLIsOutOfSpace
FLMutexLocks	FLMutexWaits

Buffer allocator

The buffer allocator manages buffer pinning quotas for (temporary) data structures. sp_iqsysmon generates main (Table A-11) and temporary (Table A-12) statistics. Statistic names are sorted alphabetically.

Table A-11: Buffer allocator main statistics

BM	BTree
BufAllocAvailBufs	BufAllocAvailPF
BufAllocMaxBufs	BufAllocMutexLocks
BufAllocMutexWaits	BufAllocNPFUsers
BufAllocNPinUsers	BufAllocNPostedUsrs
BufAllocNPostEst	BufAllocNUnPostEst
BufAllocNUnpostUsrs	BufAllocPinQuota
BufAllocReserved	BufAllocSlots
BV	ClientCountOfPinners
DBCC	FP
Garray	Hash
LOB	NActiveCommands
NumClients	PfUserRegisters
PinUserQuota	PinUserRegisters
PrefetchUserQuota	Row
RowColumn	Sort
Store	

Table A-12: Buffer allocator temporary statistics

BM	BTree
BufAllocAvailBufs	BufAllocAvailPF
BufAllocMaxBufs	BufAllocMutexLocks
BufAllocMutexWaits	BufAllocNPFUsers
BufAllocNPinUsers	BufAllocNPostedUsrs
BufAllocNPostEst	BufAllocNUnPostEst
BufAllocNUnpostUsrs	BufAllocPinQuota
BufAllocReserved	BufAllocSlots
BV	ClientCountOfPinners
DBCC	FP
Garray	Hash
LOB	NActiveCommands
NumClients	PfUserRegisters

PinUserQuota	PinUserRegisters
PrefetchUserQuota	Row
RowColumn	Sort
Store	

Memory manager

sp_iqsysmon generates statistics (Table A-13) that identify how memory manager allocates server heap memory. Statistic names are sorted alphabetically.

Table A-13: Memory Manager statistics

MemAllocated	MemAllocatedEver
MemAllocatedMax	MemNAllocated
MemNAllocatedEver	MemNTimesLocked
MemNTimesWaited	

Thread manager

sp_iqsysmon generates statistics (Table A-14) that identify how the thread manager handles the global thread pool for the server. Statistic names are sorted alphabetically.

Table A-14: Thread Manager statistics

NumTeamsAlloc	NumThrUsed
SingleThrAlloc	TeamThrAlloc
ThreadLimit	ThrMaxTeams
ThrMutexLocks	ThrMutexWaits
ThrNTeamsInUse	ThrNumFree
ThrNumOfCpus	ThrNumThreads
ThrReserved	UsedPerActiveCmd

CPU time statistics

sp_iqsysmon generates CPU time statistics (Table A-15) that identify IQ CPU usage as reported by the operating system. Statistic names are sorted alphabetically.

Table A-15: CPU time statistics

CPU Sys Seconds	CPU Total Seconds
CPU User Seconds	Elapsed Seconds

Transaction manager

sp_iqsysmon generates statitistics (Table A-16) that identify how the transaction manager manages transaction serialization and version allocation. Statistic names are sorted alphabetically.

Table A-16: Transaction Manager statistics

TxnMgrNBlocked	TxnMgrNPending
TxnMgrNWaiting	TxnMgrOAVI
TxnMgrPCcondvar	TxnMgrtxncblock
TxnMgrTxnIDseq	TxnMgrVersionID
TxnMgrVersionLock	

Context Server statistics

sp_iqsysmon generates context server statitistics (Table A-16) that provide resource governor metrics. Statistic names are sorted alphabetically.

Table A-17: Context Server statistics

StCntxCondVar	StCntxLock
StCntxNAdmitted	StCntxNOrigResource
StCntxNResource	StCntxNumConns
StCntxNWaited	StCntxNWaiting

Catalog, DB Log, and Repository statistics

sp_iqsysmon generates statistics (Table A-17) for the catalog and statistics repository. Statistic names are sorted alphabetically.

Table A-18: Catalog, DB Log, and Repository statistics

CatalogLock	DbLogMLock
DbLogSLock	RepositoryLock
RepositoryNList	

Statistics

BlockmapMutexs NLocks	Number of blockmap locks.
BlockmapMutexs NWaits	Number of blockmap locks that waited.
BlockmapRegEver	Number of blockmaps instantiated since database creation.
BlockmapRegisters	Number of blockmaps instantiated in the monitor interval.
BlockmapUID	Current blockmap unique ID.
BlockmapUIDnallocs	Number of blockmaps allocated in the monitor interval.
BmapHTMaxEntries	Ideal Value.
BmapHTNClears	Number of entries removed from buffer hash.
BmapHTNCollisn	Number of buffer hash table entries inserted into non-empty hash buckets .
BmapHTNEntries	Number of entries in the blockmap hash table.
BmapHTNFinds	Number of buffer manager hash table probes.
BmapHTNHits	Number of buffer manager hash table probes that were successful.
BmapHTNHits1	Number of buffer hash find calls that hit on 1st try.
BmapHTNHits2	Number of buffer hash find calls that hit on 2nd try.
BmapHTNInserts	Number of blockmap pages that were inserted into the buffer hash table .
BmapHTNLChain	Length of longest chain in buffer hash table.
BmapHTNRehash	Number of rehashes due to buffer hash table growing.
BmapHTNumLocks	Number of times the buffer hash table was locked.
BmapHTNumWaits	Number of waits on buffer hash table lock.

BufAllocAvailBufs	Count of buffers available to be pinned.
BufAllocAvailPF	Count of buffers available for prefetch.
BufAllocMaxBufs	Ideal Value.
BufAllocMutexLocks	Count of locks taken on the buffer allocator.
BufAllocMutexWaits	Count of buffer allocator locks that waited.
BufAllocNPFUsers	Count of indexes/structures using prefetch buffers.
BufAllocNPinUsers	Count of indexes/structures using pinned buffers.
BufAllocNPostEst	Number of calls to post estimated future users.
BufAllocNPostedUsrs	Count of pinned buffer requested in advance.
BufAllocNUnPostEst	Number of calls to unpost estimated future users.
BufAllocNUnpostUsrs	Count of pinned buffers not requested in advance.
BufAllocPinQuota	Ideal Value.
BufAllocReserved	Ideal Value.
BufAllocSlots	Total buffers managed by buffer allocator.
BufHTMaxBucketSize	Length of longest buffer hash chain.
BufHTNBuckets	Number on buffer hash table buckets.
BufHTNEntries	Ideal Value.
BufHTNFoiledOps	Number of operations that had to reacquire the table lock.
BufHTNw2orMore	Number of buffer hash table buckets containing 2 or more entries.
BusyWait	Find requests that waited while a page was being read from disk or written to disk.
Cloned	Buffers that were cloned to create new versions since previous versions were locked by queries.
CacheTeamNum Asleep	Number of buffer cache threads currently inactive.
CacheTeamTimes Woken	Description.
CatalogLock	Lock used to access IQ catalog.
ClientCountOfPinners	Number of buffer allocator clients that have pinned buffers.
CPU User Seconds	Number of user mode CPU seconds.

CPU Sys Seconds	Number of system mode CPU seconds.
CPU Total Seconds	Total of system and user seconds.
Creates	Number of New Pages Created.
DbLogMLock	IQ Message File lock.
DbLogSLock	IQ Message File lock.
Destroys	Number of pages destroyed.
Dirties	Number of page modifications requests.
Dirty	Percentage of pages that have been modified.
Elapsed	Number of CPU seconds used during the run interval.
FalseMiss	Secondary cache lookup found the correct buffer.
FLBitCount	Number of blocks allocated in store (main or temp).
Finds	Number of Finds that were already in the cache.
FLIsOutOfSpace	Flag to indicate when the store is fully allocated (full).
FLMutexLocks	Number of times the freelist was locked.
FLMutexWaits	Number of lock waits on free list.
FlushedBufferCount	Number of buffers written to disk by flush.
Flushes	Number of times the flush operator was called.
GetPageFrame	Number of clean buffers requested from buffer manager.
GetPageFrameFailure	Number of request failed (error due to all buffers locked).
GotEmptyFrame	Number of buffers that were found clean.
GrabbedDirty	Pages that were grabbed dirty and synchronously written to disk to free a buffer.
HitRate	Percentage of Finds that were already in the cache.
IONumLocks	Number of locks on temporary compression buffer pool.
IONumWaits	Locks on temporary compression buffer pool that waited.
InUse	Number of buffers in cache marked as being used.
LRUNumLocks	Number of times that the buffer manager LRU was locked.
LRUNumSpinsWoTO	Number of times that a thread waited to acquire the buffer manager LRU lock .
LRUNumSpinLoops	Number of spin cycles that the buffer manager LRU lock went through .

LRUNumTimeOuts	Number of times that a thread timed out waiting to acquire the buffer manager LRU lock.
MemAllocated	Amount of memory currently allocated from Heap.
MemAllocatedEver	Maximum amount of memory allocated since server started.
MemAllocatedMax	Maximum amount of memory allocated in monitor interval.
MemNAllocated	Number of requests to allocate memory.
MemNAllocatedEver	Number of requests to allocate memory since server started.
MemNTimesLocked	Number of times the heap was lock.
MemNTimesWaited	Number of lock waits on heap.
MovedToMRU	Number of buffers that after use were put back to MRU.
MovedToWash	Number of buffers that after use were put straight into the wash area .
NActiveCommands	Current count of active commands.
NumClients	Number of clients currently registered with the buffer allocator.
NumTeamsAlloc	Total number of teams during monitor interval.
NumThrUsed	Actual threads allocated by server.
PFMgrCondVar	Statistics for prefetch manager lock.
PFMgrNDropped	Number of prefetch requests that were dropped.
PFMgrNRead	Number of Prefetch requests started.
PFMgrNReading	Number of PF reads completed.
PFMgrNSubmitted	Number of prefetch requests submitted.
PFMgrNThreads	Number of threads reserved for prefetch.
PFMgrNValid	Number of valid prefetch requests.
Pages	Number of buffers in cache.
PrefetchInMem	A page requested for prefetch was already in memory.
PrefetchNotInMem	Find of page requested for prefetch that resulted in a read.
PfUserRegisters	Number of times a client registered with the prefetch manager allocator for prefetch.
Pinned	Percent of buffers that are pinned in memory (locked).

PinUserQuota	Maximum number of clients that can register with the buffer allocator for pinned buffers.
PinUserRegisters	Number of times a client registered with the buffer allocator for pinned buffers.
PReadBlks	Number of database blocks read.
PReadKB	Size of database blocks read in KB.
PrefetchReqs	Pages Requested for asynchronous prefetch.
PrefetchUserQuota	Maximum number of clients that can register with the buffer allocator for prefetch.
PWriteBlks	Number of database blocks written.
PWriteKB	Description.
Realdirties	Description.
RemovedFromLRU	Number of buffers that in the LRU when a Find occurred.
RemovedFromWash	Number of buffers that were in the wash area when a Find occurred .
RemovedInScanMode	Number of buffers that were in scan mode when a find occurred.
RepositoryLock	Stats repository lock.
RepositoryNList	Number of stats in status repository.
ReReads	Pages that were re-read from disk multiple times.
SingleThrAlloc	Number of teams that had only one thread.
StCntxCondVar	Resource gate lock.
StCntxLock	Database context lock.
StCntxNAdmitted	Total cost of completed commands.
StCntxNOrigResource	Maximum number of concurrent commands.
StCntxNResource	Number of available concurrent commands.
StCntxNWaited	Number of commands that waited at resource gate in monitor interval .
StCntxNWaiting	Number of commands currently blocked at resource gate.
StCntxNumConns	Current number of connections.
TeamThrAlloc	Total number of thread teams allocated in monotor interval.
ThrMaxTeams	Maximum number of thread teams.

ThrMutexLocks	Number times thread manager was locked.
ThrMutexWaits	Number of thread manager locks that waited.
ThrNTeamsInUse	Number of thread teams currently in use..
ThrNumFree	Number of threads available for assignment.
ThrNumOfCpus	Number of CPUs IQ is using.
ThrNumThreads	Actual threads allocated by server.
ThrReserved	Description.
ThreadLimit	Total number of threads requested for server.
TimesSweepers Woken	Number of times sweeper threads waited for work.
TxnMgrNBlocked	Number of transactions blocked in begin transaction.
TxnMgrNPending	Number of transactions with pending commit.
TxnMgrNWaiting	Number of transactions waiting in begin transaction.
TxnMgrOAVI	Oldest version in use.
TxnMgrPCcondvar	Transaction manager post commit lock statistics.
TxnMgrTxnIDseq	Next Transaction ID.
TxnMgrVersionID	Transaction manager share main version ID.
TxnMgrVersionLock	Lock used for version ID assignment.
TxnMgrtxncblock	Lock used during transactional commands.
UnOwnRR	Unowned Re-read. Number of pages re-read on a query server because they were potentially stale.
UsedPerActiveCmd	Average number of threads per command.
WashMaxSize	Maximum number of buffers can be in the wash area.
Washed	Number of buffers flushed from wash area.
washIntensity	Current value of Buffer_Cache_Wash_Intensity option.
washNActive Sweepers	Number of sweepers with buffers to write.
washNBuffers	Number of clean buffers in the wash area.
washNDirtyBuffers	Number of dirty buffers in the wash area.
washSignalThreshold	Number of dirty buffers in wash area before sweepers are woken.

washTeamSize	Number of threads in the sweeper team.
Writes	Pages written to disk.

Page Types

BTREEV	Variable B-Tree. Used by HG, LF for varchar, varbinary datatypes.
BTREEF	Fixed B-Tree. Used by HG, LF, WD, DT, TM, DTTM indexes for fixed datatypes. Table VDO to store index metadata.
BV	Persistent Bit Vector. Used by Join bloom filters and range predicate processing.
VDO	Variable Sized Data Object. Used by Table VDO to store index metadata.
DBEXT	DBSpace Header Block. Used by First block of each dbspace.
DBID	Database Identity. Used by One block per database.
SORT	Sort data. Used by Sort objects in the temp cache.
STORE	Temporary data storage. Used by Scrolling cursor store. Temporary query storage: union, cursors, OLAP.
GARRAY	Group Array structure. Used by HG, WD index, Join Index.
BARRAY	Byte Array. Used by FP and LOB index.
BARRAY	Blockmap. Used by Metadata for all objects.
HASH	Hash Table data. Used by Hash Objects in the temp cache.
CKPT	Checkpoint Log. Used by Checkpoint log entries in the main store.
BM	Bitmap Data. Used by Indexes, query processing and database structures.
CMID	Multiplex Commit Identity. Used by Multiplex to notify query servers of new versions.
RIDCA	Recid Cache Data. Used by Used by sp_iqcheckdb in temp only.
LOB	Long Binary Storage. Used by Long varchar and long binary storage.

Lock Statistics

Locks	Number of times the lock obtained.
Lock-Waits	Number of times the lock waited because of contention.

Signals	Number of times the condition variable was signaled to schedule a single thread.
Broadcasts	Number of times the condition variable was signaled to schedule all threads.
Waits	Number of times a thread the waiting on the condition variable.
RdLocks	Number of times that a read lock was acquired from a readwrite mutex.
RdWaits	Number of times that a read lock waiting on a readwrite mutex.
RdTryFails	Number of times that a read trylock failed to acquire a readwrite mutex.
WrLocks	Number of times that a write lock waiting on a readwrite mutex.
WrWaits	Number of times that a write lock waiting on a readwrite mutex.
WrTryFails	Number of times that a write trylock failed to acquire a readwrite mutex.
Locks	Number of times that a lock was acquired from a spinlock.
SpinsWoTO	Number of times that a spinlock timed out waiting for the lock.
Spins	Number of times that a spinlock spun waiting for the lock.
TimeOuts	Number of times that a spinlock timed out waiting for the lock.

Buffer Allocator/Prefetch Client Types

BM	Bitmap . Used by: Most database structures.
BTree	B-Tree. Used by: HG, LF for varchar, varbinary datatypes .
BV	(Semi) Persistent Bit Vector. Used by: Join bloom filters and range predicate processing.
DBCC	Database Consistency Checker. Used by: sp_iqcheckdb.
FP	Fast Projection Index. Used by: Primary Column Storage.
Garray	Group Array. Used by: HG, WD index, Join Index.
Hash	Hash Tables. Used by: Queries, Enumerated FP indexes.
LOB	Long Binary. Used by: Long Binary Columns.
Row	FP Row Fetch. Used by: Query Row projection.
RowColumn	FP Column Fetch.. Used by: Query Column projection.
Sort	Sort. Used by: Queries, Insert, Update, Mid Delete, DBCC.

Store	Temporary data storage. Used by: Scrolling cursor store. Temporary query storage: union, cursors, OLAP.

Index

A

address space
- virtual 140

AGGREGATION_ALGORITHM_ PREFERENCE option 47

alphabetical order 4

AND keyword 8

apostrophes
- using 7

AVG function 11

B

backups
- tuning block size 143

BETWEEN conditions 10

BLANK PADDING
- effect on joins 30
- support of OFF 30

block size
- relationship to IQ page size 65

buffer allocator 151

Buffer Allocator/Prefetch Client Types
- BM 161
- BTree 161
- BV 161
- DBCC 161
- FP 161
- Garray 161
- Hash 161
- LOB 161
- Row 161
- RowColumn 161
- Sort 161
- statistics 161
- Store 162

buffer cache monitor 115
- examples 123

buffer caches 63, 141
- determining sizes 59
- layout 127
- monitoring 115
- setting sizes 63

buffer manager
- statistics 147
- thrashing 129

buffer pool
- statistics 149

buffers
- disabling operating system buffering 70

C

cache
- buffer 141
- NTFS 141
- *See Also* buffer cache 115

cache pages
- prefetching 83

cache size
- IQ main and temporary buffers 63

cache statistics 102

case sensitivity 3, 7

Catalog Store 5
- file growth 89

Catalog, db log, and repository statistics 154

CIS functional compensation
- performance impact 41

clearing procedure profiling
- SQL 109
- Sybase Central 109

columns
- about 3, 4
- ordering 5
- selecting from a table 5
- significant number of null values 6, 36

commands

long 7
comparisons
about 6, 7
conditions
and GROUP BY clause 12
search 6, 7, 10
connection statistics 103
connections
limiting statements used by 83
context server statistics 153
correlation names
about 23
defined 32
COUNT function 11, 12
CPU time statistics 153
CPU usage
monitoring 131, 136
CPU usage statistics 101
CPUs
availability to Sybase IQ 81
CREATE DBSPACE statement 77
cursors
limiting number of 83

D

data
storage 141
data loading improvements 93
data types
requirements for joins 28
database options
FLATTEN_SUBQUERIES 40
database procedures
viewing profiling data 107
database segments
locating for best performance 77
databases
benefits of denormalizing 90
denormalizing for performance 89
managing 87
dates 7, 10
dbspaces
locating for best performance 77
usage statistics 105
DEFAULT_HAVING_SELECTIVITY option 47
DEFAULT_LIKE_MATCH_SELECTIVITY option 47
DEFAULT_LIKE_RANGE_SELECTIVITY option 47
denormalization
disadvantages 90
performance benefits 90
reasons for 89
DENSE_RANK function 17
devices
I/O on Windows 143
direct I/O 70
disabling procedure profiling
SQL 109
Sybase Central 109
disk cache
definition 85
disk caching
performance impact 85
disk space
multiplex databases 84
swap space 57
disk striping
definition 74
internal 76
rules 75
Sybase IQ 74
use in loads 76
documentation
SQL Anywhere x
Sybase IQ ix
dummy Sybase IQ table 5
dynamic performance monitor 98

E

EARLY_PREDICATE_EXECUTION option 47
enabling procedure profiling
SQL 107
Sybase Central 107
events
viewing individual profiling information 112
viewing profiling data 107
viewing summary profiling data 111

F

files
  locating for best performance 77
fixed-length I/O
  Windows 143
FLATTEN_SUBQUERIES option 40
flattening subqueries 40
FORCE_NO_SCROLL_CURSORS 82
fragmentation 69
FROM clause 5, 92
  and joins 22
functions
  aggregate 11
  analytical 17
  SOUNDEX function 10
  viewing individual profiling information 112
  viewing profiling data 107
  viewing summary profiling data 111

G

GROUP BY
  performance recommendation 36
grouped data 11

H

HASH_THRASHING_PERCENT option 129
heap
  low-fragmentation 69
HG indexes
  multicolumn 39
hyperthreading
  server switch 81

I

I/O
  direct 70
  performance recommendations 73
I/O devices 143
IN conditions 10
IN_SUBQUERY_PREFERENCE option 47
index advisor 86
index types
  choosing for performance 86
INDEX_ADVISOR option 42
INDEX_PREFERENCE option 48
indexes
  multicolumn 39
inequality, testing for 7
insert operations
  tuning 142
IQ page size
  determining 64
IQ PATH option
  choosing a raw device 74
IQ Store
  buffer cache size 63
IQ store free list 150
iq_dummy table 5
iqgovern switch
  restricting queries to improve performance 81
IQGOVERN_MAX_PRIORITY option 46
IQGOVERN_PRIORITY 46
IQMSG log
  setting maximum size 79
iqnumbercpus
  setting number of CPUs 81
iqwmem switch 69

J

join indexes
  performance impact 87
JOIN_ALGORITHM_PREFERENCE option 48
joins
  and BLANK PADDING 30
  and data types 28
  cross product 22
  optimizer simplications 48
joins between databases
  performance impact 41

K

key joins

using 25

L

lightweight processes 72
LIST function 11
load balancing
  among query servers 85
load improvements 93
loading data
  access during loads 93
  HG and WD indexes 93
  parallel loads 93
  tuning for 142
  using striped disk 76
Lock Statistics
  Broadcasts 161
  Locks 160, 161
  Lock-Waits 160
  RdLocks 161
  RdTryFails 161
  RdWaits 161
  Signals 161
  Spins 161
  SpinsWoTO 161
  TimeOuts 161
  Waits 161
  WrLocks 161
  WrTryFails 161
  WrWaits 161
lock statistics
  Performance statistics 160
  sp_iqsysmon 160
low-fragmentation heap 69

M

main database
  buffer cache size 63
MAIN_CACHE_MEMORY_MB option 63
MAX function 11
MAX_CURSOR_COUNT option 83
MAX_HASH_ROWS option 48
MAX_QUERY_TIME option 46
MAX_STATEMENT_COUNT option 83
MAX_TEMP_SPACE_PER_CONNECTION option 81
maximum function 11
memory
  fragmentation 69
  multiplex databases 58
  overhead 60
  paging 57
  reducing requirements 66
  restricting use by queries 81
  See Also buffer caches 63
  wired 69
memory manager 152
memory usage statistics 101
message log
  Sybase IQ 79
MIN function 11
monitor
  IQ UTILITIES syntax 116
  setting output file location 116
  starting and stopping 116
monitoring
  page faults 141
  physical memory 139
  transaction status 98
  virtual address space 140
  working set 140
monitoring workloads 49
multicolumn indexes 39
multiplex
  performance monitor 98
multiplex databases
  disk space 84
  memory 58
multithreading
  performance impact 72

N

network statistics 106
networks
  large transfers 94
  performance suggestions 94
  settings 94

NOEXEC option 42
NT CreateFile option 142
NT Performance Monitor 139
NT Task Manager 139
NTFS cache
  improving performance 141
  stealing memory 138
NTILE function 17

O

optimizing queries 39, 86
option value
  truncation 42
options
  FLATTEN_SUBQUERIES 40
  INDEX_ADVISOR 42
  NOEXEC 42
  QUERY_DETAIL 43
  QUERY_PLAN 43
  QUERY_PLAN_AFTER_RUN 43
  QUERY_PLAN_AS_HTML 43
  QUERY_PLAN_AS_HTML_DIRECTORY 43
  QUERY_PLAN_TEXT_ACCESS 43
  QUERY_PLAN_TEXT_CACHING 43
  QUERY_TIMING 44
  unexpected behavior 5
OR keyword 10
ORDER BY clause 7, 39
OS_FILE_CACHE_BUFFERING option 70
OS_FILE_CACHE_BUFFERING_TEMPDB option 70
outer references
  defined 32
OVER clause 18

P

page faults 138
  monitoring 141
Page Types
  BARRAY 160
  BM 160
  BTREEF 160
  BTREEV 160
  BV 160
  CKPT 160
  CMID 160
  DBEXT 160
  DBID 160
  GARRAY 160
  HASH 160
  LOB 160
  RIDCA 160
  SORT 160
  STORE 160
  VDO 160
page types
  sp_iqsysmon 160
pages
  decompressing 141
paging
  effect on performance 57
  memory 57
  monitoring on UNIX 131
  monitoring on Windows 130
parallelism
  query tree 39
parameters
  to functions 11
partitioned table 92
partitions
  definition 74
pattern matching 9
PERCENT_RANK function 17
PERCENTILE_CONT function 17
PERCENTILE_DISC function 17
performance 137
  balancing I/O 73
  benefits of denormalizing databases 90
  choosing correct index type 86
  CIS functional compensation impact 41
  definition 56
  designing for 56
  disk caching 85
  dynamic monitor 98
  improving 13
  monitoring 115
  multi-user 83
  restricting queries with iqgovern 81

subqueries 40
performance and tuning issues 143
performance monitor
2-D bar chart 99
3-D bar chart 99
customizing chart 99
customizing statistics 99
examples 123
printing chart 99
saving chart 99
server level 99
statistics monitored 100
switching chart view 99
time-series view 99
Performance statistics
buffer allocator, defined 151
buffer allocator, statistics 151
Buffer Allocator/Prefetch Client Types 161
buffer manager, defined 147
buffer manager, statistics 147
buffer pool, defined 149
buffer pool, statistics 149
catalog, db log, and repository statistics 154
catalog, db log, and repository statistics, defined 154
context server statistics 153
context server statistics, defined 153
CPU time statistics 153
CPU time statistics, defined 153
field name links 146
field reference links 146
IQ store free list, defined 150
IQ store free list, statistics 150
lock statistics 160
memory manager, defined 152
memory manager, statistics 152
prefetch manager, defined 150
prefetch manager, statistics 150
section name links 146
section statistic definition links 146
sp_iqsysmon 145
sp_iqsysmon, defined 145
statistics type definition links 146
statistics, list of 154
thread manager, defined 152
thread manager, statistics 152
transaction manager, defined 153
transaction manager, statistics 153
performance tuning
introduction 98
physical memory
monitoring 139
prefetch manager 150
PREFETCH_BUFFER_LIMIT option 83
Prefetched cache pages 83
procedure profiling
clearing in Sybase Central 109
clearing with SQL 109
disabling in Sybase Central 109
disabling with SQL 109
enabling in Sybase Central 107
enabling with SQL 107
events 111, 112
information for individual procedures 112, 115
resetting in Sybase Central 108
resetting with SQL 108
stored procedures and functions 111, 112
summary of procedures 113
system triggers 111, 113
triggers 111, 113
viewing data in Interactive SQL 113
viewing data in Sybase Central 107, 110
process threading model 72
processes
growth 69
monitoring 139
working sets 140
processing queries without 5
profiling database procedures
about 107
profiling information
events 112
stored procedures and functions 112
system triggers 113
triggers 113
pushdown join 92

Q

queries
indexing recommendations 86
optimizing 42, 47, 86

processing by SQL Anywhere 5
restricting concurrent 81
restricting memory use 81
structuring 36
tuning for 143
query parallelism 39
query performance
Catalog Store tables 41
CIS functional compensation impact 41
cross-database joins 41
processing by SQL Anywhere rules 41
query plans 42
generating without executing 42
graphical 44
using 44
query server
balancing loads 85
QUERY_DETAIL option 43
QUERY_PLAN option 43
QUERY_PLAN_AFTER_RUN option 43
QUERY_PLAN_AS_HTML option 43
QUERY_PLAN_AS_HTML_DIRECTORY option 43
QUERY_PLAN_TEXT_ACCESS option 43
QUERY_PLAN_TEXT_CACHING option 43
QUERY_TEMP_SPACE_LIMIT option 81
QUERY_TIMING option 44
querying tables 5
quotation marks
using 7

R

range 7
RANK function 17
raw devices
effect on performance 74
raw partitions
memory use 61
RAWDETECT
disk striping option 76
request statistics 104
resetting procedure profiling
SQL 108
Sybase Central 108
response time 56
restrictions 6
rows
about 3
selecting 6

S

search conditions
date comparisons 7
introduction to 6
short cuts for 10
subqueries 31
segments
database, using multiple 77
SELECT statement
about 2, 31
sequential disk I/O 77
servers
monitoring performance 99
sp_iqcolumnuse procedure
sp_iqindexuse procedure 49
sp_iqsysmon
buffer allocator, defined 151
buffer allocator, statistics 151
Buffer Allocator/Prefetch Client Types 161
buffer manager, defined 147
buffer manager, statistics 147
buffer pool, defined 149
buffer pool, statistics 149
catalog, db log, and repository statistics 154
catalog, db log, and repository statistics, defined 154
context server statistics 153
context server statistics, defined 153
CPU time statistics 153
CPU time statistics, defined 153
field name links 146
field reference links 146
IQ store free list, defined 150
IQ store free list, statistics 150
lock statistics 160
memory manager, defined 152
memory manager, statistics 152
page types 160

prefetch manager, defined 150
prefetch manager, statistics 150
section name links 146
section statistic definition links 146
statistics 154
statistics type definition links 146
thread manager, defined 152
thread manager, statistics 152
transaction manager, defined 153
transaction manager, statistics 153
sp_iqtable procedure 2
sp_iqtableuse procedure 49
sp_iqunusedcolumn procedure 49
sp_iqunusedindex procedure 49
sp_iqunusedtable procedure 49
sp_iqworkmon procedure 49
Statistics
BlockmapMutexsNLocks 154
BlockmapMutexsNWaits 154
BlockmapRegEver 154
BlockmapRegisters 154
BlockmapUID 154
BlockmapUIDnallocs 154
BmapHTMaxEntries 154
BmapHTNClears 154
BmapHTNCollisn 154
BmapHTNEntries 154
BmapHTNFinds 154
BmapHTNHits 154
BmapHTNHits1 154
BmapHTNHits2 154
BmapHTNInserts 154
BmapHTNLChain 154
BmapHTNRehash 154
BmapHTNumLocks 154
BmapHTNumWaits 154
BufAllocAvailBufs 155
BufAllocAvailPF 155
BufAllocMaxBufs 155
BufAllocMutexLocks 155
BufAllocMutexWaits 155
BufAllocNPFUsers 155
BufAllocNPinUsers 155
BufAllocNPostedUsrs 155
BufAllocNPostEst 155
BufAllocNUnPostEst 155
BufAllocNUnpostUsrs 155
BufAllocPinQuota 155
BufAllocReserved 155
BufAllocSlots 155
BufHTMaxBucketSize 155
BufHTNBuckets 155
BufHTNEntries 155
BufHTNFoiledOps 155
BufHTNw2orMore 155
BusyWait 155
CacheTeamNumAsleep 155
CacheTeamTimesWoken 155
CatalogLock 155
ClientCountOfPinners 155
Cloned 155
CPU Sys Seconds 156
CPU Total Seconds 156
CPU User Seconds 155
Creates 156
DbLogMLock 156
DbLogSLock 156
Destroys 156
Dirties 156
Dirty 156
Elapsed 156
FalseMiss 156
Finds 156
FLBitCount 156
FLIsOutOfSpace 156
FLMutexLocks 156
FLMutexWaits 156
FlushedBufferCount 156
Flushes 156
GetPageFrame 156
GetPageFrameFailure 156
GotEmptyFrame 156
GrabbedDirty 156
HitRate 156
InUse 156
IONumLocks 156
IONumWaits 156
LRUNumLocks 156
LRUNumSpinLoops 156
LRUNumSpinsWoTO 156
LRUNumTimeOuts 157
MemAllocated 157

MemAllocatedEver 157
MemAllocatedMax 157
MemNAllocated 157
MemNAllocatedEver 157
MemNTimesLocked 157
MemNTimesWaited 157
MovedToMRU 157
MovedToWash 157
NActiveCommands 157
NumClients 157
NumTeamsAlloc 157
NumThrUsed 157
Pages 157
PFMgrCondVar 157
PFMgrNDropped 157
PFMgrNRead 157
PFMgrNReading 157
PFMgrNSubmitted 157
PFMgrNThreads 157
PFMgrNValid 157
PfUserRegisters 157
Pinned 157
PinUserQuota 158
PinUserRegisters 158
PReadBlks 158
PReadKB 158
PrefetchInMem 157
PrefetchNotInMem 157
PrefetchReqs 158
PrefetchUserQuota 158
PWriteBlks 158
PWriteKB 158
Realdirties 158
RemovedFromLRU 158
RemovedFromWash 158
RemovedInScanMode 158
RepositoryLock 158
RepositoryNList 158
ReReads 158
SingleThrAlloc 158
StCntxCondVar 158
StCntxLock 158
StCntxNAdmitted 158
StCntxNOrigResource 158
StCntxNResource 158
StCntxNumConns 158
StCntxNWaited 158
StCntxNWaiting 158
TeamThrAlloc 158
ThreadLimit 159
ThrMaxTeams 158
ThrMutexLocks 159
ThrMutexWaits 159
ThrNTeamsInUse 159
ThrNumFree 159
ThrNumOfCpus 159
ThrNumThreads 159
ThrReserved 159
TimesSweepersWoken 159
TxnMgrNBlocked 159
TxnMgrNPending 159
TxnMgrNWaiting 159
TxnMgrOAVI 159
TxnMgrPCcondvar 159
TxnMgrtxncblock 159
TxnMgrTxnIDseq 159
TxnMgrVersionID 159
TxnMgrVersionLock 159
UnOwnRR 159
UsedPerActiveCmd 159
Washed 159
washIntensity 159
WashMaxSize 159
washNActiveSweepers 159
washNBuffers 159
washNDirtyBuffers 159
washSignalThreshold 159
washTeamSize 160
Writes 160
statistics
dynamic 98
Performance statistics 154
sp_iqsysmon 154
store I/O statistics 105
stored procedures 2
performance monitoring 106
viewing individual profiling information 112
viewing profiling data 107
viewing summary profiling data 111
subqueries
using 31
subquery flattening 40

subquery performance 40
SUM function 11
summary profiling data
  events 111
  stored procedures and functions 111
  system triggers 111
  triggers 111
swap files
  effect on performance 57
swapping
  disk space requirement 57
  effect on performance 57
  memory 57
sweeper threads 127
SYSTEM dbspace 5
system procedures
  sp_iqcolumnuse 49
  sp_iqindexuse 49
  sp_iqtableuse 49
  sp_iqunusedcolumn 49
  sp_iqunusedindex 49
  sp_iqunusedtable 49
  sp_iqworkmon 49
system stored procedures 2
system triggers
  viewing individual profiling information 113
  viewing profiling data 107
  viewing summary profiling data 111

T

tables
  and foreign keys 24
  collapsing 87
  correlation name 23
  iq_dummy 5
  joining 87
  listing 2
  primary keys 24
tape devices
  Windows 143
Task Manager 139
TEMP_CACHE_MEMORY_MB option 63
Temporary Store
  buffer cache size 63
thrashing
  actions to take 129
  HASH_THRASHING_PERCENT option 129
thread manager 152
thread statistics 103
threads
  buffer caches 127
  controlling use with -iqnumbercpus switch 81
  management options 73
  monitoring 120
throughput 56
  maximizing 138
transaction log
  about 78
  offline database 78
  stopped database 78
  truncating 78
transaction manager 153
transaction statistics 104
transaction status
  monitoring 98
triggers
  viewing individual profiling information 113
  viewing summary profiling data 111
tuning 137
  for insert operations 142
  for queries 143

U

UNION ALL
  rules 92
  views 92
user-defined functions
  performance impact 41
user-supplied conditions
  for queries 48

V

viewing procedure profiling data
  Sybase Central 110
viewing procedure profiling information in Interactive SQL 113

viewing profiling information for a specific procedure
        in Interactive SQL 115
virtual address space 140
virtual memory
    fragmentation 69
vmstat command
    monitoring buffer caches on UNIX 131

W

WHERE clause
    and pattern matching 9
    BETWEEN conditions 10
    date comparisons 7
    examples 6
    ORDER BY clause 7
Windows tape devices 143
wired memory 69
WITHIN GROUP clause 18
working set 140
workload monitoring 49