



**Introduction to Sybase IQ**

---

**Sybase IQ 15.4**

DOCUMENT ID: DC38159-01-1540-02

LAST REVISED: May 2012

Copyright © 2012 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Java and all Java-based marks are trademarks or registered trademarks of Oracle and/or its affiliates in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

# Contents

<b>About Sybase IQ .....</b>	<b>1</b>
Related Products .....	3
Documentation .....	3
Architecture .....	5
Benefits .....	6
Components .....	6
Tools and Utilities .....	6
Windows Options .....	7
Database Server .....	8
Demo Database .....	8
<b>Concepts .....</b>	<b>11</b>
Servers .....	11
Database Computing .....	11
Sybase IQ Database Files .....	12
How the Pieces Fit Together .....	13
SQL and Database Computing .....	13
Relational Databases .....	13
Database Tables .....	14
Primary and Foreign Keys .....	15
Other Database Objects .....	16
Queries .....	16
Other SQL Statements .....	17
System Tables .....	17
<b>Running and Connecting to Servers .....</b>	<b>19</b>
Windows Client Connections .....	19
UNIX Client Connections .....	20
Sybase Central and Database Management .....	21
The Sybase Central Interface .....	21
Running the Sybase IQ Agent .....	22
Navigating the Main Sybase Central Window .....	23
Viewing a Database Schema .....	23

Viewing Other Database Objects .....	24
Connecting to Databases With Interactive SQL .....	26
Starting Interactive SQL from Sybase Central .....	26
Starting Interactive SQL from the Command Line .....	26
Starting Interactive SQL on Windows .....	27
Sending Commands To the Database .....	27
Shutting Down the Database Server .....	27
<b>Selecting Data .....</b>	<b>29</b>
Viewing Table Information .....	29
Ordering Query Results .....	30
Selecting Columns and Rows .....	31
Using Search Conditions .....	32
Comparing Dates in Queries .....	33
Compound Search Conditions in the WHERE Clause .....	34
Pattern Matching in Search Conditions .....	35
Matching Rows by Sound .....	36
Using Shortcuts for Search Conditions .....	36
Obtaining Aggregate Data .....	37
Using Aggregate Functions to Obtain Grouped Data .....	37
Restricting Groups .....	38
Improving Subtotal Calculation .....	39
ROLLUP .....	39
Cube .....	41
Obtaining Analytical Data .....	42
Rank Analytical Functions .....	42
Inverse Distribution Functions .....	43
Eliminating Duplicate Rows .....	43
<b>Joining Tables .....</b>	<b>45</b>
Cross Product Joins .....	45
Restricting Joins .....	45
Table Relationships .....	46
Join Operators .....	47

Key Joins .....	47
Natural Joins .....	48
Using Ad Hoc Joins vs. Join Indexes .....	49
Joins and Data Types .....	50
Join Support Between Stores or Databases .....	51
Querying Remote and Heterogeneous Databases .....	52
Joining Remote Databases .....	52
Joins and Subqueries .....	52
<b>Managing Databases .....</b>	<b>55</b>
Managing Tables .....	55
Creating an Office Table .....	55
Creating Columns For the Office Table .....	56
Editing Column Properties .....	56
Deleting the Office Table From the Demo Database .....	57
Managing Procedures .....	57
Viewing Stored Procedures .....	57
Managing Users and Groups .....	58
Adding a New Login Policy .....	59
Adding a Group To a Database .....	59
Adding a User to the Demo Database .....	59
Restoring the Demo Database to its Original State .....	60
Managing User Connections .....	60
<b>Indexing and Loading Data .....</b>	<b>61</b>
Creating Column Indexes .....	61
Column Index Types .....	61
Adding a Low Fast Index To a Column .....	62
Creating Join Indexes .....	63
Creating a Join Index for SalesOrders and SalesOrderItems .....	64
Creating a Join Index for Department and Employee .....	64
Updating Join Index Data .....	65
Loading Data .....	65

<b>Managing Dbspaces .....</b>	<b>67</b>
Creating Dbspaces .....	67
Altering Dbspaces .....	69
<b>Updating Databases .....</b>	<b>71</b>
Adding Rows To a Table .....	71
Making Changes Permanent .....	72
Ensuring Data Consistency .....	72
Savepoints .....	73
Testing Data Consistency .....	73
Setting Savepoints .....	74
Updating Multiple Tables From Multiple Accounts .....	74
Deleting Rows .....	75
Updating Rows .....	76
Synchronizing Join Indexes .....	76
Deleting dbspaces, Tables, and Indexes .....	76
<b>Using Views .....</b>	<b>79</b>
Defining a View .....	79
Creating a View .....	80
Displaying View Data .....	80
Changing a View .....	80
Deleting Views .....	81
View Restrictions and Advanced Uses .....	81
Using Views for Security .....	82
Restricting Access to Information In the Database .....	83
<b>Glossary .....</b>	<b>85</b>
<b>Index .....</b>	<b>89</b>

## About Sybase IQ

Sybase® IQ is a high-performance decision-support server designed specifically for data warehousing.

Sybase IQ is part of the Sybase product family that includes Adaptive Server Enterprise and SQL Anywhere. Component Integration Services within Sybase IQ provide direct access to relational and nonrelational databases on mainframe, UNIX, or Windows servers.

### *Product Editions*

Sybase IQ is available in several product editions.

<b>Product Edition</b>	<b>Description</b>
Enterprise Edition	Supports all options and features.  Optional features require additional licenses that are purchased separately.
Small Business Edition	Supports most of the functionality available in the Enterprise Edition, but does not support multiplex or any optional features.
Single Application Server	Provides the same functionality as the Small Business Edition, but is restricted to use in a one- or two-machine environment.
Evaluation Edition	Supports all options and features available in the Enterprise Edition for a 30-day evaluation period.
Express Edition	Supports all options and features available in the Evaluation Edition.  Unlike the Evaluation Edition, the Express Edition does not expire, but does restrict the size of all IQ main dbspaces, permit other incremental license options, and cannot be deployed in a production environment.

### *Optional Features*

Optional features extend the utility of the Enterprise Edition.

Option	Description
Unstructured Data Analytics	<p>Supports Binary Large Object (BLOB) and Character Large Object (CLOB) storage and retrieval.</p> <p>Supports Binary Large Object (BLOB) and Character Large Object (CLOB) storage and retrieval.</p>
Advanced Security	<p>Supports these optional security features:</p> <ul style="list-style-type: none"> <li>• Column encryption</li> <li>• FIPS network encryption</li> <li>• Kerberos connection authentication</li> </ul>
Multiplex Grid	<p>Lets you add additional nodes to a multiplex environment.</p> <p>Sybase IQ requires an additional license to start secondary multiplex nodes (readers/writers).</p>
Very Large Database Management	<p>Lets you add multiple table spaces and dbspaces to logically partition data into manageable subsets.</p> <p>Sybase IQ requires an additional license when the server creates or starts with two or more IQ user dbspaces.</p>
Express Edition	<p>Supports all options and features available in the Evaluation Edition.</p> <p>Unlike the Evaluation Edition, the Express Edition does not expire, but does restrict the size of all IQ main dbspaces, permit other incremental license options, and cannot be deployed in a production environment.</p>

*Licensing*

All product editions except the Evaluation Edition require a license. Optional features are sold and licensed separately.

The Evaluation Edition is an unlicensed server, which provides full access to all features and options available in the Enterprise Edition. To run the Evaluation Edition beyond the thirty-day evaluation period, however, you must purchase and install the appropriate license.



## Related Products

---

Sybase IQ is part of a large group of Sybase products that provide a complete analytics infrastructure.

- SQL Anywhere® – SQL Anywhere is a comprehensive data management package for server, desktop, mobile, and remote office environments. Sybase IQ is an extension of SQL Anywhere, and incorporates many of its features.  
 See *Products > Database Management > SQL Anywhere* on the *Sybase Web* site for more information.
- Sybase RAP – The Trading Edition® is an analytics platform for capital markets. Part of the underlying technology of Sybase RAP is provided by Adaptive Server Enterprise and Sybase IQ .  
 See *Products > Financial Services Solutions > RAP - The Trading Edition* on the *Sybase Web* site for more information.
- Sybase PowerDesigner® – PowerDesigner is the industry-leading modeling and metadata management solution for data, information, and enterprise architecture.  
 See *Products > Modeling & Development > PowerDesigner* on the *Sybase Web* site for more information.
- Sybase IQ InfoPrimer – Sybase IQ InfoPrimer is a data extraction, transformation, and load platform that can quickly and efficiently move your data from multiple sources into a Sybase IQ reporting environment.  
 See *Products > All Products A-Z > Sybase ETL* on the *Sybase Web* site for more information.
- Sybase WorkSpace – WorkSpace integrates the most important design and development tools in an easy-to-use open-source framework.  
 See *Products > Modeling & Development > WorkSpace* on the *Sybase Web* site for more information.

## Documentation

---

Refer to this summary to locate information about specific topics.

### *New Release Information*

Document Name	Contents
Installation and Configuration	Installation and configuration procedures.
Release Bulletins	Late breaking product information.
Guide to Licensed Options	Features you can buy separately.

Document Name	Contents
New Features Summary	Release-specific feature summaries.

*Getting Started*

Document Name	Contents
Introduction to Sybase IQ	Hands-on introduction to Sybase IQ.
Sybase IQ Quick Start	Steps to create and query an IQ demo database.
Utility Guide	Command-line utility reference.
Using Sybase IQ Multiplex	Multiplex setup and administration.
Performance and Tuning Guide	Database, system, and query tuning options.

*Reference*

Document Name	Contents
System Administration Guide: Volume 1	Database setup and administration.
System Administration Guide: Volume 2	Database application development.
Reference: Statements and Options	SQL syntax, parameters, and options.
Reference: Building Blocks, Tables, and Procedures	SQL statements, functions, and procedures.
Error Messages	Error and warning messages (Web only).

*Optional Features*

Document Name	Contents
Unstructured Data Analytics	Binary Large Object (BLOB) and Character Large Object (CLOB) storage and retrieval.
User-Defined Functions Guide	C/C++ interface for user-defined functions.
Using Sybase IQ Multiplex	Multiplex setup and administration.
Advanced Security	Advanced Sybase IQ security options.
IMSL Numerical Library User's Guide: Volume 2 of 2 C Stat Library	IMSL C Stat library functions.
Time Series Guide	Time series forecast and analysis functions.

*SQL Anywhere Reference*

Document Name	Contents
SQL Anywhere Server Database Administration	Additional administrative details referenced by the Sybase IQ documentation.
SQL Anywhere Server Programming	API programming documentation and examples.
SQL Anywhere Server SQL Reference	Additional SQL functionality referenced by the Sybase IQ documentation.

*Licensing*

Document Name	Contents
Sybase Software Asset Management (SySAM) 2	License generation, options, and management.
SySAM 2 Quick Start Guide	SPDC and SMP license generation.
FLEXnet Licensing End User Guide	FLEXnet Licensing utilities.

## Architecture

---

Sybase IQ architecture differs from most relational databases. Sybase IQ focuses on readers, not writers, which provides a fast query response for many users.

- Data is stored in columns, not rows
- Placing indexes on all columns provides a performance advantage
- A large page size provides a performance advantage
- A large temporary cache provides a performance advantage for most operations
- Access to data occurs at the table level
- Most query results focus on data at the table level
- Most insertions and deletions write data for an entire table, not for a single row.

For details about setting up Sybase IQ for optimal performance, see *System Administration Guide: Volume 1*, *System Administration Guide: Volume 2* and *Performance and Tuning Guide*.

## Benefits

---

Sybase IQ is a decision support system optimized to deliver superior performance for mission-critical business solutions.

- Intelligent query processing that use index-only access plans to process any type of query.
- Ad hoc query performance on uniprocessor and parallel systems.
- Multiplex capability for managing large query loads in a multi-server configuration.
- Fully-flexible schema support.
- Efficient query execution without query-specific tuning under most circumstances.
- Fast initial and incremental loading.
- Fast aggregations, counts, comparisons of data.
- Parallel processing optimized for multi-user environments.
- Stored procedures.
- Increased productivity due to reduced query time.
- Entire database and indexing stored in less space than raw data.
- Reduced input/output (I/O).

## Components

---

Sybase IQ includes tools and utilities that help you work with the server.

### Tools and Utilities

---

Sybase IQ includes utilities that help you perform routine management tasks.

**Table 1. Sybase IQ Utilities.**

Utility	Description
Sybase Control Center	Sybase Control Center (SCC) is a Web-based administrative console for Sybase products. At least one SCC server is required to deploy Sybase Control Center across the enterprise.  See the Sybase Control Center Documentation for more information.
Sybase Central	Sybase Central is an administrative console for Sybase products. The Sybase IQ Plug-in is required to administer tasks for Sybase IQ.  Most tutorials and procedures in this book use Sybase Central.
Interactive SQL	Interactive SQL is a tool included with Sybase IQ that lets you execute SQL statements, build scripts, and display database data.  See the <i>Utility Guide</i> for more information about Interactive SQL.

Utility	Description
Command line utilities	Command-line utilities perform database administration tasks. See the <i>Utility Guide</i> for more information.

---

**Note:** Available utilities depend on your operating system and installation options.

---

## Windows Options

On Windows, the Sybase IQ Program Group provides quick access to common utilities. Available options differ for Sybase IQ Server and Sybase IQ Client installations.

### *Server Installations*

**Table 2. Sybase IQ Server Options.**

Option	Description
Interactive SQL	Starts the Java-based Interactive SQL utility for sending SQL statements to a database.
ODBC Administrator	Manages ODBC connections to your databases for 32-bit or 64-bit platforms.
Raw Device Access	Adds or removes users and raw devices from the registry.
Sample Applications and Projects	Displays a list of sample applications and projects. For information on running these samples, see <i>SQL Anywhere Server – Programming</i> .
Start Sybase IQ Demo Database	Starts the database server running the demo database.
Sybase Central	Starts the Sybase Central.
Sybase IQ ReadMe File	Lists last-minute changes to documentation and software and special material that needs emphasis.
Sybase Service Manager	Lets you configure, modify, or delete Sybase IQ as a Windows service.
Sybase on the Web	Opens a Web page that introduces Sybase online resources.

*Client Installations*

**Table 3. Sybase IQ Client Options.**

Option	Description
Sample Applications and Projects	Displays a list of sample applications and projects. For information on running these samples, see <i>SQL Anywhere Server – Programming</i> .
Sybase on the Web	Opens a Web page that introduces Sybase online resources.
ODBC Administrator	Manages ODBC connections to your databases for 32-bit or 64-bit platforms.
Interactive SQL	Starts the Java-based Interactive SQL utility for sending SQL statements to a database.

**Database Server**

Sybase IQ network server supports client/server communications across a network and multiple users.

To start the server, run the appropriate startup utility for your operating system:

- On UNIX or Linux operating systems, use `start_iq`.
- On Windows, use `start_iq.exe`.

**Demo Database**

Many of the examples throughout the documentation use the Sybase IQ demo database (`iqdemo.db`) as a data source. The demo database is installed as part of the Sybase IQ Server Suite and resides on the server. Client access is provided by tools installed as part of the Sybase IQ Client Suite.

**Table Names and Owners**

The demo database includes data and utility tables.

Tables in the demo database are delineated by ownership. Most of the SQL examples in this document require access to `GROUPO` tables as a minimum.

*GROUPO Tables*

`GROUPO` tables contain internal information about a fictional company that sells athletic clothing. Sample data includes information about the company (employees, departments, and financial data) as well as product information (products) and sales information (sales orders, customers, and contacts).

**Table 4. GroupO Table Names.**

Name	Description
Contacts	Customer contacts and sales leads.
Customers	Customer names and addresses.
Departments	Company departments, such as manager and name.
Employees	Employee information, such as name, salary, and location.
FinancialCodes	Each expense and revenue item has a financial code.
FinancialData	Quarter-by-quarter financial information about the company.
Products	Product information, such as price and quantity available.
SalesOrderItems	Sales order items. Each order consists of one or more items. Information about sales order items is held in a separate table.
SalesOrders	Individual sale orders, including customer ID, OrderDate, Financial-Code, Region, and SalesRepresentative.

**DBA Tables**

Tables owned by the DBA include utility and sample data tables. Access to DBA tables requires DBA Authority.

**Table 5. DBA Table Names.**

Table	Description
iq_dummy	<p><code>iq_dummy</code> is a one-row, one-column utility table that you can use to extract information from the database. For example, running the <b>NOW()</b> function against <code>iq_dummy</code> returns the current date and time:</p> <pre>SELECT NOW() FROM iq_dummy</pre> <p>Use of the DUMMY system table in Sybase IQ is implied for all queries that do not have a FROM clause. For more information, see System Administration Guide: Volume 1 &gt; Dummy Tables for Performance Monitoring.</p>
empl	Sample employee table that includes <code>dept_id</code> , <code>start_date</code> , <code>name</code> , and <code>salary</code> columns.
sale	Sample sales table that includes <code>prod_id</code> , <code>month_num</code> , <code>rep_id</code> , and <code>sales</code> columns.

*SYSOPTION - DEFAULTS Table*

SYSOPTIONDEFAULTS is a utility table owned by DBO that contains all Sybase IQ option names and values. You can query this table to see all default option values.

---

**Note:** The demo database is case insensitive. This means that case is not considered in comparison and string operations. For example, you can type user IDs and passwords in either uppercase or lowercase when using the demo database. Note that, unlike the demo database, any Sybase IQ databases you create are case sensitive by default.

---



# Concepts

Understanding some basic terms and concepts will help you work with Sybase IQ.

## Servers

---

Access to your database must be made through a server, which provides a communications channel and a manipulation device for the database.

A Sybase IQ server can dynamically start and stop a database, and accept connections from applications or users running on the same machine or on other computers by means of the network that links the two machines. Users can have connection rights to a database, not to the server. Sybase strongly recommends that you manage only one database for each server.

You can use multiple Sybase IQ servers to combine the use of Sybase IQ, SQL Anywhere, and Enterprise databases. You can also use multiple Adaptive Server Enterprise servers to combine the use of Sybase IQ and Adaptive Server Enterprise databases.

A Sybase IQ multiplex is an Sybase IQ database spread across a shared disk array. Each server in a multiplex runs Sybase IQ. Each set of an Sybase IQ Temporary Store and Catalog Store make up one server, and the servers share a common Sybase IQ Store. Users with large query loads and shared disk arrays may create a Sybase IQ multiplex to gain additional CPU power and memory space. A Sybase IQ multiplex supports many users, each executing complex queries against the shared database.

You can manage all Sybase IQ servers using Sybase Central.

## Database Computing

---

Database applications and the database server work together to manage databases.

Any information system contains the following pieces:

- A database - data is stored in a database.
- Database server - manages the database. No other applications address the database file directly; they all communicate with the database server.
- Language interface - applications communicate with the database server using an interface. You can use ODBC, JDBC, Sybase Open Client, or Embedded SQL.

The language interface provides a set of function calls available to client applications for communicating with the database. For ODBC and JDBC, the library is commonly called a driver. The interface is typically provided as a shared library on UNIX operating systems or a dynamic link library (DLL) on Windows operating systems. The JDBC interface uses the Sybase jConnect driver, which is a zip file of compiled Java classes.

## Concepts

If you are working with a Sybase IQ network server, the language interface resides on the client computer.

- Client applications - use one of the language interfaces to communicate with the database server.

If you develop an application using a rapid application development (RAD) tool such as one of the Sybase Enterprise Application Studio™ tools, you may find that the tool provides its own methods for communicating with database servers, and hides the details of the language interface. Nevertheless, all applications do use one of the supported language interfaces.

## Sybase IQ Database Files

When you create a database, Sybase IQ creates a number of files.

**Table 6. Database files**

Default physical file name	Internal name	Description	Number created
dbname.db	Not applicable	SQL Anywhere dbspace file. Part of the catalog store.	One or more per server
dbname.log	Not applicable	SQL Anywhere transaction log. Part of the catalog store.	One for each server
dbname.iq	IQ_SYSTEM_MAIN	The initial dbfile file for the IQ main store. (User specifies the names for additional files).	One or more for each database
dbname.iqtmp	IQ_SYSTEM_TEMP	The initial IQ temporary dbfile file.	One or more for each database
dbname.iqmsg	IQ_SYSTEM_MSG	IQ human-readable trace file of debugging output.	One for each database
dbname.lmp	Not applicable	IQ human-readable license file.	One for each database

---

**Note:** The DBA can override the default file names and extensions shown.

---

You can create a database using a relative or fully qualified pathname for each of the files for the database. If you omit the directory path, Sybase IQ creates the files as follows:

- The catalog store is created relative to the working directory of the server.
- The IQ store is created relative to the working directory of the server.
- The temporary store is created in the same directory as the catalog store. (This also occurs if you do not specify any file name.)
- The message log is created in the same directory as the catalog store. (This also occurs if you do not specify any file name.)

- The transaction log is created in the same directory as the catalog store. (This also occurs if you do not specify any file name.)

---

**Note:** For best performance, place the transaction log on a different physical device from the catalog store and IQ store, because they are randomly accessed files and the transaction log is a sequentially accessed file.

---

The main and temporary stores contain most of your tabular data. Each database has its own unique files for temporary data. If you don't specify a file, Sybase IQ creates one automatically for use as a temporary storage space.

## **How the Pieces Fit Together**

A complete application in a client/server environment includes a database server with one or more client applications.

The database server may be a network server, which supports network communications. No changes are needed to the client application for it to work in a client/server environment.

## **SQL and Database Computing**

Client applications use Structured Query Language (SQL) statements to carry out database tasks, such as retrieving information or inserting rows into tables.

Depending how a client application is developed, SQL statements could be supplied in function calls from the programming language, or you may build them graphically in a special window provided by the application development tool.

The programming interface delivers SQL statements to the database server. The database server executes them and returns results to the application.

Client/server communication protocols carry information between the client application and the database server. Programming interfaces define how an application sends the information. No matter which interface and network protocol you use, SQL statements are sent to a server, and the results of SQL statements return to the client application.

---

**Note:** There are slight differences in the SQL (Structured Query Language) syntax supported by Sybase IQ, SQL Anywhere, and Adaptive Server Enterprise. For an overview of SQL compatibility, see *Reference: Building Blocks, Tables, and Procedures > Compatibility with Other Sybase Databases*.

---

## **Relational Databases**

A relational database management system stores and retrieves data in tabular format. A relational database consists of a collection of tables that store interrelated data.

## **Database Tables**

In a relational database, all data is contained in tables, which are made up of rows and columns.

Each table has one or more columns, and each column is assigned a specific data type, such as an integer number, a sequence of characters (for text), or a date. Each row in the table has at most one value for each column. Where there is no value for a particular row and column, we sometimes say that the value is NULL, which may be interpreted as not currently known, or not applicable, or not yet available.

An example of a table containing employee information may appear as follows:

<b>emp_ID</b>	<b>emp_lname</b>	<b>emp_fname</b>	<b>emp_phone</b>
10057	Huong	Zhang	1096
10693	Donaldson	Anne	7821

### **Characteristics of Relational Tables**

The tables of a relational database have some important characteristics:

- There is no logical significance to the order of the columns or rows. However, the column order does affect a few special queries. Likewise, the row order does affect the order in which some queries return the row, and can affect the performance of queries. If you care about the order of the rows, then the query should specify the order in which the rows are to be returned. If a particular order is very common and all other orders are uncommon, it may be best to keep the rows in that particular order.
- Each row contains either no value (a NULL column) or contains one and only one value for each column.
- Each value for a given column is of the same type.

The following table lists some of the formal and informal relational database terms describing tables and their contents, together with their equivalent term in other nonrelational databases. This manual uses the informal terms.

<b>Formal relational term</b>	<b>Informal relational term</b>	<b>Equivalent nonrelational term</b>
Relation	Table	File
Attribute	Column	Field
Tuple	Row	Record

### *What Do You Keep in Each Table?*

When you design a database, make sure that each table in the database holds information about a specific thing, such as employees, products, or customers.

A relational database is not only a set of unrelated tables. You can use primary and foreign keys to describe relationships between the information in different tables.

## Primary and Foreign Keys

Primary and foreign keys enable each row in the database tables to be identified, and enable relationships between the tables to be defined. These keys define the relational structure of a database.

### *Primary Keys*

Each table in a relational database may or may not have a primary key. The primary key is a column, or set of columns, that allows each row in the table to be uniquely identified. No two rows may have the same value of a primary key.

You can select a primary key from all of the available columns. Keep your primary key for each table as compact as possible. If possible, the primary key should be an unsigned integer data type, rather than `char` or `varchar`.

For example, the `SalesOrderItems` table in the demo database includes these columns:

- An `ID` column that identifies the customer's order.
- A `LineID` column that provides an identification number for each item of the sales order.
- A `ProductID` column that identifies the product that the customer ordered.
- A `Quantity` column that shows how many items were ordered.
- A `ShipDate` column that identifies the date the order shipped.

To identify a particular item, both the `ID` and the line `LineID` are required. The primary key is made up of both these columns.

### *Foreign Keys*

The information in one table is related to that in other tables by foreign keys.

For example, the demo database has one table holding employee information and one table holding department information. The `Departments` table has these columns:

- `DepartmentID` – An ID number that identifies the department. This is the primary key for the table.
- `DepartmentName` – A column holding the name of the department.
- `DepartmentHeadID` – The employee ID for the department manager.

To find the name of a particular employee's department, there is no need to put the name of the employee's department into the `Employees` table. Instead, the `Employees` table contains a column holding the employee's department ID. This is called a foreign key to the `Departments` table. A foreign key references a particular row in the table containing the corresponding primary key or unique constraint. The primary key and unique constraint so referenced are known as the candidate key.

In this example, the `Employee` table (which contains the foreign key in the relationship) is called the foreign table or referencing table. The `Department` table (which contains the referenced primary key) is called the primary table or the referenced table.

## Other Database Objects

A relational database holds more than a set of related tables.

Objects that make up a relational database include:

Object	Description
Indexes	<p>Indexes allow quick lookup of information. In a database, the index relates each indexed column value to the physical location at which the row of data containing the indexed value is stored.</p> <p>Indexes are an important design element for high performance.</p>
Java Objects	<p>You can install Java classes into the catalog store of your database. Java classes provide a way of building logic into your database.</p> <p>For details about Java data type support in Sybase IQ, see <i>System Administration Guide: Volume 2</i>.</p>
Procedures & Functions	<p>These are routines held in the database itself that act on the information in the database.</p> <p>You can create and name your own stored procedures to execute specific database queries and to perform other database tasks. Stored procedures can take parameters. For example, you might create a stored procedure that returns the names of all customers who have spent more than an amount you specify as a parameter in the call to the procedure.</p>
Users & Groups	<p>Each user of a database has a user ID and password. You can set permissions for each user, so that confidential information is kept private. Users can be assigned to groups, in order to make the administration of permissions easier.</p>
Views	<p>Views are computed tables, or virtual tables. They look like tables to client applications, but they do not hold data. Instead, whenever they are accessed, the information in them is computed from the underlying tables.</p> <p>The tables that actually hold the information are sometimes called base tables to distinguish them from views.</p>

## Queries

Basic query operations in a relational system are selection, projection, and join. The SELECT statement implements all of them.

## **Projections and Restrictions**

A projection is a subset of the columns in a table. A restriction (also called selection) is a subset of the rows in a table, based on some conditions.

For example, the following **SELECT** statement retrieves the names and prices of all products that cost more than fifteen dollars:

```
SELECT Name, UniPrice
   FROM Products
  WHERE UnitPrice > 15
```

This query uses both a restriction (**WHERE** unit\_price > 15) and a projection (**SELECT** name, unit\_price)

## **Joins**

A join links the rows in two or more tables by comparing the values in key columns and returning rows that have matching values. For example, this query joins the SalesOrderItems, Products, Employees, and SalesOrders tables to calculate sales totals and revenue:

```
SELECT Products.ID AS 'Product Code',
       Products.Name AS Item,
       Products.Description AS Style,
       SUM(SalesOrderItems.Quantity) AS Sales,
       Products.UnitPrice,
       SUM(SalesOrderItems.Quantity * Products.UnitPrice)
       AS Revenue FROM Employees
 JOIN SalesOrders ON SalesOrders.SalesRepresentative =
   Employees.EmployeeID
 JOIN SalesOrderItems ON SalesOrderItems.ID =
   SalesOrders.ID
 JOIN Products ON Products.ID =
   SalesOrderItems.ProductID
 GROUP BY Products.ID, Products.Description,
   Products.Name, Products.UnitPrice
 ORDER BY Sales
```

## **Other SQL Statements**

You can do more with SQL than just query. SQL includes statements that create tables, views, and other database objects. It also includes statements that modify tables (the insert and delete statements), and commands that perform many other database tasks discussed in this manual.

## **System Tables**

Every database contains a set of system tables, which are special tables used by the system to manage data and the system. These tables are also sometimes called the data dictionary or the system catalog. In Sybase IQ they are stored in the catalog store.

System tables contain information about the database. You never alter the system tables directly in the way you can alter other tables. The system tables hold information about the

## Concepts

tables in a database, the users of a database, the columns in each table, and so on. This information is data about data, or metadata. You can use the system tables to check the size and data type for various columns before inserting data.



# Running and Connecting to Servers

Tasks in the tutorial require access to Sybase Central and a client connection to the demo database. Using Sybase Central to establish a client connection depends on whether you use the Windows or UNIX client tools.

## Windows Client Connections

---

To run the sample queries on Windows, start the Sybase Central network client, and connect to the demo database.

### Prerequisites

Ask your DBA or System Administrator to:

- Install the Sybase IQ Client Suite on your workstation.
- Create and start the demo database. The demo database is installed as part of the Sybase IQ Server Suite. Instructions for starting `iqdemo.db` appear in the *Installation and Configuration Guide*.
- Create an account that allows you to access the demo database. To recreate the sample queries, your account must include the authority to access GROUPO tables.

### Task

1. Click **Start > Programs > Sybase > Sybase Central**.
2. Close the **Tips** and **Welcome** windows, if necessary.
3. Click **Connections > Connect with Sybase IQ 15**.
4. On the **Connection** dialog, enter:

Field	Value
Authentication	Database
User ID	DBA or dba (case-insensitive)
Password	sql (case-sensitive)
Action	Connect to a running database on another computer

5. From the **Server Name** box, click **Find**.
6. Choose `<localhost>_iqdemo` from the Servers dialog.

This document uses the placeholder, **<localhost>** to identify the host. When you see this placeholder in any instruction, substitute your **host** name.

7. Click **Connect**.

## UNIX Client Connections

---

To run the sample queries on UNIX-like operating systems, start the Sybase Central network client, and connect to the demo database.

### Prerequisites

Ask your DBA or system administrator to:

- Install the Sybase IQ Client Suite on your workstation. Make sure that your environment is set properly.
- Create and start the demo database. The demo database is installed as part of the Sybase IQ Server Suite. Instructions for starting `iqdemo.db` appear in the *Installation and Configuration Guide*.
- Create an account that allows you to access the demo database. To recreate the sample queries, your account must include the authority to access GROUPO tables.

### Task

1. From a console, change to `$IQDIR15/bin64`, enter `scjview`.
2. Close the **Tips** and **Welcome** windows, if necessary.
3. Click **Connections > Connect with Sybase IQ 15**.
4. On the **Connection** dialog, enter.

Field	Value
Authentication	Database
User ID	DBA or dba (case-insensitive)
Password	sql (case-sensitive)
Action	Connect to a running database on another computer

5. From the **Server Name** box, click **Find**.
6. Choose `<localhost>_iqdemo` from the Servers dialog.

This document uses the placeholder, **<localhost>** to identify the host. When you see this placeholder in any instruction, substitute your **host** name.
7. Click **Connect**.

## Sybase Central and Database Management

Sybase Central is a database management tool installed with Sybase IQ. You can use Sybase Central to perform typical database administration tasks in a graphical user interface.

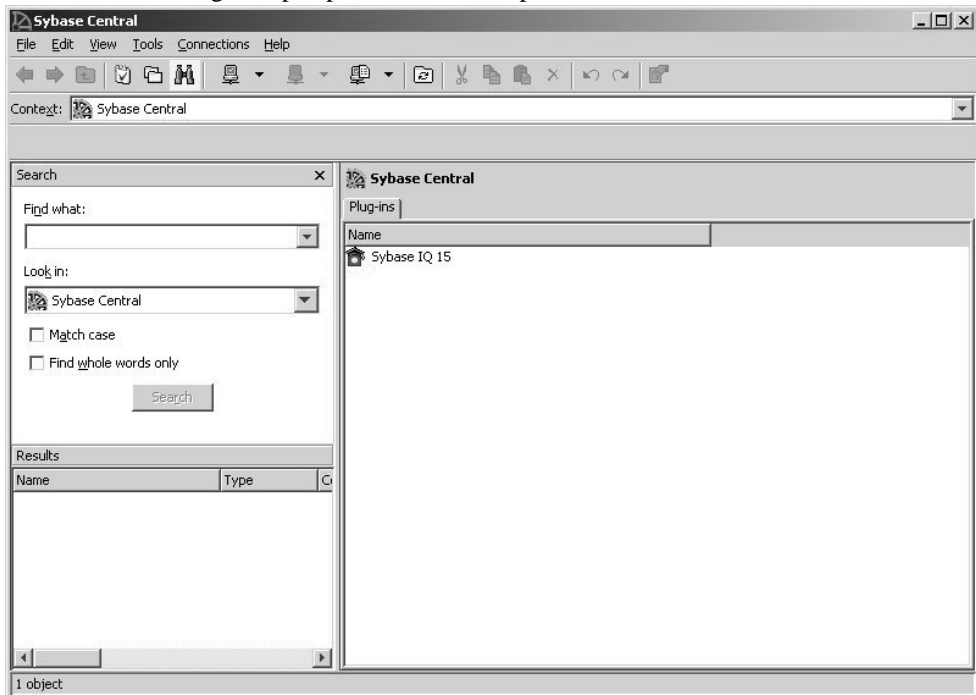
Database administration tasks typically fall into two categories:

- Tasks carried out by sending SQL statements to the database engine.
- Tasks carried out by Sybase IQ utilities.

You can also use Interactive SQL to perform many of the same data definition tasks that you can carry out with Sybase Central. Some tasks, however, like creating multiplex databases and query servers, are best performed with Sybase Central.

### The Sybase Central Interface

The main Sybase Central window is split into two vertically-aligned panels. The left pane displays tasks, folders, or search options. The right pane displays additional options, actions, or views associated with the object or task you select. You can use the options on the Toolbar or View menu to change the perspective of the left pane.



### Loading a Plug-in

Installing a Sybase product also installs its plug-in. The Sybase IQ plug-in should load automatically. If you do not see the plug-in, you can load it manually.

1. From the Tools menu, choose Plug-ins.
2. Choose Sybase IQ, click Register.

If Sybase IQ is not listed, click **Load > Browse**, choose *\$IQDIR15/java/iq.jpr*.

### Stopping Sybase Central

Stopping Sybase Central closes the main window.  
To stop Sybase Central, click File, choose Exit.

## Running the Sybase IQ Agent

The Sybase IQ Agent must be running to do many multiplex operations (such as creating query servers) on Sybase Central. The Sybase IQ Agent generally is started whenever you reboot a Windows machine where Sybase IQ is installed. Before you administer a multiplex, verify that the Sybase IQ Agent is running.

### Checking the Sybase IQ Agent on Windows

Use the Services console to see status of the Sybase IQ Agent.

1. On the Control Panel, choose Administrative Tools | Services.
2. On the Services console, check the status of the Sybase IQ Agent.
3. If Status does not say Started, right-click the agent, choose Start.

### Checking the Sybase IQ Agent on UNIX

Use the **-agent** option to check the status of Sybase IQ Agent.

1. Run the **stop\_iq** utility with the **-agent** option:

```
stop_iq -agent
```

2. If no running agent is owned by your username, change to *\$IQDIR15/bin<platform>* and run **S99SybaseIQAgent15**.

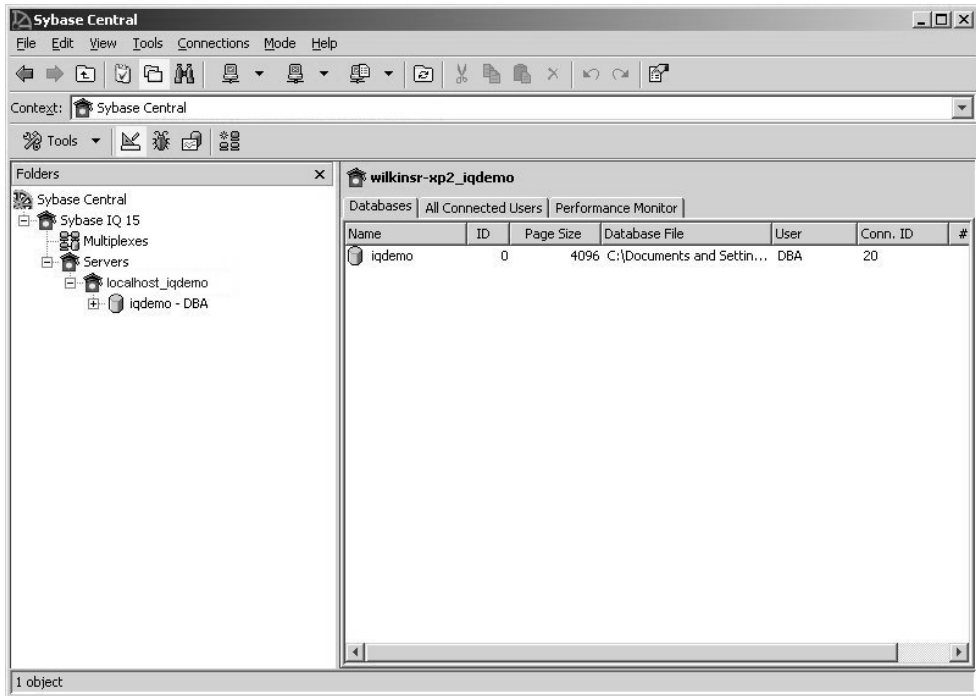
- To start the agent using the host name returned by **uname -n** use the optional **-host** parameter:

```
S99SybaseIQAgent15 -host
```

- Start the agent with the host's alias: **S99SybaseIQAgent15 -host <foo>**  
where *foo* is an alias present in the */etc/hosts* file.

## Navigating the Main Sybase Central Window

Follow the steps in this tutorial to view tables, columns, indexes, joined tables and users in the database.



## Viewing a Database Schema

A database schema is the collection of all objects in the database. Sybase Central displays a database schema as a hierarchy of containers and their contents.

There are several ways to expand a database container and view its contents.

- In Tasks view, the Contents tab in the right pane displays a folder view of database objects. To perform a particular task, double-click the object in the right pane and a set of related tasks appear in the left pane. Click the task you want to perform from the task list.
- In Folders view, database objects appear in a hierarchical tree in the left pane. Clicking a branch of the tree expands or contracts the view; details about the object appear in the right panel. For example, if you select the Sybase IQ plug-in from the left panel, the right panel displays servers, utilities, and services for Sybase IQ.
- The Search pane lets you search for items in the object hierarchy. To search for an item, type the name of the object in the Find what box, choose an object class from the Look in

## Running and Connecting to Servers

drop down, then click Search. Click on an object in the Results panel to display the object in the right pane.

**Note:** If you cannot display certain fields, use the facilities your windowing system provides to change the Sybase Central display to use dark text on a white or light background.

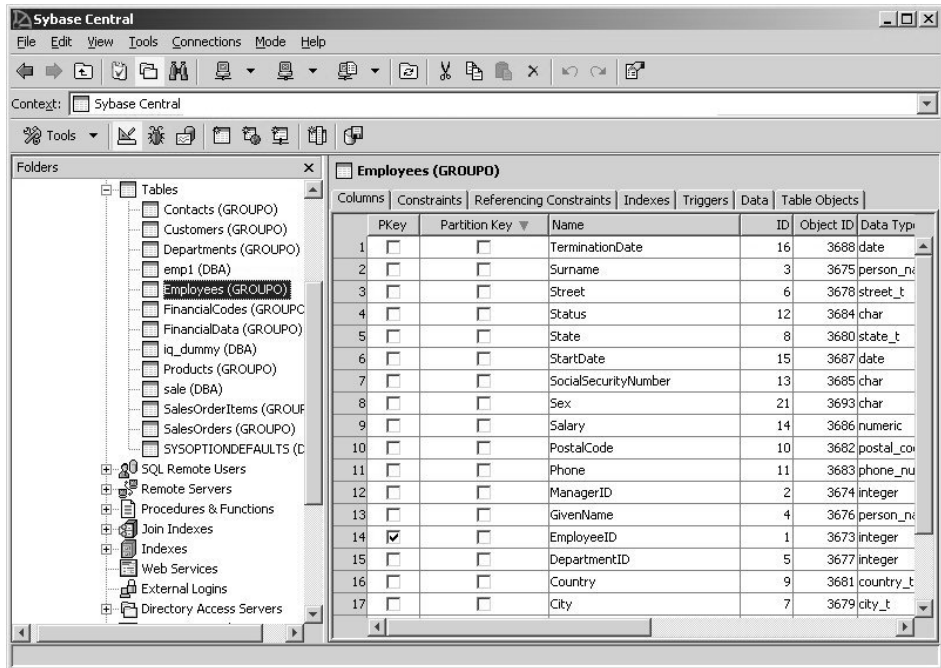
### Viewing Tables in a Database

The Tables folder lets you access the database tables.

1. Connect to a database, click the Tables folder, then click the table to see the column definitions in the right panel.

The right panel includes tabs that expose tables's columns, Constraints, Referencing Constraints, Indexes, Each table object contains folders for columns, foreign keys, relations, indexes, and more.

2. Click on a tab in the right panel to expose the table's Columns, Constraints, Referencing Constraints, Indexes, Triggers, Data, and Table Objects.



### Viewing Other Database Objects

Every Sybase IQ sever contains a database folder.

Object	Description
Dbspaces	A logical collection of dbfiles within a database.

Object	Description
Directory Access Servers	Gives you access to the local file system of the computer running the database server. You can create proxy tables for the directory access server to access the files and directories on the server computer.
Domains	Sometimes called user-defined data types, domains are aliases for built-in data types.
Events	Automate database administration, such as a stored procedure scheduled to execute out at a certain time.
External Environments	Refer to programming languages, utilities, and objects external to the database server.
External Logins	Provide access to a directory on a remote or directory access server.
Indexes	Structures that store and represent data for query processing. Indexes allow quick lookup of information. In a database, the index relates each indexed column value to the physical location at which the row of data containing the indexed value is stored.
Join Indexes	Joins that are created on sets of columns rather than individual columns, and represents a full outer join of two or more tables.
Login Mappings	Map a Windows user profile or Kerberos principal to an existing database user.
Login Policies	Set of options that define rules to be applied when a user connects to a database.
Procedures & Functions	SQL procedures for a module-based language.
Publications	Identify replicated data in SQL Remote. In a Sybase IQ database, publications can only be created on SQL Anywhere tables.
Remote Servers	Give you access to data located on separate database servers.
SQL Remote Subscriptions	Creates links from a publisher to a remote database (subscriber).
SQL Remote Users	Define remote databases in a SQL remote setup.
System Triggers	Events that modify the system tables.
Tables	Base tables stored in the database.
Triggers	Execute a SQL statement or procedure automatically when someone modifies the data.
Users & Groups	Mechanisms that identify database users and privileges.
Views	Computed tables, stored in the database as a query and evaluated when accessed.

Object	Description
Web Services	Describe how to process HTML and XML requests made to the web server.

### **Disconnecting From a Database Server**

Disconnection options log you off the database server. If you are connected to more than one server, you can disconnect from a specific server. If only one database is connected, it is automatically disconnected.

Do one of the following:

- From the Connections menu, choose Disconnect.
- Click the Disconnect button on the Toolbar.
- Right-click the database container, and choose Disconnect.

## **Connecting to Databases With Interactive SQL**

Interactive SQL is a tool included with Sybase IQ that lets you execute SQL statements, build scripts, and display database data. Depending on your operating system, you can start Interactive SQL from the command line or as an application in Windows.

### **Starting Interactive SQL from Sybase Central**

Start Interactive SQL from Sybase Central.

1. In the left pane, select the Sybase IQ plug-in and do one of the following:
  - Choose **Tools > Sybase IQ 15 > Open Interactive SQL**.
  - Right-click a database, and choose **Open Interactive SQL**.
  - Right-click a stored procedure, and choose **Execute from Interactive SQL**.
2. In the **Connect** window, supply parameters.

### **Starting Interactive SQL from the Command Line**

You can start Interactive SQL from the command line and to connect to a database.

From a console or command line, enter:

```
dbisql -c "uid=DBA;pwd=sql" -host <host name> -port 2638
```

The **-c** parameter specifies connection parameters. For a complete list of connection parameters, see *System Administration Guide: Volume 1 > Connection and Communication Parameters*.

---

**Note:** The default user ID and password for the iqdemo database and other Sybase IQ databases are *DBA* and *sql*. If you have trouble connecting to a database, try changing the case of the user ID and password that you supply; the case sensitivity for that database may be set ON.

---



## Starting Interactive SQL on Windows

On Windows, you can start Interactive SQL from the Sybase IQ Program Group.

1. Click **Start > Programs > Sybase > Sybase IQ > Interactive SQL**.
2. On the Identification tab, enter DBA in the User box, and sql in the Password box.  
This is the default user ID and password for Sybase IQ databases. DBA is not case-sensitive; sql is case-sensitive, and must appear in all lowercase letters.
3. On the Database tab, choose a server from the Server name box. Click Find to search for running servers.  
Server names appear in this format *<hostname>\_dbname*. If the host name is *localhost-xp* and you are running the sample *iqdemo* database, the server name would be *localhost-xp\_iqdemo*. If the server is remote, type *host:port* instead of the server name.
4. Choose a database from the Database name box.  
When there is only one database running, you can leave this field blank.
5. What you do next depends on the server location:
  - To run Interactive SQL against a server installed on the same machine, you must also choose an appropriate Database file.
  - To run Interactive SQL against a network server, click the Network tab, click the TCP/IP box, then specify the Host name and Port number.
6. Click OK.

Other applications connect to the database server in much the same way. To make the connection, they must submit both user ID and password. The *Utility Guide* gives more details about how to use Interactive SQL.

## Sending Commands To the Database

After you connect, you can send commands to the database.

For example, type the following line in the SQL Statements window:

```
SELECT * FROM SalesOrders
```

Click the right-facing triangle button on the Toolbar or press F9 to execute the query. The SQL statement displays all (\*) the columns from the SalesOrders table.

## Shutting Down the Database Server

Use Sybase Central or Interactive SQL to stop a server.

Do one of the following:

To stop a server with...	Do this...
<b>Sybase Central</b>	<ul style="list-style-type: none"> <li>• From the File menu, choose Stop server.</li> <li>• Right-click the server, and choose Stop Server.</li> </ul>
<b>Interactive SQL</b>	<p>Use the <b>STOP ENGINE</b> statement.</p> <p>In Interactive SQL, run:</p> <pre>STOP ENGINE [database-server-name] [UNCONDITIONALLY]</pre> <p>The STOP ENGINE statement stops the specified database server. If you supply the UNCONDITIONALLY keyword, the database server is stopped even if there are other connections to the database server. By default, the database server will not be stopped if there are other connections to it.</p>

# Selecting Data

This section is designed to help you become familiar with the construction and design of basic queries. Most topics focus on basic single-table **SELECT** statements and include sample statements that you can use to practice writing queries.

All of the examples in this section use the Sybase IQ demo database (iqdemo.db) as a data source. To run the queries in this section, start the demo database, open Interactive SQL, and connect to iqdemo.

## Viewing Table Information

---

Queries retrieve information from the database. All queries include a **SELECT** statement and can contain additional clauses that determine what data the query returns. Query results are stored in a result table called the result set.

### *Listing Tables*

You can use a stored procedure to list the tables in the demo database. To list the tables, enter

```
sp_iqtable
```

in the SQL Statements window.

### *Using the SELECT Statement*

Use a **SELECT** statement to retrieve data from a database table. To retrieve all rows and columns from the Employees table, enter:

```
SELECT * FROM Employees
```

The asterisk in the SQL statement is a wildcard that indicates you want to retrieve all the columns in the table. The Interactive SQL Results window displays these results:

Employ- eeID	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	...

EmployeeID	ManagerID	Surname	GivenName	DepartmentID	...
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:** Tables 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 demo database is case insensitive. This means that case is not considered in comparison and string operations. The `Employees` table name, for example, 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 section were created case insensitive, using the **CREATE DATABASE** qualifier **CASE IGNORE**. The default is **CASE RESPECT**, which gives better performance.

---

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.

## Ordering Query Results

---

Sybase IQ does not return the rows in a result set in any particular order. Adding an **ORDER BY** clause to a **SELECT** statement returns the results in alphabetical or numerical order.

### Listing Employees in Alphabetical Order

Adding an **ORDER BY** clause to a **SELECT** statement returns the result set in alphabetical or numeric order. The order of the clauses is important. The **ORDER BY** clause must follow the **FROM** clause and the **SELECT** clause. For example:

```
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	...
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	...

**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*

This example selects 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

## Selecting Data

Surname	DepartmentID	BirthDate
Chin	200	1966-10-30
Jordan	300	1951-12-13
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

---

Adding a **WHERE** clause to a **SELECT** statement retrieves only those records that meet a specific condition. .

### *Selecting Records That Meet a Specific Condition*

Sometimes you will not want to see information on all the employees in the **Employees** table. To look at employees with the first name, John, you would enter:

```
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	Leticq	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
```

### *Ordering Clauses*

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.

### *Splitting Lines*

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**

---

Use a comparison 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

Using comparison operators such as greater than (>), greater than or equal (>=), less than or equal (<=), and not equal (<>) with **AND** and **OR** creates more complex 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

---

Use the **LIKE** operator in a **WHERE** clause to search for patterns in a column.

### *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.

### *Searching Surnames by Sound*

Suppose a phone message was left for a name that sounded like “Brown.” Which employees in the company have names that sound like Brown? 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

---

Use the **BETWEEN** and **IN** operators in a **WHERE** clause to retrieve multiple values.

### *Using BETWEEN*

The **BETWEEN** operator is used in a **WHERE** clause to select a range of data between two 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 IN*

Use the **IN** operator to specify multiple values in a **WHERE** clause. 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

Aggregate functions perform calculations on a set of values rather than on a single value.

### *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

Use aggregate functions on groups of rows to provide information about an entire table.

### *Using an Aggregate Function on Groups of Rows*

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

## Selecting Data

```
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

Use the **HAVING** function with the **GROUP BY** function to retrieve aggregate values that meet specific conditions.

### *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

SalesRepresentative	count( * )
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. **ROLLUP** and **CUBE** are extension to the **GROUP BY** operator that allow you to use one query to compute data using multiple levels of grouping, instead of a separate query for each level.

### **ROLLUP**

**ROLLUP** is an extension to the **GROUP BY** operator that calculates multiple levels of subtotals across a specified group of dimensions, and calculates a grand total, **ROLLUP**. Subtotals “roll up” from the most detailed level to the grand total.

#### *Summarizing Product Inventory*

Suppose you want to summarize inventory in the Products table by name, size, and color:

## Selecting Data

```
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 (name, size, color), then for all grouping expressions except the last one (color). In the third row, NULL indicates the **ROLLUP** value for the color column.

**ROLLUP** requires an ordered list of grouping expressions as arguments. When listing groups that contain other groups, list the larger group first. 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.

## Cube

CUBE is an extension to the GROUP BY operator that analyzes data by forming the data into groups in more than one dimension.

CUBE requires an ordered list of grouping expressions (dimensions) as arguments and enables the SELECT statement to calculate subtotals for all possible combinations of the group of dimensions.

### *Calculating Average Salaries*

The following query uses data from the Employees table to calculate the average salary by state, gender, and department.

```
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;
```

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

c_state	c_gender	c_dept	COUNT()	AVERAGE
CA	M	ALL	1	39,300.00

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.

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..

## Obtaining Analytical Data

---

Analytical functions manipulate data within result sets. Rank analytical functions rank items in a group. Inverse distribution analytical functions return a k-th percentile value.

### Rank Analytical Functions

Rank analytical functions rank items in a group, compute distribution, and divide a result set into a number of groupings. Rank analytical functions include: **RANK**, **DENSE\_RANK**, **PERCENT\_RANK**, and **NTILE**. The inverse distribution analytical functions are **PERCENTILE\_CONT** and **PERCENTILE\_DISC**.

#### *Ranking Employee Salaries*

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



Name	Salary	Ranking
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 an 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.

## Inverse Distribution Functions

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.

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.

## Eliminating Duplicate Rows

Use the **DISTINCT** keyword to eliminate the duplicate rows.

### *Selecting DISTINCT Rows*

For example, the following command returns many duplicate rows:

## Selecting Data

```
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.

---

# Joining Tables

Joins let you construct queries that retrieve data from more than one table.

## Cross Product Joins

---

A cross product join returns a result set that produces all possible combinations of rows from the two tables. The number of rows in the result set is the product of the number of rows in the first table and the number of rows in the second table.

### *Example*

The `FinancialData` and `FinancialCodes` in the sample database contain financial data for a fictitious company. Both tables have a `code` column that you can use to join the two tables.

The following **SELECT** statement lists all the data in the `FinancialCodes` and `FinancialData` tables:

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

The result set matches 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. More selective joins apply restrictions to the cross product table.

## Restricting Joins

---

Adding conditions to a query restricts the result set to records that meet specific criteria. Join conditions use comparison operators to eliminate rows from the cross-product result.

### *Example 1*

The **WHERE** clause in this query specifies that the `SalesRepresentative` in the `SalesOrders` table match the `EmployeeID` in the `Employees` table. This creates a result set where each row contains information about an order and the sales representative responsible for it:

```
SELECT *  
FROM SalesOrders, Employees  
WHERE SalesOrders.SalesRepresentative = Employees.EmployeeID
```

## Joining Tables

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.

---

## Table Relationships

---

Understanding how one table is related to another lets you construct different types of joins.

### *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 *System Administration Guide: Volume 1 > Data Integrity > Data Integrity Overview > Data Integrity Tools*.)

## Join Operators

---

Joins are used to query two or more tables based on the relationship between columns in these tables.

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.

## Key Joins

A key join generates a join condition based on the foreign key relationships in the database.

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
```

## Joining Tables

```
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.

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.

## Natural Joins

The **NATURAL JOIN** operator joins two tables based on common column names. This means that 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

---

A join index is an internal structure that defines a relationship between the columns in two or more tables. Ad hoc joins are joins that do not use 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 *System Administration Guide: Volume 1 > Sybase IQ Indexes*.)

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.

## Joining 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

---

Understanding Sybase IQ handles join column data types can help you write more efficient queries.

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 than 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 “Moving Data In and Out of Databases” in System Administration Guide: Volume 1.



## Join Support Between Stores or Databases

---

Any join within a given Sybase IQ database are supported. This means that you can join any system or user table in the IQ catalog store with any table in the IQ main 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.
- 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' )
```

## Joining Tables

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

---

Component Integration Services (CIS) let you query remote databases and nonrelational data sources. 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.

### Joining Remote Databases

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.

1. Create proxy tables.

See *System Administration Guide: Volume 2*.

2. Map the remote tables to the proxy tables.

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
```

See “Accessing Remote Data” and “Server Classes for Remote Data Access” in *System Administration Guide: Volume 2*.

## Joins and Subqueries

---

A subquery appears in the main query's **SELECT**, **FROM**, **WHERE** or **HAVING** clause, and allows you to select rows from one table according to specifications obtained from another table.

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.

### Using a Join

To create a chronological list of sales orders (order\_id, OrderDate, and CompanyName) starting with 1994, enter:

```
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
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

## Joining Tables

### 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.

# Managing Databases

To see what actions you can perform on an object in Sybase Central, right-click the object. You can use the resulting submenu to view properties or launch creation wizards. Selecting an object within a database also displays a creation wizard toolbar button.

## Managing Tables

---

Use the appropriate wizard to add a table to your database and define its properties.

**Table 7. Table creation wizards**

For...	Use this wizard...
Persistent tables.	Table Creation wizard
Data that persists only during your connection. Each user has a separate instance of the data in a global temporary table. Those rows are visible only to the connection that inserts them.	Global Temporary Table Creation
Tables on remote servers.	Proxy Table Creation wizard

### Creating an Office Table

Adding a table to a database extends the utility of that database. This example shows you how to add an office table to the demo database.

1. Right-click Dbspaces, point to New, and choose Dbspace.
2. Name the new table `office`, and click Next.
3. Click `IQ_SYSTEM_MAIN` as the dbspace you want to use to store the new table, and click Next.
4. In the Comment field, enter: `Company offices`.
5. Click Finish.

The Table wizard creates the `office` table, opens the columns tab in the right pane, and prompts you to define the columns. Follow the steps below to add columns to the new office table.

## Creating Columns For the Office Table

On the Columns tab, each row represents a single column, and contains all of the attributes for that column.

1. Use the values in the following table to create the first column. Leave the PKey box checked. Leave the ID, Size, Scale, Nulls, Unique and Comments attributes blank

Pkey	Name	Data type
<input checked="" type="checkbox"/>	office_id	integer

2. To create a new column:
  - Tab to the end of the current row.
  - Click the New Column button on the Toolbar.
  - Click File, point to New, and choose Column.
3. Use these attributes to create columns for the `office` table.

Name	Data type	Size	Nulls
region	char	7	<input type="checkbox"/>
street	char	40	<input type="checkbox"/>
city	char	20	<input type="checkbox"/>
state	char	2	<input type="checkbox"/>
zip	char	9	<input type="checkbox"/>

4. To save the column definitions:
  - Click the Save Table button on the Toolbar.
  - Click File, and choose Save.

The `office` table is now visible on the tables tab in the right pane of Sybase Central. The `office` table is now part of the `iqdemo` database. To add data, click the Data tab, right-click, and choose Add Row, or you can use the Interactive SQL utility. The following section describes how to edit an existing table.

## Editing Column Properties

Every column in a table has a set of properties that define the characteristics of that column. Sometimes you may need to change the properties of a column after it exists and contains data.

This example shows you how to rename a column in the `office` table.

1. To display the properties of a column:

- In Tasks view, choose Work with another table in this database from the list of Other Tasks. Double-click the `office` table
  - In Folders view, click the `office` table.
2. If necessary, click the Columns tab in the right panel.
  3. Change the name `zip` column to `postal_code`. Notice that the row that you changed is marked by an icon that indicates that the column is modified.
  4. To save the new column definitions;
    - Click the Save Table button on the Toolbar.
    - Click File, and choose Save.

## **Deleting the Office Table From the Demo Database**

Deleting or dropping a table lets you remove an unnecessary table from the database.

This example shows you how to delete the office table from the demo database.

Right-click the `office` table, and choose Delete.

You can delete proxy tables in the same way. If you try to delete a proxy table that was created with the new local remote option and the remote server is inaccessible, you get an error message indicating that the remote server is inaccessible.

---

**Warning!** Deleting a proxy table that was created with the new local and remote table option also deletes the remote table. You will lose all data stored in the remote database.

---

## **Managing Procedures**

A stored procedure is a group of SQL statements that perform a particular task, on a Sybase IQ server. In Sybase Central, stored procedures are located in the Procedures & Functions folder.

Opening a stored procedure in Sybase Central also opens a SQL editor that you can use to test and modify the procedure. The SQL editor provides:

- Automatic syntax highlighting.
- Automatic formulation of DDL statements.
- Ability to find and replace text, and to jump to specific line numbers.
- Ability to execute the DDL code against the database.

## **Viewing Stored Procedures**

There are several stored procedures included in the demo database. The following steps explain how to view and edit stored procedures.

### **Viewing and Editing Procedures**

The Procedures & Functions folder includes several stored procedures that you can run against the demo database. This example shows you how to open stored procedure, review the SQL statements, and save any changes in the SQL editor.

1. To open a procedure, open the Procedures & Functions folder, and choose ShowCustomers.

Sybase Central displays the procedure in a SQL editor in the right pane. Right-click anywhere in the SQL editor to display standard editing options. You can also use the editing options on the Toolbar.

2. Do one of the following:

- Click the Save button on the Toolbar.
- Click File, and choose Save.

### **Viewing and Altering Procedure Permissions**

To run a stored procedure, you must have explicit permission as a user or member of a group with permission to run the procedure. This example shows you how to review and grant user and group permissions.

1. In the Procedure & Functions folder, right-click a procedure, and choose Properties.
2. Click the Permissions tab to see which users have permissions on this procedure.
3. Click Grant, choose the user or group you want to grant permission to execute this procedure, click OK.
4. Click Apply.

### **Running a Stored Procedure**

You can start run a stored procedure in Sybase Central.

To run a stored procedure, open the Procedures & Functions folder, right-click the procedure you want to run, and choose Execute from Interactive SQL.

## **Managing Users and Groups**

---

Users and groups are governed by login policies that are applied when a user connects to a database.

All new databases include a root login policy. You can modify the root login policy values, but you cannot delete the policy. Login policies govern only the rules for user login and are separate from authorities and permissions. Login policies are not inherited through group memberships.

The following settings are governed by a login policy:



- Password life time
- Password grace time
- Password expiry on next login
- Locked
- Maximum connections
- Maximum failed login attempts
- Maximum days since login
- Maximum non-DBA connections

---

**Note:** For additional information about login policies, see *Sybase IQ online help > Managing users and groups*.

---

### **Adding a New Login Policy**

Use the Login Policy Wizard to create new login policies.

1. Right-click Login Policies, point to New, and choose Login Policy.
2. Name the new login policy, Sales access, then click Next.
3. On the Policy Options dialog, set the appropriate policy options, then click Next to display the comments dialog.
4. On the Comments dialog, type an appropriate comment, then click Finish.

### **Adding a Group To a Database**

Use the Group Wizard to add a Sales group with Sales access login policy to the demo database.

1. Right-click Users & Groups, point to New, and choose Group.
2. Name the new group Sales, then click Next.
3. Click the Enable Password box.
4. In the Password box, type an appropriate password, then re-type the password in the Confirm password box, click Next.

If you click the Require a new password to be specified at next login box, Sybase Central prompts first group member to login to choose a new password at the next login.

5. Choose Sales access as the group login policy, click Next.
6. Choose the authorities you want to assign to this group, click Next.
7. Type an appropriate comment in the comment box and click Finish.

### **Adding a User to the Demo Database**

Use the Create User Wizard to add a new sales person with Sales access permission.

1. Right-click Users & Groups, point to New, choose User.

## Managing Databases

2. Name the new user, “Sales person” then click Next.
3. Click the Enable Password box.
4. In the Password box, type an appropriate password, then re-type the password in the Confirm password box, click Next.

If you click the Require a new password to be specified at next login box, Sybase Central prompts the user to choose a new password the first time that user logs in.

5. Choose Sales access as the login policy, click Next.
6. Choose the authorities you want to assign, click Next.

For more information, see "Database permissions and authorities overview" in *SQL Anywhere Server - Database Administration > Configuring Your Database > Managing user IDs, authorities, and permissions*.

7. Type an appropriate comment in the comment box and click Finish.

## Restoring the Demo Database to its Original State

To restore the demo database, delete the Sales group and the Sales person user.

1. Click Users & Groups.
2. In the right-pane, click the user or group you want to delete, choose Delete.

## Managing User Connections

Connected users are identified by Connection IDs assigned by the server.

1. In the right pane, click the Connected Users tab.
2. Do one of the following:

To...	Do this...
View connected users	In the right pane, click the Connected Users tab. The Connected Users tab lists all connected users. Each user is identified by a Connection ID assigned by the server.
Disconnect users	Right-click the user's Conn ID, choose Disconnect.
End your current connection	Do one of the following: <ul style="list-style-type: none"><li>• Click the Disconnect button on the Toolbar.</li><li>• Click Connections, choose Disconnect.</li></ul>

# Indexing and Loading Data

Sybase IQ's column-based architecture optimizes your ability to perform selections or calculations on attributes of interest to you. For the best performance, you need to choose the right set of indexes for your data and queries. Your database should have an index on every column that affects performance.

Indexes are used to improve data retrieval performance. Traditional indexes often use a B-tree index strategy to point to the data records. That strategy is valuable only if many unique data values are used to filter down to a very small set of records, as with columns of order numbers or customer names, as you would encounter in a transaction-processing system.

Sybase IQ indexes actually represent and store the data so that the data can be used for accelerating a wide variety of queries. This strategy is designed for the data warehousing environment, in which queries typically examine enormous numbers of records, often with relatively few unique values, and in which aggregate results are commonly required.

## Creating Column Indexes

---

When you create a table and specify its columns, Sybase IQ automatically creates certain default storage structures to optimize query processing. If you know what kinds of queries you plan to run, you can add multiple indexes to any column.

It is faster to create all the indexes needed before you insert data into your database. You can drop any of the optional indexes later if you decide you don't need it.

---

**Warning!** Be sure to verify that the index is not important before you drop it. Different queries use different indexes, even when they appear to be similar. Dropping the wrong indexes may adversely affect performance. Capture queries, run times, and query plans to determine which indexes are required.

---

## Column Index Types

Each type of column index is designed to speed processing of a certain kind of data.

Sybase IQ always uses the fastest index available for the current query or join predicate. If you did not create the index types the query optimizer would ideally like to use for a column, Sybase IQ can still resolve queries involving the column, but response time may be slower than it would be with the correct index type(s).

When you designate a column or set of columns as either a **PRIMARY KEY** or **UNIQUE**, Sybase IQ creates a High\_Group index for it automatically. Choose one **PRIMARY KEY** from all **UNIQUE** constraints for the table. Both **PRIMARY KEY** and **UNIQUE** constraints do not allow nulls; however, a unique index would allow them. **UNIQUE** constraints also provide hints on

column constraints to the query optimizer. The *System Administration Guide: Volume 1* describes when to use each type of index and the space and time trade-offs of each.

These index types are unique to Sybase IQ data and cannot be applied to SQL Anywhere tables. For more information, see the *System Administration Guide: Volume 1*.

**Table 8. Sybase IQ column index types**

Index type	Purpose
Compare (CMP)	Stores the binary comparison (<, >, =, <=, >=, or NE) of any two columns with identical data types, precision, and scale.
DATE	An index on columns of data type <b>DATE</b> used to process queries involving date quantities.
Datetime (DTTM)	An index on columns of data type <b>DATETIME</b> or <b>TIMESTAMP</b> used to process queries involving datetime quantities.
High_Group (HG)	An enhanced B-tree index to process equality and group by operations on high-cardinality data. (Recommended for more than 1,000 distinct values or for a table with less than 25,000 rows.)
High_Non_Group (HNG)	A non value-based bitmap index ideal for most high-cardinality DSS operations involving ranges or aggregates.
Low_Fast (LF)	A value-based bitmap for processing queries on low-cardinality data. (Recommended for up to 1,000 distinct values and more than 25,000 rows in the table. Can support up to 10,000 distinct values.)
TIME	An index on columns of data type <b>TIME</b> used to process queries involving time quantities.
WD	Used to index keywords by treating the contents of a <b>CHAR</b> , <b>VARCHAR</b> , or <b>LONG VARCHAR</b> column as a delimited list.

---

**Note:** Sybase IQ assumes you will add either a LF or a HG index to every column in a **WHERE** clause and in a **GROUP BY** clause.

---

### Adding a Low Fast Index To a Column

In this example, appropriate index types and storage locations have been determined for you. Simply follow the steps to index columns in your table.

In this example, appropriate index types and storage locations have been determined for you. Simply follow the steps to index columns in your table.

1. Connect to the *iqdemo* database, if necessary.
2. Right-click Indexes, point to New, choose Index on IQ Table...
3. Select the *Customers* table from the list, name the new index IQ\_IDX\_LF\_ID, then click Next.

4. Choose the Low Fast option as the Index type, then click Next.

The default number of records to add before notification is sufficient for this tutorial. For details about the notify count, see the *System Administration Guide: Volume 1*.

5. On the dbspace dialog, choose an appropriate dbspace to store the index. For this tutorial, choose *iq\_main*, then click Next.
6. On the Columns screen, select the *ID* column and click Add, then Next.
7. Type the comment `Low Fast index for id column` in the Comment box and click Finish to create the index. The Indexes container is updated with the new index.
8. Now add a Low Fast index to the State column, using steps 3 through 7.

---

**Note:**

- The remaining columns in the table do not require any of the optional indexes. The default storage structures are sufficient.
  - You can use the **CREATE INDEX** command instead of Sybase Central to create column indexes if you prefer. For syntax, see *Reference: Statements and Options > CREATE INDEX statement*.
- 

## Creating Join Indexes

---

Join indexes are Sybase IQ internal structures that optimize joins of related tables.

Join indexes are created on sets of columns rather than individual columns. A join index represents a full outer join of two or more tables. The query engine may use this full outer join as a starting point for queries that include left outer, right outer, and inner joins. You can create a join index for any set of columns that your users commonly join to resolve queries. For guidelines, see the *System Administration Guide: Volume 1* and *Performance and Tuning Guide > Joining Tables*.

Three important rules about creating join indexes:

- Create join indexes *after* indexing columns, because the join index will rely on the column indexes of related tables.
- The creator of the join index must also be the owner of the tables used in the index, or the join index will fail.
- Join index columns must have identical data type and scale.

---

**Note:** Some additional tasks are required for multiplex databases. Please see *Using Sybase IQ Multiplex* instead of this chapter if you need to create join indexes in a multiplex.

---

## Creating a Join Index for SalesOrders and SalesOrderItems

This example shows you how to create join indexes. The first join index, *so\_soi\_jdx*, joins the *sales\_order* and *sales\_order\_items* tables.

1. Right-click Join Indexes, point to New, and choose Join Index.
2. In the Name and Table dialog, enter *so\_soi\_jdx* for the name.
3. In the Join Type dropdown list, select Natural, because you are joining equivalent columns with the same name (*id*) from two tables.
4. Choose *SalesOrders* for the left table.
5. Choose *SalesOrderItems* for the right table.
6. Click Next. Sybase IQ locates the join columns.
7. In the Comment space, type the following to describe the join index, *Join index for SalesOrders and SalesOrderItems*.
8. Click Finish. The *so\_soi\_jdx* join index appears in the Join Indexes folder.

---

**Note:** When joining with the NATURAL keyword, take care that *only* the columns to be joined have the same name. (For example *id = id*.) Watch out for common column names like *date* and *description*.

---

## Creating a Join Index for Department and Employee

This example creates a join between *Departments* table and *Employees* table, using equivalent values in columns from the two tables. Follow these steps to create a joined index called *dept\_emp\_jdx*.

1. Right-click Join Indexes, point to New, choose Join Index.
2. Type the name of your new join index, *dept\_emp\_jdx*, in the Name box.
3. In the Join Type dropdown box, select ON. The **ON** clause is required to join equivalent columns with the same name (*dept\_id*) from two tables.
4. Choose *Departments* for the left table.
5. Choose *Employees* for the right table.
6. Click Next.
7. In the Choose Index Columns dialog, select the desired Left Table column, *DepartmentID* and the Right Table column *DepartmentID*.
8. Click Add. After you add the two columns, they appear in the Join Index Columns box as *GROUP0 . Departments . DepartmentID=GROUP0 . employees . DepartmentID*. (You may have to scroll to read the entire join.)
9. Click Next.

10. On the Add Comment screen, type the following to describes the join index, `Join index for Departments and Employees`. Click Finish.

11. The `dept_emp_jdx` join index appears in the Join Indexes folder.

For detailed information about planning and creating join indexes, see *System Administration Guide: Volume 1*.

## Updating Join Index Data

Synchronize your join indexes after loading the tables.

Join indexes are unavailable to other writers while tables referenced by the index are being updated. If you load your tables before you create a join index using those tables, you must synchronize your join indexes after loading the tables.

Sybase Central does not currently support the synchronize operation; you must use the **SYNCHRONIZE JOIN INDEX** command.

## Loading Data

Use Interactive SQL or Sybase Central to load your tables.

### *Interactive SQL*

To load a table with Interactive SQL, use:

- **INSERT FROM SELECT** loads information from a database or attached database.
- **INSERT VALUES** inserts data manually into specified table columns.
- **LOAD** inserts ASCII and binary data from files as well as data from backups and upgrades.

Syntax and examples for the **INSERT** and **LOAD** commands are in *Reference: Statements and Options*.

### *Sybase Central*

To add a row to a table using Sybase Central:

- In Folder view, open the Tables folder, click the table you want to update, then click the Data tab in the right pane. Right-click a row, and choose Add.
- In Tasks view, double-click the Tables folder, choose the table you want to update, then click the Data tab. Right-click a row, and choose Add.





# Managing Dbspaces

Sybase IQ distributes user data across multiple disks at the application level by representing each device as a dbspace. A dbspace can be an operating system file (dbfile) or a raw disk partition. Dbspaces can contain both user data and internal database structures used for startup, recovery, backup, and transaction management

A *store* is one or more dbspaces that store persistent or temporary data for a special purpose. Sybase IQ has four stores:

- The catalog store contains the SYSTEM dbspace and up to twelve additional catalog dbspaces.
- The Sybase IQ main store contains the IQ\_SYSTEM\_MAIN dbspace and other user dbspaces.
- The Sybase IQ temporary store contains the IQ\_SYSTEM\_TEMP dbspace.
- The Sybase IQ shared temporary store contains the IQ\_SHARED\_TEMP dbspace.

For more about dbspaces, see *System Administration Guide: Volume 1 > Working with Database Objects*.

## Creating Dbspaces

---

A dbspace is composed of one or more files. The total number of dbfiles is 16383. A Sybase IQ agent must be running on the machine where you create the dbspace.

1. Right-click Dbspaces, point to New, and choose Dbspace.
2. Describe the dbspace you want to create, including the name, storage location, and striping options.

See *Reference: Statements and Options > ALTER DBSPACE statement* for more information about striping options.

3. Click Next.
4. Click Add.
5. On the File details dialog, specify the properties of the new file:

Option	Description
Specify the logical name for the file	<p>Logical name for the file.</p> <p>The logical name for the file is the actual file name of the dbspace, with a path where necessary. Any relative directory is relative to that initial dbspace. Be sure to specify the file name suffix – <code>.iq</code> for a Sybase IQ main store, <code>.iqtmp</code> for a Sybase IQ temporary store.</p>
Specify the path to the physical file on disk	<p>Path to the file location.</p> <p>A file name without an explicit path reference is created in the same directory as the initial dbspace of that store.</p>
This is a raw device	<p>If the dbspace is a dbfile, deselect the option <code>This is a raw device</code>, then specify the file size in MB, KB, GB or TB.</p>
Reserve size... Megabytes	<p>Amount of space to reserve in megabytes, so that the dbspace can be increased in size in the future. The size can be any number greater than 0. The reserve cannot be changed after the dbspace is created.</p> <p>When you specify reserve size, the database uses more space for internal (free list) structures.</p>

6. Click OK.

7. Click one of the following:

- Add to add another file to the dbspace.
- Click Finish to close the Wizard.

### Next

Sybase recommends that, immediately after creating a database, administrators create a second main dbspace (a user main dbspace) and set it to be the default dbspace. To prevent users from placing tables and indexes in `IQ_SYSTEM_MAIN`, revoke `CREATE` privilege in `IQ_SYSTEM_MAIN` from `PUBLIC` and grant it to the new user main dbspace. Although it is best to create all required dbspaces at database creation, Sybase IQ allows database administrators to add new dbspaces at any time to increase space available to the database.

Once a dbspace is created, you can add or remove files. You can add multiple files to any dbspace, but you can only add multiple dbspaces to the main store.

## Altering Dbspaces

---

To alter a dbspace, right-click a dbspace and choose Properties. You must have DBA privileges to change dbspace properties.

To change the mode, choose a mode from the Mode submenu:

- Read Write – Default for new dbspaces, writes to the dbspace are allowed
- Read Only – Transactions active at the time of the status change may encounter an error and rollback.



# Updating Databases

These exercises show you how to insert rows, cancel and confirm changes, and delete data.

## Adding Rows To a Table

---

Use the **INSERT** statement to add data about a new department.

Suppose that a new eastern sales department is created, with the same manager as the current Sales department. You can add this information to the database using the following **INSERT** statement in Interactive SQL:

```
INSERT
INTO Departments ( DepartmentID, DepartmentName, DepartmentHeadID )
VALUES ( 220, 'Eastern Sales', 902 )
```

If you make a mistake and forget to specify one of the columns, Sybase IQ reports the following error:

```
Error at line 1
The number of insert VALUES does not match the column list.
```

---

**Note:** Case sensitivity may matter when inserting values.

---

### *Nulls*

The **NULL** value is a special value used to indicate that something is either not known or not applicable. However, the **NULL** value is a legal value in any data type. There are two separate and distinct cases where **NULL** is used:

Situation	Description
Missing	The field does have a value, but that value is unknown.
Inapplicable	The field does not apply for this particular row.

Some columns are allowed to contain the **NULL** value, and others are not. To check whether a column in a table allows the **NULL** value, use the **sp\_iqcolumn** stored procedure in Interactive SQL. For example, type the following:

```
sp_iqcolumn Employees
```

In the *nulls* column, a “Y” value indicates that nulls are allowed.

### *A Short Form for INSERT*

You can use a short form to enter values for all the columns in a table in the order they appear when you **SELECT \*** from the table (the order in which they were created). The following is equivalent to the previous **INSERT** command:

```
INSERT
INTO Departments
VALUES ( 220, 'Eastern Sales', 902 )
```

### *Canceling Changes*

The ROLLBACK statement undoes all changes you have made to the database since the last time you made changes permanent.

## Making Changes Permanent

---

Use the COMMIT statement to make all changes permanent.

The default action in DBISQL is to do a COMMIT on exit. This can be controlled with the dbisql option COMMIT\_ON\_EXIT.

dbisql has another option named AUTO\_COMMIT. If this option is on, DBISQL does a COMMIT operation after every command. The default for this option is OFF. Usually you will want it OFF, giving you the opportunity to ROLLBACK the changes if, for example, a delete operation doesn't produce the intended results.

For more information on dbisql options, see *Utility Guide*.

---

**Note:** Use COMMIT with care. When trying the examples in this tutorial, be careful not to COMMIT any changes until you are sure that you want to change the database permanently.

While Sybase IQ excels at bulk changes in a single transaction, overhead for each transaction is higher than a traditional OLTP database. (Also true at the statement level.) This means that COMMIT statements may take longer to complete than in an OLTP system, but in a data warehousing environment, there tend to be fewer commits than in OLTP systems.

Increased overhead from frequent COMMIT operations affects your system performance.

---

## Ensuring Data Consistency

---

Sybase IQ has special features that ensure data consistency in a data warehouse environment.

In a data warehouse environment, many users need to read from the database, but only the DBA needs to update it. However, there is often a need to make updates while other users continue to request and receive query results. Sybase IQ keeps track of database changes using *table-level snapshot versioning*. It keeps a record of what the table looks like when a user begins a write transaction.

### *Data consistency*

Sybase IQ allows multiple readers, but only one writer to a table. To test data consistency, you will connect as two different users and try to write to a table from both connections at the same

time. One statement will be rolled back and will receive an error message while the other commits.

### *Checkpoints*

After each transaction commits, Sybase IQ writes updated data pages to disk. This approach is ideal for the data warehouse, where a single application may write millions of rows of data. A *checkpoint* is a point in a transaction when the database writes information to disk. Most OLTP databases write data to disk at checkpoints. Sybase IQ does not wait for a checkpoint to write physical data. Sybase IQ uses checkpoints to write certain information to disk for internal tracking, which is used if you need to recover your database. While you may set explicit checkpoints, most Sybase IQ checkpoints occur automatically.

## **Savepoints**

---

To limit the amount of data that Sybase IQ writes to disk, set a savepoint. *Savepoints* define a point in a transaction after which all changes can be undone by a ROLLBACK TO SAVEPOINT statement. This allows you to commit data before the entire transaction finishes.

## **Testing Data Consistency**

This example shows you how to test for data consistency.

1. Connect to the database using a connection name in Interactive SQL:

```
CONNECT DATABASE iqdemo
AS sales
```

2. Start a read/write transaction:

```
INSERT INTO FinancialCodes
(Code, Type, Description)
VALUES ('e6', 'expense', 'Services')
```

3. Connect to the same database with a different connection name and run a query:

```
CONNECT DATABASE iqdemo
AS marketing;
SELECT CustomerID, OrderDate
FROM SalesOrder
```

Sybase IQ returns the requested information; this is a read-only transaction.

4. Connect to the same database using a different connection name:

```
CONNECT DATABASE iqdemo
AS accounting
```

5. Start a read/write transaction:

```
INSERT INTO FinancialCodes
(Code, Type, Description)
VALUES ('r3', 'revenue', 'Sales & Marketing')
```

## Updating Databases

This command will fail and be rolled back.

6. As user `sales`, commit your transaction.
7. Retry your transaction as user `accounting`. You may now commit or roll back this transaction.

Set some savepoints to return to as you run two similar transactions. This time, you can put the second transaction on hold until the first one completes.

## Setting Savepoints

You can identify important states within a transaction and return to them selectively by using savepoints to separate groups of related statements.

1. Connect to the database:

```
CONNECT DATABASE iqdemo
```

2. Start a read/write transaction:

```
INSERT INTO FinancialCodes  
(Code, Type, Description)  
VALUES ('e8', 'expense', 'Services')
```

3. Set a savepoint:

```
SAVEPOINT TUES9_45
```

Naming your savepoint (as shown) is optional. You must follow the rules for object names described in *Reference: Statements and Options*.

4. Start a read/write transaction:

```
INSERT INTO FinancialCodes  
(Code, Type, Description)  
VALUES ('r3', 'fees', 'Administration')  
COMMIT
```

If this insert command were to fail, the transaction would roll back to Savepoint `TUES9_45`. The data inserted in Step #2 would not be lost.

## Updating Multiple Tables From Multiple Accounts

If you are working on several tables within one database, Sybase IQ allows multiple readers and writers in a database, as long as the writers write to different tables.

1. Connect to the demo database using connection name `marketing`:

```
CONNECT DATABASE iqdemo  
AS marketing
```

2. Start a read/write transaction:

```
INSERT INTO FinancialCodes  
(Code, Type, Description)  
VALUES ('e9', 'expense', 'R & D')
```



3. Connect to the same database using a different connection name:

```
CONNECT DATABASE iqdemo
AS sales
```

4. Start a read/write transaction using a different table:

```
INSERT INTO SalesOrders
(ID, CustomerID, OrderDate, SalesRepresentative)
VALUES ('2088', '140', '05-29-98', '195')
COMMIT
```

This transaction will commit immediately because, although both are in `iqdemo` database, the insert operations affect different tables. For more details about transaction support, see *System Administration Guide: Volume 1 > Transactions and Versioning*.

## Deleting Rows

---

Sometimes you will want to remove rows from a table.

Suppose Rodrigo Guevara (employee ID 249) leaves the company. The following statement deletes Rodrigo Guevara from the employee table.

```
DELETE
FROM Employees
WHERE EmployeeID = 249
```

### *Using the DELETE Command*

You can delete more than one row with one command. For example, the following statement would delete all employees who had a termination date that is not NULL from the employee table.

```
DELETE
FROM Employees
WHERE TerminationDate IS NOT NULL
```

This example would not remove any employees from the database as the `termination_date` column is NULL for all employees.

With DELETE, the search condition can be as complicated as necessary. For example, if the employee table is being reorganized, the following statement removes from the employee table all employees in the 617 area code with employee ID 902 as manager. This WHERE clause is a compound search condition including a function (LEFT).

```
DELETE
FROM Employees
WHERE LEFT(Phone, 3 ) = '617'
AND ManagerID = 902
```

Since you have made changes to the database that you do not want to keep, you should undo the changes as follows:

```
ROLLBACK
```

## Updating Rows

---

Other times, you may want to change certain rows based on the value of a particular column. The SET clause specifies the columns to update and their new values, and the WHERE clause specifies the rows to update.

For example, if the telephone area code changed from '508' to '978', you could change every row with the old area code to the new area code with the following command:

```
UPDATE Customers
SET LEFT(Phone,3) = '978'
WHERE LEFT (Phone, 3) = '508'
```

## Synchronizing Join Indexes

---

While you update a table that is involved in a join index, that join index is unavailable to other writers. Sybase IQ lets you control when it updates join indexes, because such updates may take a lot of time depending on the order of updates to various tables referenced by the indexes.

The **SYNCHRONIZE JOIN INDEX** statement updates one or all of the join indexes in the current database belonging to the connected user. If you do not specify one or more indexes as argument(s), it synchronizes all join indexes in the current database.

To update a join index, Sybase IQ must update all the other join indexes that reference a table in the join, as well as all the join indexes that reference tables in those join indexes.

You may choose to issue this command before you finish a group of **INSERT** or **DELETE** statements, so that the data will become available to users as soon as the command finishes. You may alternatively choose to run a batch job to issue the command at a time when you know the load on the machine is light to avoid system slowdowns. After each **SYNCHRONIZE JOIN INDEX** statement, you need to issue a **COMMIT** statement.

To synchronize the Employees/SalesOrders join index (EMP\_SO\_JDX), enter:

```
SYNCHRONIZE JOIN INDEX
EMP_SO_JDX
COMMIT
```

---

**Note:** For best performance, avoid issuing the **SYNCHRONIZE JOIN INDEX** command after every insert or delete.

---

## Deleting dbspaces, Tables, and Indexes

You can use Sybase Central or Interactive SQL to delete databases, dbspaces, tables, and indexes (including join indexes).

Do one of the following:

- In Sybase Central, right-click the object, choose Delete.

- In Interactive SQL, use the **DROP** command. For complete syntax, see *Reference: Statements and Options*.



# Using Views

A view is a computed table defined by the result set of its view definition, which is expressed as a SQL query. You can use views to show database users exactly the information you want to present, in a format you can control.

A *regular view* describes a view that is recomputed each time you reference the view, and the result set is not stored on disk. This is the most commonly used type of view.

A *materialized view* describes a view whose result set is precomputed and materialized on disk similar to the contents of a base table. Materialized views are useful in data warehousing scenarios, where frequent queries of the actual base tables can be extremely expensive. Examples in this section refer to regular views.

---

**Note:** Materialized views are only supported for SQL Anywhere tables in the Sybase IQ catalog store.

---

## Defining a View

---

Define views for queries that you use frequently.

Suppose you frequently need to list employees by department. The following query creates a list of all employees by department and state that you can run against `iqdemo`:

```
SELECT Departments.DepartmentID,
       Departments.DepartmentName, Employees.EmployeeID,
       Employees.GivenName, Employees.Surname,
       Employees.Phone
FROM Departments, Employees
ORDER BY Employees.State
```

Running the query creates a results set that looks like this:

Department-tID	Department-Name	Employ-eeID	Given-Name	Sur-name	Phone
100	R & D	148	Julie	Jordan	6175557835
200	Sales	148	Julie	Jordan	6175557835
300	Finance	148	Julie	Jordan	6175557835
400	Marketing	148	Julie	Jordan	6175557835
---	---	---	---	---	---

## Creating a View

In Interactive SQL, you can use a `CREATE VIEW` statement to create and store a view.

This example creates a view called `emp_dept` in `iqdemo`:

```
CREATE VIEW emp_dept AS
  SELECT Departments.DepartmentID,
     Departments.DepartmentName, Employees.EmployeeID,
     Employees.GivenName, Employees.Surname,
     Employees.Phone
FROM Departments, Employees
ORDER BY Employees.State
```

In Sybase Central, you can use a wizard to create a view. To start the wizard, connect to `iqdemo`, right-click Views, point to New, choose View. When the wizard starts, follow the instructions on the screen.

## Displaying View Data

Remember that the information in a view is not stored separately in the database. Each time you refer to the view, SQL executes the associated **SELECT** statement to find the appropriate data. This means that if the sales data changes, information in the view will be automatically up to date.

You can query a view just like a table:

```
SELECT *
FROM emp_dept
```

To see the results in Sybase Central, right-click the View, choose View Data in Interactive SQL.

## Changing a View

In Interactive SQL, you can use an `ALTER VIEW` statement to change a view definition with a modified version.

The `emp_dept` view lists all employees by department. You can use the following command to change the view to display only those employees in the Sales department:

```
ALTER VIEW emp_dept AS
  SELECT Departments.DepartmentID,
     Departments.DepartmentName, Employees.EmployeeID,
     Employees.GivenName, Employees.Surname,
     Employees.Street, Employees.City,
     Employees.State, Employees.Phone
FROM Departments, Employees
WHERE Departments.DepartmentID = 200
ORDER BY Employees.State
```

`ALTER VIEW` replaces the existing `SELECT` statement in the view with the `SELECT` you defined in the `ALTER VIEW` command. Existing permissions on the view are maintained.

To change a view in Sybase Central, right-click the View, and choose Edit in New Window.

## Deleting Views

You can delete a view in Interactive SQL or Sybase Central

To delete a view from the database in Interactive SQL, use the DROP statement:

```
DROP VIEW <view name>
```

To drop a view in Sybase Central, right-click the view you want to delete, and choose Delete from the shortcut menu.

## View Restrictions and Advanced Uses

Views can be thought of as virtual tables. Any **SELECT** command can be used in a view definition except commands containing **ORDER BY**.

Views can use **GROUP BY** clauses, subqueries, and joins. Disallowing **ORDER BY** is consistent with the fact that rows of a table in a relational database are not stored in any particular order. When you use the view, you can specify an **ORDER BY**.

You can also use views in more complicated queries:

```
CREATE VIEW Q1_Revenue AS
SELECT Products.Name AS Item, Products.Description AS Style,
SUM(SalesOrderItems.Quantity) AS Sales, Products.UnitPrice,
SUM(SalesOrderItems.Quantity * Products.UnitPrice) AS Revenue FROM
EmployeesJOIN SalesOrders ON SalesOrders.SalesRepresentative =
    Employees.EmployeeID
JOIN SalesOrderItems ON SalesOrderItems.ID =
    SalesOrders.ID
JOIN Products ON Products.ID =
    SalesOrderItems.ProductID WHERE SalesOrders.OrderDate
>='2001-01-01' AND SalesOrders.OrderDate <='2001-04-30' GROUP BY
Products.Description, Products.Name, Products.UnitPrice
```

Running the query creates a results set that looks like this:

Item	Style	Sales	Unit Price	Revenue
Tee Shirt	Tank Top	744	9.00	6696.00
Sweatshirt	Hooded Sweatshirt	756	24.00	18144.00
Visor	Cloth Visor	756	7.00	5292.00
Baseball Cap	Wool cap	685	10.00	6850.00
Tee Shirt	Crew Neck	672	14.00	9408.00
Shorts	Cotton Shorts	1524	15.00	22860.00
Baseball Cap	Cotton Cap	900	9.00	8100.00

Item	Style	Sales	Unit Price	Revenue
Tee Shirt	V-neck	780	14.00	10920.00
Sweatshirt	Zippered Sweatshirt	564	24.00	13536.00
Visor	Plastic Visor	684	7.00	4788.00

---

**Note:** Views cannot be used in a join index.

---

## Using Views for Security

---

Views can be used to restrict access to information in the database.

### *Scenario*

You want to create user ID for the sales department head, Moira Kelly, and restrict its use so that it can only examine information about employees in the sales department.

### *Creating the New User ID*

First, create the new user ID for Moira Kelly using the **GRANT** statement. From dbisql, connect to the demo database as **dba**, type the following:

```
GRANT CONNECT TO M_Kelly
IDENTIFIED BY SalesHead
```

### *Granting Permissions*

Next you need to grant user M\_Kelly the right to look at employees of the sales department.

```
CREATE VIEW SalesEmployee AS
SELECT EmployeeID, Surname, GivenName
FROM Employees
WHERE DepartmentID=200
```

Now you must give M\_Kelly permission to look at the new view by entering:

```
GRANT SELECT ON SalesEmployee TO M_Kelly
```

### *Looking at the View*

Connect to the database as M\_Kelly and now try looking at the view:

```
CONNECT USER M_Kelly IDENTIFIED BY SalesHead;
SELECT * FROM "dba".SalesEmployee
```

emp_id	emp_lname	emp_fname
129	Chin	Philip
195	Dill	Marc



emp_id	emp_lname	emp_fname
299	Overbey	Rollin
467	Klobucher	James
641	Powell	Thomas
...	...	...

However, you do not have permission to look directly at the `employee` and `department` tables. If you execute the following commands, you will get permission errors.

```
SELECT * FROM Employees;
SELECT * FROM Departments
```

### *Using Built-In Functions for Secure Views*

You can also take advantage of built-in Sybase IQ functions when creating secure views. In the following example, the view `secure_view` is intended to restrict access to records in the `secure_table` to specific users.

The view definition uses the `suser_name()` function, a built-in Sybase IQ function that stores a server user name for each user. The following view allows a user to see records if he or she has all of the following:

- A specific server user name (`suser_name`)
- A grade equal to or higher than the viewed record(s) (as defined by records in the `security_auth_table`)

```
CREATE VIEW secure_view AS
SELECT a.secure_name, a.security_code
FROM secure_table a, security_auth_table b
WHERE SUSER_NAME() = b.opr_name
      AND a.department_id = b.department_id
      AND a.grade <= b.grade
```

You may also create a view that restricts access by using the built-in function `suser_id()`, which stores a unique identification number for each user.

## **Restricting Access to Information In the Database**

When you define a view, include only the columns which will be used in queries in the **SELECT** list of the view.

Do not use the **CONTAINS** predicate in a view that has a user-defined function, because the **CONTAINS** criteria will be ignored. Use the **LIKE** predicate with wildcards instead, or issue the query outside of a view.

The following may degrade the performance of views in queries (when compared to similar queries that do not use views).

## Using Views

- Nested views having any items appear on the select-list of both the inner and outer views that are not used by the query referencing the outer view.
- Cases where there are joins both inside and outside the view, and the view does not contain an explicit or implicit **GROUP BY** clause.
- Cases where a user breaks up a query by placing part of it into a view and now multiple predicates reference the same column inside and outside the view.
- In cases where a predicate outside the view contains a reference to a view select-list item plus either an outer reference or an uncorrelated subquery.

When a view specification contains an outer join, a predicate in a query that references the view cannot be “pushed down” into the indexes (since it would change the semantics at the outer join). So, you may notice that such queries run slower than a similar query that does not use a view.

Predicates are described in *Reference: Statements and Options*.

# Glossary

Sybase IQ terms and concepts.

- **Adaptive Server Enterprise** – Adaptive Server Enterprise is a high-performance relational database management system. You can use Sybase IQ to query data in ASE databases.
- **Catalog store** – A catalog store is the portion of each Sybase IQ database that contains its metadata. The catalog store contains the SYSTEM dbspace and up to 12 additional other catalog dbspaces. The default name is dbname . db.
- **Component Integration Services (CIS)** – Component Integration Services (CIS) provides Sybase IQ users with direct access to relational or nonrelational databases on the mainframe, UNIX, or Windows servers.
- **Connection Profile** – Connection profiles store connection information to a running Sybase IQ server. The profile is primarily used to simplify user connections to a server. Sybase IQ extends connection profiles to facilitate starting servers and creating databases.
- **Dbfile** – A dbfile is an operating system file used to store data for a Sybase IQ database. Each dbfile has a corresponding logical filename and physical file path. Each dbspace name, dbfile name, and physical file path must be unique. The dbfile name can be the same as the dbspace name.

The *SYSDBFILE* view shows all the dbfiles in your database, including the catalog dbspace file, the IQ message file, dbfiles in the IQ main and temporary dbspaces, the transaction log file, and the SA temporary file.

- **Dbspace** – A dbspace is a logical collection of dbfiles. If a database runs out of room, you can expand it by adding additional dbspaces. Users can move Sybase IQ data off of disks and take the disks offline without any downtime.
- **Free list** – A free list is a structure that Sybase IQ uses to track which blocks are in use by a dbspace.
- **IQ database** – An IQ database is a database that you create using a Sybase IQ server. IQ databases are specially indexed to take advantage of the query speed of Sybase IQ.

Each IQ database that you create includes three stores: an IQ main store (for data), a catalog store (for metadata), and an IQ temporary store (for temporary data). It also generates an IQ message log file.

- **IQ main store** – The IQ main store is the portion of each Sybase IQ database that contains the IQ\_SYSTEM\_MAIN dbspace and other user dbspaces. The IQ main store is contains persistent database structures, such as backup metadata and rollback data for committed transactions.

Sybase recommends that you avoid storing user tables and indexes in IQ\_SYSTEM\_MAIN and instead create additional dbspaces, called user main dbspaces, to store user tables and indexes. The IQ main store is sometimes called the IQ store.

- **IQ message log** – An IQ message log file created when the first user connects to an IQ database. The default name for this file is dbname.iqmsg.

IQ\_SYSTEM\_MSG is a system dbspace that points to the file path of the database IQ message log file. IQ\_SYSTEM\_MSG is not considered a store because it doesn't store any data. Sybase IQ logs error messages, status messages and insert notification messages in this file.

- **IQ temporary store** – The IQ temporary store contains the IQ\_SYSTEM\_TEMP dbspace. The IQ temporary store is the portion of each IQ database that stores temporary tables and temporary scratch space data structures.

The database server uses temporary data structures to sort and process data. Data in these tables persists only as long as you are connected to the database.

- **IQ transaction log** – The IQ transaction log records changes to the database. The transaction log includes version information, free space, and other information you can use to recover from a system failure. By default, the transaction log is created in the same directory as the catalog store. The default name for this dbfile is dbname.log.
- **Join index** – A join index is a special type of index used in Sybase IQ. Conceptually, an index in a database is like an index in a book. In a book, the index relates each indexed term to the page or pages on which that word appears. In a database, the index relates each indexed column value to the physical location at which the row of data containing the indexed value is stored.

Join indexes may improve response time for queries that join two or more tables. In certain situations, ad hoc queries may be preferred instead of join indexes.

- **metadata** – Metadata is data that describes the data in your database – for example, the size and data type of each column in a table. Metadata for each Sybase IQ database is stored in the catalog store.
- **Multiplex** – A powerful feature in Sybase IQ that provides application scalability through a clustered server configuration. Sybase IQ multiplex allows concurrent data loads and queries via independent data processing nodes connected to a shared data source. Each multiplex server has its own catalog store and IQ temporary store and all the servers share a common IQ store. For more information, see *Using Sybase IQ Multiplex*.
- **Object** – An object can be a user-created table, index, or join index. Objects are divided into persistent objects, which remain in the database over user disconnects and server restarts, and temporary objects, tables and views that only remain in the database during the current session. Permanent tables are also called base tables.
- **Partition key** – A partition key is a table column defined by the table creator that determines how a table should be partitioned.
- **Proxy table** – A proxy table is a table object that maps to a table on a remote server, and whose column attributes and index information are derived from the object at the remote location.

You can use proxy tables to search data in multiple SQL Anywhere servers, ASE databases, and non-Sybase databases. Conversely, you can also create proxy tables that enable you to query your Sybase IQ, SQL Anywhere, and ASE databases.

- **Range partition** – A range partition is logical subset of table rows based on the values of a single table column.
- **SQL Anywhere** – SQL Anywhere is Sybase's transaction-processing relational database management system which can be used standalone or as a network server in a multiuser client/server or three-tier environment.

SQL Anywhere is specifically designed to use fewer memory and disk resources than the average database management system. Sybase IQ is an extension of SQL Anywhere, and supports many of the same features.

- **SQL Anywhere database** – Every Sybase IQ database uses an SQL Anywhere database for the catalog store.
- **Store** – A store is one or more dbspaces that store persistent or temporary data for a special purpose. Sybase IQ has three stores: the catalog store, the IQ main store, and the IQ temporary store.
- **Synchronize command** – A synchronize command updates join index data. You use the SYNCHRONIZE JOIN INDEX statement to do this. The system administrator needs to synchronize Sybase IQ join indexes periodically if multiple users update tables that may be involved in join indexes.
- **Synchronization** – Synchronization brings an outdated multiplex secondary node server up to date.
- **Table partition** – A table partition is a collection of rows that is a subset of a user-created table. A given row cannot be placed in two different partitions. Each partition can be placed in its own dbspace and managed individually.
- **Tablespace** – A tablespace unit of storage within the database that may be administered as a logical subset of total storage. You may allocate individual objects and subobjects to individual tablespaces. A tablespace in Sybase IQ is referred to as a dbspace.



# Index

## A

- About Sybase IQ 1
- ad hoc queries 49
- Adaptive Server Enterprise 85
- adding users to databases 59
- Advanced Security 1
- alphabetical order 30
- ALTER VIEW statement 80
- AND keyword 34
- apostrophes
  - using 32
- Architecture
  - Sybase IQ 5

## B

- base tables 16
- BETWEEN conditions 36
- BLANK PADDING
  - CREATE DATABASE 51
  - effect on joins 51
  - support of OFF 51

## C

- case sensitivity 29, 32
- catalog
  - system 17
- Catalog Store 30, 85
- CIS 52
  - Adaptive Server Enterprise 51
  - SQL Anywhere 51
- Clauses
  - FROM clause 30
  - GROUP BY clause 38, 83
  - GROUP BY CUBE clause 41
  - GROUP BY ROLLUP clause 39
  - ORDER BY clause 32, 43
  - SET clause 76
  - WHERE clause 32, 35, 36, 75, 76
  - WITHIN GROUP clause 43
- Client Connections
  - UNIX 20
  - Windows 19

- Column indexes
  - adding low fast indexes 62
  - comparing 61
  - creating 61
  - High\_Group indexes 61
  - High\_Non\_Group indexes 61
  - join indexes 63
  - Low\_Fast indexes 61
  - types 61
  - WD indexes 61
- columns
  - ordering 31
  - selecting from a table 31
  - significant number of null values 31
- Columns
  - about 29
  - creating 56
  - editing properties 56
- command line connections 26
- commands
  - long 32
- COMMIT statement 72, 76
- COMMIT\_ON\_EXIT statement 72
- compare indexes 61
- comparisons
  - about 32, 33
- component integration services (CIS) 85
- Concepts 11
  - database computing 11
  - language interface 11
  - queries 16
  - Queries 16
  - relational databases 13
  - servers 11
  - SQL statements 17
  - system tables 17
- conditions
  - and GROUP BY clause 38
  - search 32, 33, 36
- CONNECT statement 82
- connection profile 85
- connectivity sample applications 7
- containers
  - expanding 23
  - selecting 23

## Index

- correlation names
  - about 45
  - defined 52
- COUNT function 37
- CREATE DATABASE Statement
  - BLANK PADDING 51
- CREATE VIEW statement 79, 80
- CUBE clause 41
- D**
- data
  - loading 65
  - tabular 13
- Data consistency
  - checkpoints 72
  - savepoints 73
  - setting savepoints 74
- Data consistency
  - savepoints 73
- data dictionary 17
- data types
  - about 14
  - joins 50
- data, selecting 29
- database server
  - starting 8
- database tables
  - characteristics 14
  - columns 14
  - rows 14
  - tuples 14
- Database tables
  - foreign keys 15, 16
  - indexes 16
  - Java 16
  - other database objects 16
  - primary keys 15
  - stored procedures 16
  - users and groups 16
  - views 16
- Database, demo
  - iqdemo 8
- databases
  - managing 55
- Databases
  - adding rows 71
  - administering 21
  - changing 72
  - connecting to 26
  - data consistency 72
  - dbspaces 76
  - deleting rows 75
  - demo 8
  - files 12
  - heterogeneous 52
  - indexes 76
  - joining remote 52
  - managing 21
  - relational 13
  - remote 52
  - restricting access 83
  - savepoints 73
  - synchronizing join indexes 76
  - tables 76
  - updating 71
  - updating multiple tables 74
  - updating rows 76
- dates 33, 36
- dba tables 8
- dbfile 85
- dbisql
  - See Interactive SQL
- dbspace 85
- dbspaces
  - altering 69
  - creating 67
  - deleting 76
  - IQ\_SHARED\_TEMP 67
  - IQ\_SYSTEM\_MAIN 67
  - IQ\_SYSTEM\_TEMP 67
  - managing 67
  - SYSTEM 67
  - viewing with Sybase Central 24
- DELETE statement 76
- deleting
  - tables 57
- Deleting
  - dbspaces 76
  - indexes 76
  - objects 76
  - tables 76
- demo database
  - starting 7
- Demo database
  - adding users 59
  - connecting to 26
  - dba tables 8
  - deleting office table 57



- GROUPO tables 8
- iq\_dummy table 8
- iqdemo 8
- restoring 60
- Demo Database 8
- DENSE\_RANK function 42
- disconnecting users from databases 60
- documentation 3
- DROP statement 76
- DROP VIEW statement 81
- dummy Sybase IQ table 30

## E

- Editing
  - column properties 56
- Enterprise Edition 1
- equijoins 64
- Evaluation Edition 1

## F

- Features, optional
  - Advanced Security 1
  - Multiplex Grid 1
  - Unstructured Data Analytics 1
  - Very Large Database Management 1
- foreign keys 46
  - about 15
  - defined 15
- free list 85
- FROM clause 30, 45
  - and joins 45
- functions
  - aggregate 37
  - analytical 42
  - AVG function 37
  - COUNT function 37
  - LIST function 37
  - MAX function 37
  - MIN function 37
  - parameters 37
  - SOUNDEX function 36
  - to functions 37
  - viewing with Sybase Central 24
- functions, analytical
  - DENSE\_RANK function 42
  - NTILE function 42
  - OVER clause 42

- PERCENT\_RANK function 42
- PERCENTILE\_CONT function 42
- PERCENTILE\_DISC function 42
- RANK function 42
- functions, inverse distribution 43

## G

- glossary 85–87
- GRANT statement 82
- GROUP BY clause 38, 39
- GROUP BY CUBE clause 41
- GROUP BY ROLLUP clause 39
- grouped data 37
- GROUPO tables 8

## H

- heterogeneous databases 52
- High\_Group indexes 61
- High\_Non\_Group indexes 61

## I

- IN conditions 36
- indexes 49
  - comparing 61
  - definition 16, 24
  - deleting 76
  - High\_Group indexes 61
  - High\_Non\_Group indexes 61
  - join indexes 63
  - low fast indexes 62
  - Low\_Fast indexes 61
  - synchronizing join indexes 76
  - types 61
  - WD indexes 61
- indexing data 61
  - comparing indexes 61
  - creating 61
  - join indexes 63
  - types 61
- inequality, testing for 33
- INSERT FROM SELECT statement 65
- INSERT statement 65, 76
  - examples 71
- INSERT VALUES statement 65
- Interactive SQL 6
  - connection parameters 26

## Index

- loading data 65, 71
- starting, Windows 27

### IQ Agent

- Running 22

IQ database 85

IQ message Log 86

IQ Store 85

IQ temporary store 86

IQ transaction log 86

iq\_dummy table 8, 30

IQ\_SHARED\_TEMP 67

IQ\_SYSTEM\_MAIN 67

IQ\_SYSTEM\_TEMP 67

iqdemo.db file

- about 8

## J

java sample applications 7

join index 86

join indexes

- creating 63

Join indexes

- Department and Employee 64

- SalesOrders and SalesOrderItems 64

- updating 65

Join Operators

- key joins 47

- natural joins 48

joins

- defined 17

Joins 45

- ad hoc joins 49

- Adaptive Server Enterprise 51

- BLANK PADDING 51

- CIS 51, 52

- cross product 45

- data types 50

- foreign keys 46

- FROM clause 45

- heterogeneous databases 52

- join indexes 49

- natural 48

- Operators 47

- ORDER BY clause 45

- outer joins 52

- primary keys 46

- remote databases 52

- restricting 45

- SQL Anywhere 51

- subqueries 52

- synchronizing 76

- two or more tables 47

- WHERE clause 45

## K

key joins 47

- using 47

keys

- foreign 15

- primary 15

Keywords

- AND 34

- DISTINCT 43

## L

licensing 1

LOAD statement 65

loading data 61

- Interactive SQL 65

- programming interfaces 65

- Sybase Central 65

- variable length format 65

Login policy

- adding 59

- adding groups 59

- adding users 59

- restoring demo database 60

- user connections 60

logs

- message 12

- transaction 13

low fast index 62

Low\_Fast indexes 61

## M

make files 7

managing

- users and groups 58

managing databases 55

Managing databases 55

- procedures 57

- tables 55

- users and groups 58

Managing tables

- creating columns 56

- creating office table 55
- deleting tables 57
- editing columns 56
- manuals 3
- materialized view 79
- maximum function 37
- Message Log
  - IQ 12
- metadata 86
- multiplex 86
- Multiplex Grid 1

## N

- natural joins 48
- network server
  - starting 8
- network server executable 8
- NTILE function 42
- NULL value
  - allowed in columns 71

## O

- object 86
- ODBC sample applications 7
- Optional features
  - Advanced Security 1
  - Multiplex Grid 1
  - Unstructured Data Analytics 1
  - Very Large Database Management 1
- options
  - unexpected behavior 30
- ORDER BY clause 32, 43, 45
- outer references
  - defined 52
- OVER clause 42

## P

- partition key 86
- passwords
  - and permissions 82
- pattern matching 35
- PERCENT\_RANK function 42
- PERCENTILE\_CONT function 42
- PERCENTILE\_DISC function 42
- performance 13
  - improving 38

- Permissions
  - stored procedures 58
- plug-ins
  - connecting 22
  - registering 22
- Predicates
  - CONTAINS predicate 83
  - LIKE predicate 83
- primary keys 46
  - defined 15
- procedures
  - viewing with Sybase Central 24
- Procedures
  - managing 57
  - permissions 58
  - running 58
  - viewing 57
- processing queries without 30
- Product editions
  - Enterprise Edition 1
  - Evaluation Edition 1
  - Single Application Server 1
  - Small Business Edition 1
- program group 7
- programming interfaces 65
- projections 17
- proxy database 86
- proxy table 57

## Q

- queries
  - processing by SQL Anywhere 30
  - writing 29
- Queries
  - ad hoc queries 49
  - indexes 49
  - joins 17
  - projections 17
  - restrictions 17
- querying tables 30
- quotation marks
  - using 32

## R

- range 33
- range partition 87
- RANK function 42

## Index

- Raw Device Access 7
- readme file 7
- regular view 79
- Related Sybase products
  - PowerDesigner 3
  - SQL Anywhere 3
  - Sybase ETL 3
  - Sybase RAP – The Trading Edition 3
  - WorkSpace 3
- relational database concepts 13
- remote databases 52
  - joining 52
- restrictions 32
- ROLLBACK statement 75
- ROLLBACK TO SAVEPOINT statement 73
- ROLLUP clause 39
- rows
  - about 29
  - selecting 32
  - updating 76

## S

- sample applications 7
- savepoints 73
- Savepoints
  - data consistency 73
  - setting 74
- search conditions
  - date comparisons 33
  - introduction to 32
  - short cuts for 36
  - subqueries 52
- SELECT DISTINCT statement 43
- SELECT statement 29, 80
  - about 52
- selecting data 29
- Server
  - shutdown 27
- servers
  - definition 11
- Servers
  - connecting 19
- Service Manager 7
- SET clause 76
- Shutdown
  - server 27
- Single Application Server 1
- Small Business Edition 1
- sp\_iqtable procedure 29

- SQL 13
- SQL Anywhere 87
- SQL statements 17
  - FROM clause 45
  - ORDER BY clause 45
  - WHERE clause 45
- SQL Statements
  - ALTER VIEW statement 80
  - COMMIT statement 72, 76
  - COMMIT\_ON\_EXIT statement 72
  - CONNECT statement 82
  - CREATE DATABASE statement 51
  - CREATE VIEW statement 79, 80
  - DELETE statement 76
  - DROP statement 76
  - DROP VIEW statement 81
  - GRANT statement 82
  - INSERT FROM SELECT statement 65
  - INSERT statement 65, 71, 76
  - INSERT VALUES statement 65
  - LOAD statement 65
  - ROLLBACK statement 71, 75
  - ROLLBACK TO SAVEPOINT statement 73
  - SELECT DISTINCT statement 43
  - SELECT statement 29, 52, 80
  - SYNCHRONIZE JOIN INDEX statement 76
- start\_iq 8
- start\_iq.exe 8
- store 87
- Store
  - Temporary 12
- stored procedures 29
  - definition 16
- Stored procedures
  - managing 57
  - permissions 58
  - running 58
  - triggers 57
- Stores
  - catalog 67
  - main 67
  - shared temporary 67
  - temporary 67
- Structured Query Language
  - See SQL
- subqueries
  - outer joins 52
  - using 52
- SUM function 37

- Sybase Central 6
    - description 21
    - in program group 7
    - loading 65
    - main window 21
    - server connections 19
    - stopping 22
    - UNIX 20
    - Windows 19
  - Sybase IQ 55
    - about 1
    - architecture 5
    - benefits 6
    - components 6
    - Concepts 11
    - database server 8
    - definition 1
    - demo database 8
    - Demo Database 8
    - documentation 3
    - indexing data 61
    - joins 45
    - licensing 1
    - loading data 61
    - managing dbspaces 67
    - optional features 1
    - product editions 1
    - related products 3
    - Service Manager 7
    - stores 67
    - updating databases 71
    - views 79
  - Sybase products, related
    - PowerDesigner 3
    - SQL Anywhere 3
    - Sybase ETL 3
    - Sybase RAP – The Trading Edition 3
    - WorkSpace 3
  - synchronization 87
  - synchronize command 87
  - SYNCHRONIZE JOIN INDEX statement 76
  - SYSTEM 67
  - SYSTEM dbspace 30
  - system stored procedures 29
  - system tables 17
- T**
- table partitions 87
  - tables
    - characteristics 14
    - correlation name 45
    - creating 55
    - creating columns 56
    - deleting 57, 76
    - editing columns 56
    - foreign keys 46
    - iq\_dummy 30
    - listing 29
    - managing 55
    - primary keys 46
    - proxy table 57
    - updating multiple 74
    - viewing with Sybase Central 24
  - tables, iqdemo
    - dba tables 8
    - GROUPO tables 8
    - iq\_dummy table 8
  - tablespace 87
  - tabular data 13
  - Temporary Store 12
  - transaction log 13
  - tuples 14
  - tutorials
    - Sybase Central 21
- U**
- Unstructured Data Analytics 1
  - User connections
    - managing 60
  - user manuals 3
  - users
    - adding to databases 59
    - viewing with Sybase Central 24
  - Users and groups
    - adding groups 59
    - adding login policy 59
    - adding users 59
    - user connections 60
    - managing 58
    - restoring demo database 60
- V**
- variable length format 65
  - Very Large Database Management 1
  - Views
    - changing 80
    - CREATE VIEW statement 79, 80

## Index

- creating 80
- defining 79
- definition 16
- deleting 81
- displaying 80
- materialized view 79
- regular view 79
- restricting 81
- restricting access 83
- security 82
- Using 79
- views 82

## W

- WD indexes 61
- WHERE clause 45, 75, 76
  - and pattern matching 35
  - BETWEEN conditions 36
  - date comparisons 33
  - examples 32
  - ORDER BY clause 32
- WITHIN GROUP clause 43