SYBASE®

User's Guide

SQL Expert

12.5

DOCUMENT ID: 33401-01-1250-01

LAST REVISED: September 2001

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608.

Contents

About This Book

This book describes Sybase® SQL Expert, a SQL performance tuning tool that helps monitor, tune, and rewrite SQL statements quickly and efficiently.

Audience

This book is for developers, Database Administrators, and other users concerned with well-written, efficient SQL statements. This book assumes familiarity with Transact-SQL®, the Sybase version of Structured Query Language (SQL).

How to use this book

This book can assist you to configure and use the Sybase SQL Expert. It includes the following chapters:

Chapter 1, “Introduction to SQL Expert” describes some problems developers encounter while tuning SQL statements, and the solutions SQL Expert offers.

Chapter 2, “Getting Started” helps you to log in and set your preferences.

Chapter 3, “Syntactical SQL Optimizer Features” explains the Optimizer special features, such as the ability to create temporary tables and the Abstract Group Plan.

Chapter 4, “Syntactical SQL Optimizer, explains the integrated module that tunes SQL statements.

Chapter 5, “Retrieving Run Times,” explains the SQL Expert Run Time window.

Chapter 6, “Viewing and Comparing SQL Statements,” explains how to view the database objects in your application.

Chapter 7, “SQL Monitor,” shows how to capture, view, and analyze SQL statements currently running on the database server.

Chapter 8, “SQL Scanner and Viewer,” shows how to detect problematic SQL statements in source code and database objects without executing them.

Appendix A, “SQL Expert Tutorial,” provides some hands-on experience in using the SQL Expert.

The Glossary contains definitions of SQL Expert terms.

Related documents

The following documents comprise the Sybase Adaptive Server Enterprise documentation:

- The release bulletin for your platform – contains last-minute information that was too late to be included in the books.

  A more recent version of the release bulletin may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use the Sybase Technical Library.

- The *Installation Guide* for your platform – describes installation, upgrade, and configuration procedures for all Adaptive Server and related Sybase products.

- *Configuring Adaptive Server Enterprise* for your platform – provides instructions for performing specific configuration tasks for Adaptive Server.

- *What's New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server version 12.5, the system changes added to support those features, and the changes that may affect your existing applications.

- *Transact-SQL User's Guide* – documents Transact-SQL, Sybase's enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database management system. This manual also contains descriptions of the `pubs2` and `pubs3` sample databases.

- *System Administration Guide* – provides in-depth information about administering servers and databases. This manual includes instructions and guidelines for managing physical resources, security, user and system databases, and specifying character conversion, international language, and sort order settings.

- *Reference Manual* – contains detailed information about all Transact-SQL commands, functions, procedures, and datatypes. This manual also contains a list of the Transact-SQL reserved words and definitions of system tables.

- *Performance and Tuning Guide* – explains how to tune Adaptive Server for maximum performance. This manual includes information about database design issues that affect performance, query optimization, how to tune Adaptive Server for very large databases, disk and cache issues, and the effects of locking and cursors on performance.

- *Utility Guide* – documents the Adaptive Server utility programs, such as isql and bcp, which are executed at the operating system level.
- *Quick Reference Guide* – provides a comprehensive listing of the names and syntax for commands, functions, system procedures, extended system procedures, datatypes, and utilities in a pocket-sized book. Available only in print version.
- *System Tables Diagram* – illustrates system tables and their entity relationships in a poster format. Available only in print version.
- *Error Messages and Troubleshooting Guide* – explains how to resolve frequently occurring error messages and describes solutions to system problems frequently encountered by users.
- *Component Integration Services User's Guide* – explains how to use the Adaptive Server Component Integration Services feature to connect remote Sybase and non-Sybase databases.
- *Java in Adaptive Server Enterprise* – describes how to install and use Java classes as datatypes, functions, and stored procedures in the Adaptive Server database.
- *Using Sybase Failover in a High Availability System* – provides instructions for using Sybase's Failover to configure an Adaptive Server as a companion server in a high availability system.
- *Using Adaptive Server Distributed Transaction Management Features* – explains how to configure, use, and troubleshoot Adaptive Server DTM features in distributed transaction processing environments.
- *EJB Server User's Guide* – explains how to use EJB Server to deploy and execute Enterprise JavaBeans in Adaptive Server.
- *XA Interface Integration Guide for CICS, Encina, and TUXEDO* – provides instructions for using Sybase's DTM XA Interface with X/Open XA transaction managers.
- *Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Sybase jConnect for JDBC Programmer's Reference* – describes the jConnect™ for JDBC™ product and explains how to use it to access data stored in relational database management systems.
- *Full-Text Search Specialty Data Store User's Guide* – describes how to use the full-text search feature with Verity to search Adaptive Server Enterprise data.

- *Historical Server User's Guide* – describes how to use Historical Server to obtain performance information for SQL Server® and Adaptive Server.
- *Monitor Server User's Guide* – describes how to use Monitor Server to obtain performance statistics from SQL Server and Adaptive Server.
- *Monitor Client Library Programmer's Guide* – describes how to write Monitor Client Library applications that access Adaptive Server performance data.

Other sources of information

Use the Sybase Technical Library CD and the Technical Library Product Manuals Web site to learn more about your product:

- Technical Library CD contains product manuals and is included with your software. The DynaText browser (downloadable from Product Manuals at http://www.sybase.com/detail/1,3693,1010661,00.html) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting the Technical Library.

- Technical Library Product Manuals Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition to product manuals, you will find links to the Technical Documents Web site (formerly known as Tech Info Library), the Solved Cases page, and Sybase/Powersoft newsgroups.

  To access the Technical Library Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **For the latest information on product certifications**

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Select Products from the navigation bar on the left.

3 Select a product name from the product list.

4 Select the Certification Report filter, specify a time frame, and click Go.

5 Click a Certification Report title to display the report.

❖ **For the latest information on EBFs and Updates**

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Select EBFs/Updates. Enter user name and password information, if prompted (for existing Web accounts) or create a new account (a free service).

3 Specify a time frame and click Go.

4 Select a product.

5 Click an EBF/Update title to display the report.

❖ **To create a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/

2 Click MySybase and create a MySybase profile.

Conventions

In this manual, the following typefaces and fonts have special significance:

Table 1:

Example	
create	commands
sql.ini	filenames, variable names, and book titles
sp_show	database objects
`myEntry`	material you type in
`x=1`	material the computer returns

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Introduction to Sybase SQL Expert

Sybase SQL Expert helps prevent inefficient or over-complex SQL statements at the development level, ensuring the quality of the SQL statements embedded in your source code.

This chapter describes some problems developers encounter while tuning SQL statements, and the solutions Sybase SQL Expert offers.

SQL statement performance problems

If a SQL statement produces the correct result, it is typically integrated into the source code. Badly written SQL statements use processing power and I/O resources. Here are some examples of mistakes even experienced programmers make when constructing SQL statements:

- Producing a SQL statement that does not utilize existing indexes.
- Applying unnecessary function calls; for example, applying ISNULL functions where the first argument is a non-null column.
- Using the wrong driving path in the table joins.

Sybase SQL Expert components

Sybase SQL Expert is a powerful Windows-based tool that identifies, evaluates, and rewrites SQL statements according to your database structure, indexes, and data distribution. It requires minimal human intervention. SQL Expert provides the following functions and modules, each of which is discussed in detail in later chapters.

Syntactical SQL Optimizer

The Syntactical SQL Optimizer analyzes the input source SQL statement and uses a feedback-searching engine to generate a list of semantically equivalent SQL statements. You can then test run the list of optimized SQL statements and determine which one best fits the needs of your database application. The Syntactical SQL Optimizer includes the following components:

- SQL Editor–An editing window that allows you to enter a SQL statement for optimization.
- Query Plan–shows the query plan that the database optimizer has chosen for the source SQL statement.
- Optimization–generates a list of semantically equivalent SQL statements.
- Optimized SQL Viewer–displays the source SQL statement and the list of semantically equivalent SQL statements after optimization, in the order of ascending Sybase Cost.
- Runtime and batch run–allows you to test run both source and optimized SQL statements, to select which statement provides better performance. This feature provides both the elapsed and response time for each statement, to further identify the most suitable statement.
- Run result–executes the SQL statement and displays the corresponding result from the database.
- SQL Comparer–compares source and optimized SQL statements and query plans.
- Activity Log–records optimization and query plan during daily processing.

SQL Formatter

The SQL Formatter ensures that the format of SQL statements is consistent throughout the application. When you enter a SQL statement, the SQL Formatter checks its syntax and highlights comments, bind variables, and force options, making the program easier to maintain.

SQL Database Explorer

The Database Explorer accesses information about database objects from the server, and provides details of tables, views, procedures, triggers, defaults, and so forth. The amount of information displayed depends on the object type and the level of user privilege.

SQL Monitor

The SQL Monitor captures real-time SQL statements from the Sybase Monitor Server, and categorizes the statement according to user-defined criteria of performance problem levels. SQL Monitor includes:

- Collector Manager–provides a work area for adding and deleting collectors. It stores information on the collector name and description, the number of SQL statements analyzed, and their start, end, and duration times.
- Monitor–locates and analyzes SQL statements at run-time, according to user-defined criteria.
- Monitored SQL Viewer–provides a window that displays the monitored SQL statements, their query plans, and any related information, such as the server process ID (SPID), login name, capture time, and database name. You can copy problematic SQL statements from the Monitored SQL Viewer to the Syntactical SQL Optimizer for optimization.

SQL Scanner

The SQL Scanner identifies SQL statements from source code and database objects without executing an application. It then analyzes and categorizes the statements identified, according to their levels of suspected performance problems. The SQL Scanner includes:

- Job Manager–provides an area for adding and deleting scan jobs. Job Manager stores information on the job name and location, the number of SQL statements scanned, their start date and time, and the total processing time required to complete the scan.
- Scan–locates and analyzes SQL statements in database objects, application source code, and executable files.
- Scanned SQL Viewer–displays the SQL statements found during the scanning process. The viewer shows the scanned statement, the corresponding query plan, and any associated information useful for helping to identify statement problems. You can copy these SQL statements directly to the Syntactical SQL Optimizer for optimization.

CHAPTER 2

Getting Started

This chapter assumes that you have installed the SQL Expert using the Sybase Installer, and that the SQL Expert Monitor Server is installed and running. For more information on installing SQL Expert and the Monitor Server, see the SQL Expert Installation Guide.

For hands-on experience, go to the tutorial in Appendix A.

Logging in

When you start SQL Expert, enter your Sybase user name, password, and the Adaptive Server name.

Note if you have trouble connecting to Adaptive Server, contact your System Administrator (SA).

Your SA can define server names in the dsedit utility.

To connect SQL Expert to Adaptive Server, click Connect after entering the user name, password, and server name. For more information on installing SQL Expert, see the SQL Expert Installation Guide.

Synchronizing the data dictionary

Each time you connect to Adaptive Server, SQL Expert acquires specific database information, such as tables, indexes, and data volumes, from the data dictionary. It is important to keep this information up to date, since the SQL Expert memory uses it during optimization, SQL analysis, and scanning.

Using the Synchronize Data Dictionary option ensures that any changes to the database are directly reflected in the SQL Expert. To use this function, select Database | Synchronize data Dictionary. Synchronize Data Dictionary updates the database information in the SQL Expert memory without breaking your connection to the database.

Preferences window

Click the Preferences icon or select File | Preferences from the menu bar to open the Preferences window and choose your settings. The Preferences window is divided into eight tabs:

- Forces
- Optimization
- Optimization Quota
- Activity Log
- SQL Scanner
- SQL Classification
- Database Settings
- General

Forces tab

The Forces tab directs the Adaptive Server query optimizer toward a particular query plan. You can specify what force options to apply, using the check boxes and buttons on this tab. Force options you add to the optimized SQL statement appear in the Optimized SQL Viewer window, in purple and navy. For more information on the Optimized SQL Viewer, see “Using the Optimized SQL Viewer” on page 34. For information on the color conventions and icons in SQl Expert, consult the Help menu.

Default settings of the following parameters should handle most optimization cases.

Set

- Set Forceplan On–specifies whether to enforce the tables to be joined in the order specified in the FROM clause.
- Set Sort_Merge On/Off–for Adaptive Server version 12.0 and later. Specifies whether to consider using merge joins. If you select this parameter it toggles the server sort_merge setting.
- Set JTC On/Off–for Adaptive Server version 12.0 and later. Specifies whether to use join transitive closure. If you select this parameter, it toggles the server JTC setting.
- Set Table Count (default 1 - 8)–for Adaptive Server version 11.5 or later. Specifies the number of tables that the optimizer considers at one time while determining table join order.

For more information on using the forceplan command, see *Adaptive Server Performance and Tuning Guide, Volume 3*.

Parallel force

This force specifies whether to enforce the use of a particular degree of parallelism to access the tale, if a parallel query is enabled in the database server. For more information on parallel force, see *Adaptive Server Performance and Tuning Guide, Volume 2.*

Index

This force specifies whether to enforce the use of a particular index to access the table. For more information on indexes, see *Adaptive Server Performance and Tuning Guide, Volume 1*.

Buttons

- No Force–deselects all chosen force indexes. The “no optimization” force applies to all transformed SQL statements. You can use this option with some programming tools that do not accept force for SQL statements.
- Minimum Force–selects the Set Forceplan On and Set Sort_Merge on/Off optimization forces. “Minimum force” attempt to alter the optimization approach with a minimum of activity.

- Default Force–selects the “Set Forceplan on, Index Optimizer,” “Temp Table Generation”, and “Set Sort_Merge On/Off optimization forces. These forces are the most generally effective and often used.
- All Forces–selects all optimization forces.

Optimization tab

The Optimization tab of the Preferences window allows you to specify whether to use temporary table generation, generate an abstract plan, or to join table transformation during the optimization process. The check boxes and buttons on this page are described below.

Temp Table

Temp Table Generation allows you to choose whether to generate temporary tables during optimization. If you select Apply Selected Forces to Temp Table Generation, you can decide whether to apply selected forces to the temporary table SQL statements. For more information about temporary table generation, see *Adaptive Server Performance and Tuning Guide, Volume 2*.

Abstract Plan Group (for Adaptive Server version 12.0 and later)

For more information about abstract plan groups, see *Adaptive Server Performance and Tuning Guide, Volume 2.*

- Dump Abstract Plan–specifies whether to retrieve the abstract plan for the SQL statement. The abstract plan in not saved on the database.

  If you select this check box, the selected abstract plan group name appears on the bottom right of the main status bar.

  Third-party application users can tune SQL statements without changing source codes by dumping the abstract plan.

  Dump Abstract Plan freezes a SQL statement query plan, which guards against performance degradation through changes in data volume or database updates.
- Group Name–specifies the abstract plan group name where you want to dump the abstract plan.

- Abstract Plan Group Manager–launches the Abstract Plan Group Manager window, which you use to view, create, and modify the abstract plan group.
- Check Compatibility with Source SQL–specifies whether to check the compatibility of the optimized SQL statements with the source SQL. If you select this option, an AP Compatibility column appears on the SQL Run Time window. This column is selected if the optimized SQL statements's abstract plan is compatible with the source SQL code.

Join Table radio buttons (for Adaptive Server version 12.0 and later)

For more information about join tables, see *Transact-SQL User's Guide.*

- Join Tables with Join Type Syntax (excluding outer join)–specifies whether to join tables in the from clause with join syntax (excluding the outer join operator).
- Join Tables Without Join Type Syntax (excluding outer join)–specifies whether to join tables in the from clause without join syntax or using a comma (excluding the outer join operator).
- Both of the Above–specifies that you can join tables in the from clause, with or without join type syntax (excluding the outer join operator).

Optimization Quota tab

You can use the Optimization Quota tab of the Preferences window to restrict the SQL transformation produced during the optimization process. The quotas that appear on this tab are described below.

Rule Searching Quota (Default = 100, Range = 50 to 5,000)

This quota specifies the maximum number of SQL statements you can generate by applying transformation rules in the feedback-searching engine. The default quota of 100 is normally sufficient for most SQL statements. However, you can increase the quota if you need to rewrite exceptionally complicated SQL statements, with very high levels of table joins or multiple levels of nested subqueries.

Parallel Quota (Default = 100, Range = 50 to 99,999)

This quota specifies the maximum number of SQL statements you can generate by applying the Degree of Parallelism force. This quota is only applicable if you select the Parallel Optimizer Force check box.

Force Quota Ratio (Default = 10%, Range = 1% to 100%)

his list specifies the percentage you can use to calculate the maximum number of SQL statements you can generate by applying force indexes.

Number of forces selected

This read-only field indicates the number of forces selected. This figure is used to calculate the maximum number of SQL statements generated by applying forces.

```
(((Rules Searching Quota + Parallel Quota) * Force Quota
Ratio %)
* Number of Forces Selected ) = Force Searching Quota
```

Total Searching Quota

This figure consists of:

```
Rules Searching Quota + Parallel Quota + Force Searching
Quota
```

Table Join Permutation Quota (Default = 200, Range = 50 to 999,999)

This list shows the maximum number of table join rearrangements you can make that will generate a new table join access path during optimization.

Note The higher the quota of table join rearrangements, the longer the SQL Expert takes to optimize a complicated SQL statement.

Activity Log tab

You can use the Activity Log tab to specify whether you want to record activities, and what information you want to log.

Log directory

You can use the Activity Log directory to store the data files the log creates while it is recording your activities. The installed directory is the default directory: for example, *C:\Sybase\SQLExp-12|Pro*.

Activity to be logged

The Activity Log records activities obtained from SQL optimization and query plan retrieval. If you do not select SQL optimization or query plan generation check boxes, the log records no activities.

Note The Query Plan Generation check box in the Preferences window only logs the activity from the Query Plan function. This check box logs activity from none of the other functions, such as Optimize, Run Time, and Run Result, that retrieve a query plan.

Information to be logged

To record SQL text and query plan information, select the desired check boxes. The log records the following default information automatically: login user, OS user, database, elapsed and response time, and SQL type.

Housekeeping

The Housekeeping feature shows a warning message when the log file exceeds the default of 5MB. The file range is from 1 to 500MB.

Purge activity log

You can use this log to delete information from the activity log. Select either Whole Log, to remove all information, or specify a range of dates within which logs are deleted. Click Purge Now to remove the logs you specify.

SQL Scanner tab

The SQL Scanner tab allows you to set the parameters for the SQL Scanner module. You also set the data directory for the SQL Monitor here. For more information on the SQL Monitor see Chapter 7, “SQL Monitor.”

Data Directory

Use the Data Directory field to store the data files created while monitoring and scanning. The default is the sub-directory DATA of the installed directory; for example, *C:\SYBASE\SQLExp-12_5\Pro\DATA*. Do not make changes to this directory while SQL Scanner and SQL Monitor are active.

Note Do not change the data directory after you select it, as files created during scanning and monitoring are kept in this directory.

Scanning options check boxes

- Skip SQL Within Comments–check to specify that the scanning algorithm ignores SQL comments enclosed by /**/, and -- , when such comments appear in the source code. The default scanning algorithm searches for statements within comments, whether set off by comment symbols or not.
- Create Scanner Temp Table–check to specify that the scanning algorithm automatically creates a temporary table after scanning a create temp table SQL statement.

  When you check Create Scanner Temp Table, an ‘Include data” check box appears. Check this box to specify that the Scanner Temp Table includes data. Checking the “Include data” check box may affect your total scanning time.

  When you check Create Scanner Temp Table, an “Override previous one” check box appears. Check this box to tell the scanning algorithm to override any existing scanner temporary table when it detects a create temp table statement of the same name. This option does not override any user-defined temporary tables. See “Creating temporary tables” on page 17.
- Maximum Scanned SQL Size–indicates the maximum size in kilobytes of the SQL statement scanned. The default = 16KB, the range = 1 to 996KB.

SQL Classification tab

You can use the SQL Classification tab to analyze the SQL statement entered on the SQL Editor window. If the SQL statement entered satisfies any of the SQL classification settings,after the query plan is retrieved the statement is classified as either complex or problematic. A statement classified as problematic indicates potential performance problems and should be tuned, while a statement classified as complex is complicated and can be improved by tuning.

Simple SQL

This field is read-only, indicating the number of table scan operations references in the query plan. If the total number of table scan operations is less than this value, the SQL statement is classified as simple. This value is the same as the lower limit of the complex table scan operations range.

Complex SQL

- Number of Table Scan Operations (Default = 2,3; Rage = 2 to 99)–specifies the number of table references in the query plan for complex SQL statements. Check boxes indicate whether to include inserted and deleted simulated temporary tables in this table range.
- With Full Index Scan–check this box to specify that SQL statements with full index scan are classified as complex SQL statements.

Problematic SQL

Number of Table Scan Operations Greater Than (Default = 3)–this field is read-only, indicating the number of table scan operations references in the query plan. If the total number of table scan operations is greater than this value, the SQL statement is classified as problematic. This value is the same as the upper limit of the complex table scan operations range.

- With Full Index Scan–check to specify that SQL statements with full table scan on a table (not including inserted and deleted logical tables in database triggers), and that are greater than or equal to the defined page number, are classified as problematic SQL statements. If you do not have sa_role privileges, you cannot check the number of pages. Check boxes indicate whether to include inserted and deleted simulated temporary tables in this table range.

- With Full Table Scan (Default = 1, Range = 1 to 9999999 pages)–check to specify whether SQL statements involving a number of worktables greater than or equal to the defined number should be defined as problematic SQL statements.

Database settings tab

Database settings are general parameter settings that apply throughout SQL Expert. You must set them in the Preferences window. Five check boxes set these parameters.

Set Ansinull On

This box specifies whether ansinull should be on or off, according to whether or not you want to compare null values. For more information about ansinull settings, see *Transact-SQL User's Guide.*

Set Quoted Identifier On

This box specifies whether to allow the use of a delimited identifier ("quotation marks") for table names. For more information on quoted identifiers, see *Transact-SQL User's Guide* or *Adaptive Server Enterprise Reference Manual.*

Set Statistics Subquerycache On (Default)

This box specifies whether to display the number of cache hits, misses, and rows in the subquery cache, for each subquery in the Elapsed or Response panes in the SQL Editor and the Optimized SQL Viewer windows. For more information about these windows, see"Using the SQL Editor" on page 27, and "Using the Optimized SQL Viewer" on page 34. For more information on subquery caches, see *Adaptive Server Performance and Tuning Guide, Volume 3.*

Set Statistics Simulate On (for Adaptive Server version 12.0 and later)

This box specifies whether to load simulated statistics into the database. You can generate simulated statistics using the optdiag command, and you can use simulated, rather than actual, statistics to optimize SQL statements. For more information on simulation, see *Adaptive Server Performance and Tuning Guide, Volume 3.*

dbcc traceon (3604, 302, 310)

This box specifies whether to display the traceon information in the SQL Editor, the Optimized SQL Viewer, and the Scanned SQL Viewer windows. The traceon information displays the reasons that Adaptive Server query optimizer resolves the SQL statement in a specific plan. This option is available only if you have sa_role privileges. For more information on dbcc traceon, see *Adaptive Server Performance and Tuning Guide, Volume 3*.

General tab

The General tab allows you to decide whether to show warning messages when particular events occur.

- Show Warning When Closing Optimized SQL Viewer (default)–check this box to show a warning when you close the Optimized SQL Viewer. The default setting is On.
- Show Warning Before Saving Data (default)–check this box to specify whether you want to show a warning message when you save data (for instance, from the Run Result and Database Explorer windows.)

CHAPTER 3

Syntactical SQL Optimizer Features

Once you have logged on and set up your preferences, you can start tuning SQL statements. This chapter discusses some of the features that are useful to know about before you begin using the Syntactical SQL Optimizer.

Creating temporary tables

Creating temporary tables to extract data from permanent tables is often necessary. Because the local temporary table (#) is session-related, you have to create your temporary tables within the SQL Expert session. The User-Defined Temp Table window enables you to create temporary tables.

❖ **Creating local temporary tables:**

1. Click the User-Defined Temp Table icon or select File | User-Defined Temp Table.
2. Select the Creation tab, and enter the SQL code to create a temporary table.
3. Click Execute. After the temporary table is created, an icon appears on the bottom right of the main window status bar. This icon disappears when all User-Defined Temp Tables are dropped.

Supported SQL statements

The User-Defined Temp Table option supports only SQL statements that create or modify local temp tables (#). These are:

- SELECT INTO
- CREATE TABLE
- CREATE INDEX
- DROP INDEX
- INSERT
- UPDATE
- DELETE

Temporary tables that you create using the User-Defined Temp Table window can be used throughout SQL Expert.

Note The User-Defined Temp Table window does not support the creation of global (##) temporary tables or permanent tables.

❖ **Viewing SQL scripts**

1. Click the User-Defined Temp Table icon or select File | User-Defined Temp Table.
2. Select the Temp Table List tab.
3. Select the temporary table you want by choosing from the Temp Table drop-down list.

❖ **Removing temporary tables**

1. Click on the User-Defined TempTable icon or select File | User-Defined Temp Table.
2. Select the Temp Table List tab.
3. To remove all temporary tables, click Drop All Tables.
4. To select a specific table to drop, select it from the Temp Table drop-down list.
5. Click Drop Table.

Activity Log

You can use the Activity Log to record activities while you optimize statements and generate query plans. You can use information that the Activity Log records to review the quality of SQL statements, and to check their improvement after optimization.

Recording activities

By default, the Activity Log is disabled. To start recording your activities, follow the procedure below.

Note If you do not choose at least one option, the Activity Log records nothing.

❖ **Starting the Activity Log**

1 Click the Preferences icon or select File | Preferences.

2 Select the Activity Log tab. Choose SQL Optimization or query plan Generation, or both.

3 Specify the information that you want to log by selecting the SQL Text check box and/or the Query Plan check box.

This information is automatically recorded for each logged activity:

- PC user
- Login name
- Database name
- Server alias or data source name (DSN)
- Database user name
- *C:\Sybase\SQLExp-12_5\Pro*.

Viewing the Activity Log report

You can print or view the information stored on the Activity Log.

❖ **Viewing information**

1 Select Report | Activity Log from the menu bar. The Activity Log Report Criteria window appears.

2 Choose the information to view and click OK to generate the report.

3 To save and print the information in the report, click the Save or Print icon from the toolbar.

Viewing information

You can display any of the information from the tables below.

Table 3-1: User information

Information	Description
PC User	PC user name
Login User	Database login name
Connection Host String	Server alias or data source name (DSN)
User	Database user
Database	Database name

Table 3-2: Activity information

Information	Description
Date	Date activity executed
Activity	Type of activity recorded: OP = SQL Optimization EP = Query Plan Generation
Source SQL Type	Source SQL statement classified type: Simple, Complex or Problematic. Depends on preferences settings
Status (Elapsed Time)	Current activity status, relating to elapsed time
Source SQL Elapsed Time	Elapsed time of the source SQL statement
Best Alternative Elapsed Time	Best alternative response time
Times of Improvement (Elapsed Time)	Times of improvement in elapsed time compared with the source SQL
Status (Response Time)	Current status of the activity relating to response time
Source SQL Response Time	Source SQL response time
Times of Improvement (Response Time)	Times of improvement for response time compared with the source SQL
Best Alternative Response Time	Best alternative response time

Note The Activity Log information is retrieved from the log directory, so if you change this directory the information may not be retrieved.

❖ **Stopping the Activity Log**

1 Click the Preferences icon, or select File | Preferences. Select the Activity Log page.

2 Deselect the SQL Optimization and Query Plan Generation check boxes to disable the Activity Log.

❖ **Purging Activity Log data**

1 Click the Preferences icon, or select File | Preferences.

2 In the Housekeeping section, select either Whole Log, to remove all information, or specify a date range.

3 Select Activity Log tab.

4 Click Purge Now.

Using the Abstract Plan Group Manager

The Abstract Plan Group Manager provides a window in which you can view, create, delete, and modify abstract plan groups.

The Abstract Plan Manager provides the following functions.

- Create a new abstract plan group
- Drop the selected abstract plan group
- Rename the abstract plan group
- Export the abstract plan group to a table
- Import an exported table to a database, user and group
- Compare the differences between two groups
- Drop the selected plan
- Copy selected plan to another group
- Copy all plans belonging to one group to another group
- Modify SQL text and abstract plan according to plan ID
- Search for a keyword in the SQL text and abstract plan

Saving the abstract plan

Saving the abstract plan for a SQL statement preserves the way the Adaptive Server query optimizer executes the statement, so that even if changes to the database environment affect the statement, the query plan is unaffected.

Database environment changes include:

- Parallel degree
- Table partitioning
- Indexing
- Database software upgrades

One major advantage of saving an abstract plan is that you can optimize SQL statements without altering the SQL text. This is an ideal solution for companies that do not have application source code but want to improve the performance of the database application.

Opening the Abstract Plan Group Manager

You can open the Manager in two ways:

- Click on the Abstract Plan Group Manager icon or select File | Abstract Plan Group Manager, or
- Select Abstract Plan Group manager from the Preferences window Optimization tab.

The default Abstract Plan Group Manager appearance displays a list of group names and plan IDs that relate to the selected database. For more information about the Preferences window, see “Preferences window” on page 6.

CHAPTER 4

Syntactical SQL Optimizer

This chapter explains how you can start using the Syntactical SQL Optimizer, sometimes referred to simply as the Optimizer, to tune your own SQL statements.

Using the Syntactical SQL Optimizer

The Syntactical SQL Optimizer is an integrated module that tunes SQL statements. It analyzes the source SQL statement, then applies the feedback searching engine, which reproduces a list of semantically equivalent and syntactically correct SQL statements. You can test each SQL statement to identify the most efficient ones.

Feedback searching engine

During optimization, the feedback searching engine transforms the source SQL statement and produces a group of optimized SQL statements. It then rewrites each optimized SQL statement, producing yet another group of alternatives. The engine continues rewriting until either the SQL statements cannot be rewritten any further, or the engine reaches a set of user-defined quotas.

One searching rule is illustrated in the following example:

```
SELECT *
    FROM table_a
```

```
    WHERE table_a.key in (SELECT table_b.key
       FROM table_b)
```

If table_b.key is an indexed column, the following transformation is executed:

```
SELECT *
    FROM table_a
    WHERE EXISTS (SELECT 'x'
       FROM table_b
       WHERE table_b.key = table_a.key)
```

Although the above two SQL statements produce the same result, the database may produce two different query plans. It is difficult to decide which SQL statement will run faster without considering other factors, such as the database structure, indexes and data volume.

Common coding errors in SQL statements

You should use the SQL Scanner, the SQL Monitor, or the SQL Optimizer to check every SQL statement on the database server and in the source code. Common coding errors in SQL statements often occur when you make database changes, such as adding a new index. SQL Expert not only improves your input SQL statements, but allows you to correct common coding errors based on database information obtained during login.

Here are some examples of common mistakes found in SQL statements, and how SQL Expert might modify them.

Note The following examples of coding errors can only be detected by SQL Optimizer.

- Enable index search

  Source SQL:

  ```
  SELECT *
   FROM table_a
  WHERE NOT (table_a.key + 5) > 15
  ```

  Optimized SQL:

  ```
  SELECT *
   FROM table_a
  WHERE (table_a.key <= 15 - 5
  ```

- Remove unnecessary function calls

Source SQL:

```
SELECT *
 FROM table_a
WHERE ISNULL (table_a.key, 0) = 10
```

Optimized SQL (If A.key is a not null column detected from database):

```
SELECT *
 FROM table_a
WHERE table_a.key = 10
```

Comments within SQL statements

SQL Expert supports the three most popular delimiters for presenting comments within source code:

- single-line comments, using --
- single-line comments, using //
- block comments,

  starting with /* and ending with */

SQL Expert does not accept any other comment delimiters. If your comment delimiters follow a different format, these comments are removed during optimization and do not appear in the list of optimized SQL statements.

SQL with forces

You can enter SQL statements with optimization forces on the SQL Editor window, but they are removed during optimization.

Using the SQL Editor

Use the left pane of SQL Editor, Source SQL, to enter the source SQL statements you want to optimize, test run or run results for. Your options include:

- Optimize Using Abstract Plan–all optimization is processed to produce the optimal abstract plan. SAfter optimization, an alternative abstract plan, with the source SQL statement, appears on the Optimized SQL Viewer window. No transformed SQL statement appears. The abstract plan tab remains blank. All abstract plans are compatible with the source SQL statement.
- Optimize for Cursor–if the source SQL statement comes from a cursor declaration, or is embedded in a cursor declaration, you should select this option, which enables cursor simulation when you retrieve your query plan and runtime information.
- Use Default Plan–the SQL Expert assumes that all the bind variables have a binary datatype, and you do not need to enter a datatype for bind variables. This is useful when you want to investigate the query plan of the source SQL statement quickly.

SQL Information

The information section of the SQL Editor window displays a message if the Editor identifies your source SQL statement as potentially problematic, according to your preference settings. SQL Expert analyzes the source SQL statement after it retrieves the query plan and classifies the source SQL statement as simple, complex, or problematic.

The right top pane of SQL Editor displays a list of tabs, which show the corresponding information in the source SQL statement. Tabs include:

Query Plan

The query plan is a combination of steps the chosen by the ASE Query Optimizer to execute a SQL statement. Each of the tree diagram branches represents one way that the ASE Query Optimizer physically retrieves rows of data from the database, or prepares the data. Examining the query plan reveals the specific method by which the ASE Query Optimizer executes the SQL statement, and determines whether the source SQL statement is the most efficient.

Total estimated I/O Cost is available only when you use Adaptive Server with the “allow resource limits” configuration parameter turned on. The query plan highlights table names and the total I/O cost, if available, to make the query plan easier to read.

When you select the Optimize for Cursor check-box for a SQL statement embedded in a cursor declaration, the retrieval of the query plan is obtained by creating a stored procedure, executing it, then dropping it. For example:

```
CREATE PROCEDURE sqlexp_expain_plan_1
AS
DECLARE testrun CURSOR FOR
<Source SQL statement>
```

Abstract Plan

An abstract plan describes the query plan for a query that uses a language created for that purpose. This language contains operators, which specify the choices and actions that the optimizer can generate.

Note The Abstract Plan tab is only available for Adaptive Server 12.0 or later.

Elapsed Information and Response Information

The Elapsed Information and Response Information tabs display logical and physical reads. The SQL Editor does not display all information at all times; you choose the reads it can display when you configure the Adaptive Server for your system and choose your preferences in the Preferences window. For more information about the Preferences window see “Preferences window” on page 6. The Elapsed and Response tabs include the following reads:

- Total Actual I/O–available only when the Adaptive Server is configured with the configuration parameter allow resource limits turned on.
- Set Statistics Subquerycache On–displays the number of cache hits and misses, and the number of rows in the subquery cache, for each subquery. To see this information, you must select Set Statistics Subquerycache On in the Preferences window.
- Transact SQL–displays the actual Transact-SQL code that the server executes to retrieve elapsed or response time information.

Trace On

- The Trace On information tab shows why the SQL Optimizer has chosen a particular method for executing the source SQL statement. This tab displays the reasons for selecting specific index and table joins.

Total cost value is also highlighted on the Trace On tab.

Note The Trace On tab appears only when you select the dbcc traceon (3604, 302, 310) option in the Preferences window. You must also have privileges to use this tab. For more information, see “Preferences window” on page 6.

Entering the source SQL statement

You can open only one SQL Editor window at a time. Also, if your SQL statements use temporary tables, you must create the temporary table in the User-Defined Temp Table window before you can optimize SQL statements. For more information on this table see “Creating temporary tables” on page 17.

Click on the SQL Editor button or select Tools | SQL Editor to type in the SQL statement you want to optimize—the source SQL statement. You enter it on the left pane of the SQL Editor window by either typing it in, opening an existing file, or copying it directly from another window. Bind variables within the SQL statement can be affixed with or without an @ sign.

Bind variables

You can embed bind variables in the source SQL statement without pre-defining the datatype and value. All variable names are highlighted in red after the optimization process. SQL Expert recognizes variables declared either with or without an @ sign. During optimization, runtime, or run result, use the Parameters window to define the datatype and value of the bind variables.

❖ **Entering bind variables**

1 Select the datatype and enter the corresponding values of the bind variables.

2 To help identify the datatype and value, click the Browse button to expand the Parameters window.

3 The bottom pane displays object and data information about objects you select from the source SQL statement. To select an object, use the Objects and Columns drop-down list. If you select a column (an asterisk symbolizes all columns from the selected objects), the corresponding datatype appears.

4 To copy a compatible datatype to the selected bind variable, you can

- Click Use Datatype, or
- Click Load Data, and select the cell in the data grid that contains the variable value, and click Use Data & Datatype. Or,
- Double-click the cell in the data grid that contains the variable value, to copy the compatible datatype and value.

5 After you have selected the datatype and value of all the bind variables, click OK.

SQL Editor functions

The main functions the SQL Editor offers are:

- Query Plan
- Run for Response Time
- Run for Elapsed Time
- Run Result
- Optimize

Query Plan

To view the query plan and trace on information of the source SQL statement, click the Query Plan button on the SQL Editor window or select SQL | Query Plan. You can also press Ctrl+L.

Plan Help

The Plan Help buttons provide online help for keywords within the query plan and abstract plan. For more information on the query plan or the abstract plan, see “SQL Information” on page 35.

❖ **Using the Plan Help buttons**

1 Make sure that the query plan or abstract plan panes are activated.

2 Click the Plan Help button or select Help | Plan Help. The mouse cursor changes to a Help Select cursor (asterisk).

3 Click within the plan section of the window to display the help that applies to the keyword on the branch of the plan you select.

Plan details

The Plan Details pop-up box provides detailed information on the branch of the query plan you select, and on objects related to it. For more information on the query plan, see “SQL Information” on page 35. Click the desired branch of the query plan to see the Plan Details pop-up box, or by selecting Help | Show Plan Details.

Runtimes

There are two runtime buttons: Run for Response Time and Run for Elapsed Time. For more information about these features, see “SQL Information” on page 35, and “Understanding the SQL Run Time window” on page 43. Before you retrieve the runtime information, determine the exact purpose of the SQL statement. For example, if the SQL statement is to produce a report, use the Run for Elapsed Time button. If the SQL statement is to make an online inquiry, use Run for Response Time.

- Run for Response Time–to retrieve the response time of the source SQL Statement, click the Run for Response button or select SQL | Run for Response Time. You can also press Ctrl+I.
- Run for Elapsed Time–to retrieve the elapsed time of the source SQL statement, click the Run for Elapsed Time button or select SQL | Run for Elapsed Time. You can also press Ctrl+T. Statistical information on the run time appears on the tabs and the actual time appears in the SQL Run Time dialog box.

Note Retrieving the response time information for UPDATE, INSERT, and DELETE SQL statements is irrelevant, as these SQL statements are processed all at once. Retrieving the response time for these statements will therefore produce the same result as retrieving the elapsed time.

Run Result

To retrieve run result data from the database, click the Run Result button or select SQL | Run Result. You can also press Ctrl+R. If the source SQL statement contains a result set, the SQL Result window appears, showing all the data retrieved; otherwise, an information dialog box appears, showing the number of rows inflected.

If you are working with UPDATE, INSERT and DELETE statements, you will be prompted to either to commit or roll back affected records.

If you are working with SQL statements with an INTO clause (not a declaration of a temporary table), you are prompted to either drop or keep the table after execution.

For more information about run results, see “Run Result” on page 51.

Note While retrieving the runtime and run result you may encounter the following error, which affects UPDATE, INSERT, and DELETE SQL statements only.

The error message reads: “Can’t allocate space for object syslogs in database sqlexp because the logsegment segment is full. If you ran out of space in syslogs, dump the transaction log. Otherwise, use ALTER DATABASE or sp_extend segment to increase the size of the segment.”

This error is due to lack of space in the system table (syslogs) in which all changes to the database are recorded. Empty the transaction log in the database and re-execute.

Optimize button

To optimize the source SQL statement, click the Optimize button or select SQL | Optimize. You can also press Ctrl+M. The time to optimizing depends on the complexity of the source SQL statement and the optimization quota value. To terminate the optimization process at any time, click the Stop Optimization button. After optimization, the SQL Run Time, Optimized SQL Viewer and Optimization Details (optional) windows appear. The higher the quota value, the longer it may take to optimize a complicated SQL statement.

Optimization Details window

The Optimization Details window is optional. It shows the number of semantically equivalent SQL statements investigated, and the number of alternative query plans. It also displays a warning message if the Searching SQL Quota, the Parallel Quota, or the Join Path Permutation Quota are reached. SQL statements with the same query plan usually produce the same runtime result, because the database executes the SQL statement the same way. Therefore, newly generated SQL statements with equivalent query plans are ignored.

To show the Optimization Details window, select View | Show Optimization Details from the Optimized SQL Viewer window. If you do not want to display the Optimization Details window, disable the Show Details box on the next optimization option in the window.

Using the Optimized SQL Viewer

The Syntactical SQL Optimizer analyzes the source SQL statement entered in the SQL Editor window, and reproduces a list of semantically equivalent SQL statements. For more information on the SQL Editor window, see “Using the SQL Editor” on page 27. The transformed SQL statements are displayed on the Optimized SQL Viewer window.

Optimized SQL

The left pane of the Optimized SQL Viewer window displays the list of SQL statements produced after optimization. The arrangement of these SQL statements is the source statement, followed by optimized SQL statements beginning with SQL1, SQL2, and so on. The optimized SQL statements are ranked in order, according to increasing total estimated I/O cost: the lower the estimated cost, the better the estimated performance of the SQL statement.

If cost is unavailable, the window presents the optimized SQL statements in the order in which they were generated.

Note Cost is a recommended testing order. It does not reflect a true indication of SQL performance.

- Auto indentation format–all SQL statements are transformed into readable format by automatic indenting and alignment.
- Text color–all bind variables are red, indicating that you need to define a value or datatype on execution. Optimizer hints, such as index and parallel, are navy blue, with their associated values purple. The keywords SET FORCEPLAN ON and SET FORCEPLAN OFF are also purple. For more information about color formats, see Appendix X, Color and Symbol Keys.
- Comments–all comments are removed.

SQL Information

The top right pane of the Optimized SQL Viewer shows a list of tabs that show the source statement and its corresponding optimized SQL statements. The tabs are

- Query Plan
- Abstract Plan
- Elapsed
- Response
- Trace On

For more information on SQL Information see “SQL Information” on page 28.

Alert

An Alert message appears on the bottom right section of the Optimized SQL Viewer pane if the statement transformation is based on table constraints or indexes. Changes to table constraints and indexes can directly affect the optimized SQL statement.

For example:

```
SELECT *
 FROM table_a
WHERE ISNULL (table_a.key, 0) = 10
```

If you know that table_a.key is a not null column, using the function ISNULL has no effect on the query. It may, in fact, degrade the performance of the SQL statements, and is therefore removed.

Another example:

```
SELECT *
 FROM table_a
WHERE table_a.key = 10
```

An Alert message appears, informing you that the conversion is based on the not null constraint of column table_a.key.

Navigating optimized SQL statements

You can find the SQL statement you want in several ways:

- Click the tab set in the Optimized SQL Viewer window.
- Select the row of the SQL statement in the SQL Run Time window.
- Use the navigation buttons on the toolbar or select the corresponding menu item.

Retrieving runtime information

There are two runtime functions: Run for Response Time and Run for Elapsed Time. These functions look and behave exactly like the panes in the SQL Editor window. For information on them see “Runtimes” on page 32.

However, statistical information on the run time is displayed on the tabs, and the actual time is displayed in the SQL Run Time window. For more information on the Run Time window, see “Understanding the SQL Run Time window” on page 43. To see more information on batch runs, see “Using the Batch Run feature” on page 47.

Retrieving run result

For more information on run results, see “Run Result” on page 33. To retrieve the run result data from the database, click on the Run Result button or select SQL | Run Result, or press Ctrl+R. If the SQL statement contains a result set, the SQL Result window appears, showing all the data retrieved. Otherwise, the SQL Result window displays an information dialog box that shows the number of rows affected.

All affected records in UPDATE, INSERT and DELETE SQL statements are rolled back .

Note All created tables are dropped after execution when the SQL statement has an INTO clause.

Showing the SQL Run Time window

To float the SQL Run Time and Optimized SQL Viewer windows to the top, click on the Show SQL Run Time button in the task bar or select SQL | Show SQL Run Time from the menu bar.

Generating a report for optimized SQL statements

You can generate the details of the Optimized SQL Viewer window into a report. Select Report | Optimized SQL to open the Optimized SQL Report Criteria window. The content of the report depends on the components you select on the Optimized SQL Report Criteria window. The window displays information on two panes, the Display Information and SQL Range.

Checking abstract plan compatibility

The Abstract Plan Checker window ensures that your abstract plan and optimized SQL statements are compatible. To check compatibility, click the Abstract Plan Matrix button or select SQL | Abstract Plan Matrix. The Abstract Plan Matrix window is divided into two main sections.

The top shows a matrix of SQL sources, (SQL1, SQL2, and so forth) and abstract plans (AP0, AP1, AP2, and so forth). The Abstract Plan Matrix provides the following features:

- Check Selected–checks the compatibility of the selected abstract plan and SQL statement.
- Check All–checks the compatibility of the all the abstract plans and SQL statements.
- Check Column–checks the compatibility of the selected SQL statement with all the abstract plans.
- Check Row–checks the compatibility of the selected abstract plan with all the SQL statements.

The lower pane shows you how to display your work. To show the abstract plan, query plan, and trace on information, select the Show SQL information option.

Saving the abstract plan

To save the abstract plan you favor into the database, click the Save Abstract Plan button on the task bar, or select SQL | Save Abstract Plan from the menu bar. Select the group name, the SQL statement, and the abstract plan. Click Save. If the abstract plan is saved successfully onto the database, the corresponding plan ID number appears.

Saving the abstract plan into the database means that when the same SQL statement is executed, the query plan will be based on the abstract plan.

Sybase automatically trims the white spaces from the SQL text, replacing it with one space. You must ensure that the SQL statement you intend to execute is the same as the original SQL text you use in the saved abstract plan, and you should be extra careful with text that contains the following:

- Spaces between functions. For example, the substring

  ```
  ( EMP_NAME, 1, 5 ) = SMITH
  ```

  is not the same as the substring

  ```
  (EMP_NAME,1,5 ) = SMITH
  ```

- Spaces between database, scheme, and object name. For example, the substring

  ```
  from sqlexp . sqlexp . EMPLOYEE
  ```

  is not the same as the substring

  ```
  from sqlexp.sqlexp.EMPLOYEE
  ```

- Bind variable replacement. For example, the substring

```
where EMP_ID = @var_a
```

  is not the same as the substring

```
if @var_a = 56
where EMP_ID = 56
```

- Comments. For example, the substring

```
where EMP_ID = 123 /*comment*/
```

  is not the same as

```
where EMP_ID = 123
```

Verifying correctness

You can verify that your optimized SQL statement is correct by comparing it with the source SQL statement, using the Run Result feature and the No. of Records and First Record columns in the SQL Run Time window. For more information on the Run Time window, see “Understanding the SQL Run Time window” on page 43. This information enables you to see whether the optimized SQL statement provides the same results as the source SQL statement.

- Run Result–retrieves the queried records from the connected database.
- No. of Records–displays the number of records influenced by the SQL statement during the specified elapsed time. This figure should remain constant in both the source and the optimized SQL statements.
- First Record–indicates the first record retrieved. It shows either 0 or 1, for response time. This figure should remain constant in both the source and the optimized SQL statements.

Using the SQL Comparer

After you optimize your statements, SQL Comparer provides a window where you can compare two SQL statements, typically the source SQL and optimized SQL statements. You can also show query plans and abstract plans.

The SQL Comparer window is divided into two panes, for source statements and optimized statements. Each pane includes tabs at the bottom of the pane, for all optimized SQL statements.

Three check boxes at the bottom of the window allow you to show the text, query plans, and abstract plans of a specified SQL statement.

The panes on the SQL Comparer are:

- Source of Comparison–shows a list of SQL statements. Select a tab to choose the statement that is the source of comparison.
- SQL to Compare–shows a list of optimized SQL statements. The SQL Comparer automatically highlights differences between the source statements and the optimized statements in blue.

The Show SQL Text check box is selected by default, to display the source SQL text and the optimized SQL text. You can display the query plan and abstract plan by selecting the Show Query Plan and the Show Abstract Plan check boxes.

Note The Show Abstract plan option is available only if you are using Adoptive Server 12.0 or later.

Opening the SQL Comparer window

After you optimize your SQL statements and activate the Optimized SQL Viewer window, click the SQL Comparer button or select SQL | SQL Comparer from the menu. By default, the the source SQL statement tab is used as the source of comparison, and highlights the differences between the source statement and the optimized statement in blue on the SQL to Compare pane of the window. This window is not available if there are no optimized SQL statements.

Starting and stopping comparisons

When you launch SQL Comparer, it automatically highlights the differences between the two SQL statements on the SQL to Compare pane of the window. If you do not want to compare the selected SQL statements, click the Stop Compare button, or select SQL | Stop Compare from the menu. To display the comparison, click the Start Compare button or select SQL | Start Comparer from the menu.

Switching views

You can rotate the panes on the SQL Comparer window. To switch to horizontal panes, click the Horizontal Switch button or select SQL | Horizontal Switch on the menu.

To switch the view back to vertical panes, click the Vertical Switch button or select SQL | Vertical Switch on the menu.

CHAPTER 5

Retrieving Run Times and Run Results

Having obtained a list of semantically equivalent SQL statements through the optimization process, you must select the SQL statement with the best performance for your application. By executing the Run for Response Time, Run for Elapsed Time, or Batch Run functions, you can obtain the execution time of each SQL statement. You can obtain SQL results with the Run Result function.

Understanding the SQL Run Time window

The columns in the SQL Run Time window display information for each optimized SQL statement.

- Cost Order–the source SQL statement is always the first item on the SQL Run Time window grid. Optimized SQL statements are organized in ascending order, according to cost. SQL statements without cost value are ranked in the order in which they are generated

- Sybase Cost–displays the estimation of performance the Sybase cost-based optimizer provides. If you are not privileged to retrieve Sybase Cost, N/A appears in this column.

  Note Sybase Cost is the recommended testing order, not necessarily the most accurate performance indicator. The only way to be sure you have the most efficient statement is to retrieve the run time.

- AP Compatibility (for Adaptive Server version 12.0 and later)–tells you whether the abstract plan is compatible with the source SQL statement. This column appears only if you select the Check Compatibility with Source SQL option in the Preferences window.
- Elapsed Time–shows the time taken to retrieve all records from the database. If you use your SQL statement to retrieve all records, including reports or batch processes, you can select the SQL statement with the shortest elapsed time from the optimized SQL statement list.
- Total Actual I/O Cost–shows the Sybase actual I/O cost value necessary to process the elapsed time of the SQL statement. If this cost value is unobtainable, N/A appears.
- Number of Records–displays the total number of records your SQL statement influences. The figure should remain constant through both the source and optimized SQL statements.
- Response Time–indicates the length of time necessary for the SQL statement to return the first record. Some online retrieval screens, interactive applications, and processes do not retrieve all records from the SQL statement at once. In these cases, the best response time of the optimized statement is the criterion for selecting the SQL statement.
- Response Actual Cost–displays the actual I/O cost necessary to process the SQL statement response time. If the cost value is unavailable, N/A appears.
- First Record–displays the first record retrieved. The figure should be 0 or 1.

Example of a simplified test run script

```
CREATE PROCEDURE sqlexp_testrun
 @response_time int output,
 @elapsed_time int output,
```

```
 @row_count  int output,


AS
 declare @start_time datetime
< declaration of output parameters >

 select @start_time = getdate();
 select @response_time = -1
     @elapsed_time = -1
     @row_count = -1

 declare cursor testrun for <SQL statement>
 open testrun
 fetch testrun into ..

/*
/* for response time information */
select @row_count = @@rowcount
 if @@sqlstatus <> 0 begin
  select @response_time = datediff(ms, @start_time,
                           getdate())
 select @elapsed_time = @response_time
 end;
 select @response_time = datediff(ms, @start_time,
                           getdate())

 /* for elapsed time information */
 while (@@sqlstatus = 0) begin
  fetch testrun into ..
  select @row_count = @@rowcount
 end
 select @elapsed_time = datediff(ms, @start_TIME,
                         getdate())
 close testrun
```

Elapsed time and response time

Both elapsed and response times indicate the fastest-running SQL statement, but their aims are different. Elapsed time is the time needed to retrieve all the records from the query. Response time is the time needed to retrieve only the first record from the query.

Generally, if you intend the SQL statement for reports, use either Run for Elapsed Time, or Batch Run with Run Time Mode set to Elapsed Time. If you intend the SQL statement for online query, use the function Run for Response Time, or Batch Run with Run Time Mode set to Response time.

If you do not know the intent of the SQL statement, Sybase recommends using elapsed time as a performance indicator.

Retrieving elapsed time for a SQL statement

You can retrieve elapsed time from the SQL Editor window or from the Optimized SQL Viewer window. For more information on these subjects, see “Syntactical SQL Optimizer” on page 2 or “Using the SQL Editor” on page 27.

- To retrieve the elapsed time for a single SQL statement from the Optimized SQL Viewer window, click on the Run for Elapsed Time button, select SQL | Run for Elapsed Time, or press Ctrl+T. The run time result appears in the SQL Run Time window displaying the Elapsed Time, Elapsed Actual Cost and No. of Records.
- To retrieve elapsed time for the source SQL statement from the SQL Editor window, click the Run for Elapsed Time button, select SQL | Run for Elapsed Time, or press Ctrl+T. The run time result appears in a dialog window displaying the Elapsed Time, Actual Cost and Number of Records.

Retrieving response time for a SQL statement

You can retrieve elapsed time from the SQL Editor window or from the Optimized SQL Viewer window. For more information on these subjects see“Syntactical SQL Optimizer” on page 2 or “Using the SQL Editor” on page 27.

- To retrieve the response time for a single SQL statement from the Optimized SQL Viewer window, click the Run for Response Time button, select SQL | Run for Response Time, or press Ctrl+I. The run time result appears in the SQL Run Time window, displaying Response Time, Response Actual Cost, and First Record.

- To retrieve the response time for the source SQL statement from the SQL Editor window, click the Run for Response Time button, select SQL | Run for Response Time, or press Ctrl+I. The runtime result appears in a dialog window displaying Response Time, Actual Cost, and First Record.

Using the Batch Run feature

You can use the Batch Run function to retrieve the runtime of a group of optimized SQL statements. This feature is only executable from the Optimized SQL Viewer window. For more information on this window see“Using the Optimized SQL Viewer” on page 34.

Click the Batch Run button, select SQL | Batch Run, or press CtrlB. Select the batch run criteria and click OK. The Batch Run dialog box displays the runtime criteria and the runtime of the SQL statements as the times are retrieved.

The selected SQL statements are executed one by one, each one retrieving the runtime, unless terminated.

Stop Current SQL and Stop Batch buttons are available to terminate the currently running SQL statement or stop the batch run process.

Understanding the Batch Run Criteria window

Click the Batch Run button, select SQL | Batch Run, or press Ctrl B to open the Batch Run Criteria window. This window has three tabs.

Selected SQL tab

Use the Selected SQL tab to select or deselect SQL statements to execute. If you make no selections Batch Run executes all SQL statements. Select or deselect the statements you want to execute by clicking a row. The selected SQL statements appear highlighted in blue and checked.

To deselect a group of SQL statements according to the cost values, apply the SQL selection filter. Select the Apply SQL Selection Filter check box and click the SQL Selection Filter button. Cost statistics are available to help you analyze the most promising SQL statements to test.

You can deselect a source SQL statement only by deselecting the Source SQL check box in the Termination Criteria tab of the Batch Run Criteria window.

Note High cost does not necessary mean slower performance. If possible, test all alternatives.

Termination criteria

Use the Termination Criteria tab of the Batch Run Criteria window to set the termination criteria for each SQL statement while retrieving the runtime. If the current runtime of a particular SQL statement exceeds the termination time, the statement is terminated automatically to save time during testing.To define termination criteria, select one of the following options:

Source SQL option

The Source SQL option enables you to retrieve runtime for SQL statements that run faster than the source SQL statement. The source SQL statement is automatically selected if you select this option. The Or User Defined Time check box enables you to use User Defined Time as an additional criteria

Best Running Time SQL option

The Best Running Time SQL option allows you to retrieve the runtime of a SQL statement that runs faster than the current best elapsed time. The best elapsed time SQL value is not static. The Or User Defined Time check box enables you to use user-defined time as additional criteria.

User Defined Time option

The User-Defined Time option retrieves the runtime of SQL statements that take less than the user-defined time (MM:SS).

Run Without Termination option

The Run Without Termination option allows the retrieval of SQL statements' runtime without any termination criteria.

However, if you do not select Run Without Termination, runtime greater than the termination time is automatically terminated.

Selecting Termination Criteria

To set termination criteria for the batch run process, define the foundation of comparison, (Source SQL, Best Running Time SQL, or User-Defined Time), and add the percentage of delay.

For example, if the comparison time is 10 minutes and the percentage delay is 5%, then all the SQL statements executed terminate if the runtime exceeds the delay time of 10.5 minutes (10 minutes + (10 * 5%)).

The reason to add a percentage delay to the comparison time is that the runtime is measured according to the server time, which excludes the time the SQL statements needs to reach the server. To account for this time, add the percentage delay to the termination time.The delay time should fall between the minimum and maximum delay time value. If not, the closest value inside the range is used.

Run Time Mode/ Repeat Test

The Run Time Mode/Repeat Test tab of the Batch Run Criteria window is divided into two sections, Run Time Mode and the Repeat Test.

Run Time Mode

Selection depends on whether you want to retrieve the elapsed or response time.

Repeat Test

The Repeat Test allows you to select one:

- Test the first SQL statement twice and other SQL statements once. This option is suitable for normal runtime SQL statements.
- Test all SQL statements twice. This option is suitable for SQL statements that you execute often, such as data that is still in the cache memory.
- Test all SQL statements once.This option is suitable for long runtime SQL statements, as it is unlikely that all the data is still kept in the cache memory.

Viewing Batch Run Details window

The Batch Run Details window displays a summary of the runtime information for every SQL statement you execute after the batch run process is completed, if you check the Show Details on Next Batch Run check box in the Batch Run Details window. You can review the Batch Run Details window by selecting View | Show Batch Run Details when the Optimized SQL Viewer window is active. For more information, see“Using the Optimized SQL Viewer” on page 34.

Stopping a running SQL statement

Terminate the runtime process and the related database session by clicking the Stop Running button in the Getting Run Time dialog box.

Terminate the Batch Run function and the related database session by clicking the Stop Current SQL or Stop Batch button in the Batch Run dialog box.

Termination of the run time is not instantaneous, because it takes time to roll back the transaction and close all opened processes.

Unsatisfactory performance results

After optimization, you may discover that the performances of the optimized SQL statements are still not satisfactory.

First, check that the searching quotas have not been reached in the Optimization Details window. For more information, see “Optimization Details window” on page 34. If you have reached the searching quotas, increase the quota values in the Preferences window. For more information, see “Optimization Quota tab” on page 9. Optimize the SQL statements again to ensure that all the transformed SQL statements appear. Review the query plan of the optimized SQL statement, in case any you have altered the database structure, for instance by adding a new index.

Re-run the SQL statement optimization after the review.

Selecting the optimal SQL statement

Before you select the best SQL statement for your application, verify and test the optimized statements from the Optimized SQL Viewer window. Then select the SQL statement that best suits the application. Copy the SQL statement from the window and paste it to the source program, or save the abstract plan into the database.

Run Result

The Run Result function retrieves the data from the database and displays the SQL Result window, showing the information your query asked for.

Commit or rollback

Two types of action, commit or rollback, are practicable after you execute Run Result, depending on the window in which you execute the Run Result function:

- SQL Editor window–allows you either to commit or rollback UPDATE, INSERT and DELETE SQL statements. If you enter a SELECT statement with an INTO clause, not a temporary table statement, the SQL Editor window prompts you whether you wish to drop the table after execution. For more information, see “Entering the source SQL statement” on page 30.
- Optimized SQL Viewer window–all the records you affect by entering UPDATE, INSERT and DELETE SQL statements are rolled back. If you create a table in a SELECT statement with an INTO clause, that table is dropped automatically. No prompt appears. For more information, see “Using the Optimized SQL Viewer” on page 34.

Retrieving the run result

The Run Result function is available in the SQL Editor and Optimized SQL Viewer windows. For more information about these windows see “Using the SQL Editor” on page 27 and “Using the Optimized SQL Viewer” on page 34.

- Using the SQL Editor window–retrieve the run result for the source SQL statement by clicking the Run Result button, selecting SQL | Run Result, or pressing Ctrl+R.
- Using the Optimized SQL Viewer window–Select the SQL statement you want to execute with the Run Result function. Click the Run Result button, select SQL | Run Result, or press Ctrl+R. The result of the optimized SQL statements should all be the same.

Note When you enter UPDATE, INSERT, and DELETE SQL statements, you are prompted to commit or rollback inflected records.

When you enter SQL statements with an INTO clause (not declaring a temporary table), you are prompted to drop the table or keep the table after execution.

Terminating the run result

To terminate the run result process, click the Stop Running button in the Getting Run Result dialog box.To terminate a SQL statement that has a result set, you must log in to Sybase SQL Expert using the sa_role. Any login account can terminate a run result for SQL statements that do not have a result set. For example, you do not need an sa_role to terminate SQL statements containing INSERT, UPDATE, DELETE, and SELECT with an INTO clause.

CHAPTER 6

Displaying Database Object Information and Formatting SQL

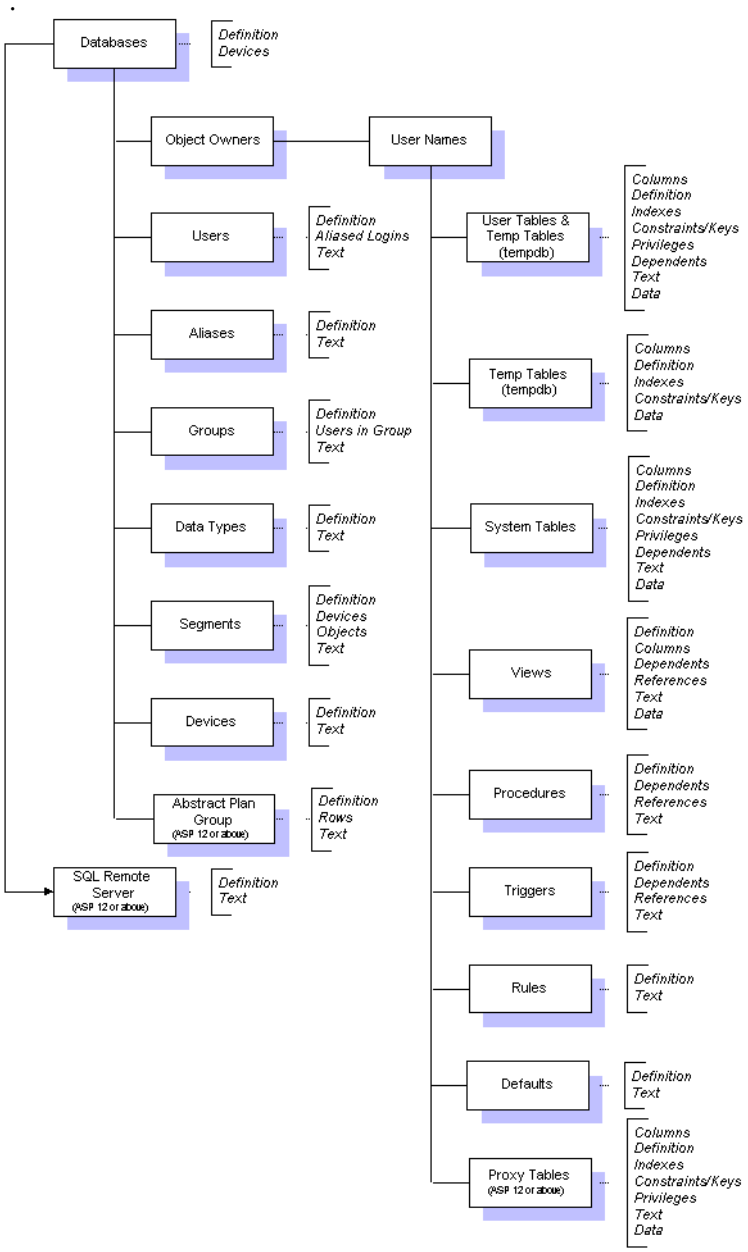
This chapter addresses the following topics:.

Using the Database Explorer

The Database Explorer allows point-and-click access to current database information. You do not need to use Transact-SQL. This information is useful during SQL statement construction.

The Database Explorer window is divided into two sections:

- Object Information–displays information relevant to the database object you select for optimizing.
- Database Objects–displays a tree diagram of all the database objects in your application.


Databases
Definition
Devices
Object Owners
User Names
Users
Definition
Aliased Logins
Text
Aliases
Definition
Text
Groups
Definition
Users in Group
Text
Data Types
Definition
Text
Segments
Definition
Devices
Objects
Text
Devices
Definition
Text
Abstract Plan Group
(ASP 12 or above)
Definition
Rows
Text
SQL Remote Server
(ASP 12 or above)
Definition
Text
User Tables & Temp Tables (tempdb)
Columns
Definition
Indexes
Constraints/Keys
Privileges
Dependents
Text
Data
Temp Tables (tempdb)
Columns
Definition
Indexes
Constraints/Keys
Data
System Tables
Columns
Definition
Indexes
Constraints/Keys
Privileges
Dependents
Text
Data
Views
Definition
Columns
Dependents
References
Text
Data
Procedures
Definition
Dependents
References
Text
Triggers
Definition
Dependents
References
Text
Rules
Definition
Text
Defaults
Definition
Text
Proxy Tables
(ASP 12 or above)
Columns
Definition
Indexes
Constraints/Keys
Privileges
Text
Data

Note In Adaptive Server you can access the syscomments system catalog table by turning on the select on syscomments.text configuration parameter:

```
sp_configure “select on syscomments.text”, 1
```

Your user privileges within the ASE server are reflected in the Database Explorer window. If you do not have access privileges to a database, the database icon is dimmed and no items are available. You must have access to the system catalog table, syscomments, to view SQL text for procedures, triggers, views, default, and rules objects. If you do not have access to the syscomments table, the message “SQL Text unavailable” appears on the Text tab.

On the Database Explorer, the current connected database or user is presented as the first item of the Database Objects tree, followed by the remaining object names in alphabetical order.

Copying column names to other windows from Database Explorer

You can copy column names to the SQL Editor or SQL Formatter windows.

Select the table or view name displayed on the tree diagram in the left pane (Database Objects) of the Database Explorer window. Select the Column tab, and select column names. Hold down the Shift key to select multiple columns.

Click the right mouse button to display a pop-up menu that provides copy choices, and select “Copy columns to SQL Editor” to copy the column names to the SQL Editor window, or select “Copy columns to SQL Formatter” to copy the column names to the SQL Formatter window.

The column names you select appear in the specified windows. A new line and a comma separate each column name. For more information on the SQL Editor or SQL Formatter windows, see “Using the SQL Editor” on page 27, and “Using the SQL Formatter” on page 56.

Copying SQL text to SQL Editor

To copy the SQL text of a database object from Database Explorer to SQL Editor, open Database Explorer by clicking the icon on the toolbar, choose the object from which you want to copy text, select the Text tab in the right pane, and click the Copy to SQL Editor button .Or select SQL | Copy to SQL Editor from the menu bar. If the object you want to copy contains multiple SQL statements, select the statement you want to tune before you use the Copy to SQL Editor option. For more information, see“Using the SQL Editor” on page 27.

Note SQL Editor accepts only SELECT, UPDATE, INSERT, and DELETE SQL statements, not Transact-SQL script.To optimize SQL statements within Transact-SQL script, retrieve the SQL statements using the SQL Scanner. Then copy the scanned SQL statements to the SQL Editor window for optimization. For more information, see Chapter 8, “SQL Scanner and Scanned SQL Viewer.”

Using the SQL Formatter

The SQL Formatter transforms a SQL statement into more readable format by automatically formatting and aligning the source code text, so that your SQL layout is consistent throughout. The SQL Formatter also checks syntax and highlights bind variables, optimizer index forces, and comments.

The SQL Formatter window is divided into two panes:

- Original SQL–provides an area where you can enter the SQL statement to be formatted.
- Formatted SQL–displays the formatted SQL statement.

Opening the Formatter window

Click the SQL Formatter button or select Tools | SQL Formatter to display the SQL Formatter window. Then enter a SQL statement into the left pane of the SQL Formatter window, either by typing it in or by pasting it from your source code, the Run Time window, the SQL Editor, or the Optimizer window.

Formatting a SQL statement

After entering the SQL statement in the SQL Formatter window, click the Format button or select SQL | Format. You can also press Ctrl+I to format the SQL statement. Before formatting, SQL Formatter checks the syntax of the SQL statement. It formats the SQL statement only if the syntax is correct. The formatted SQL statement appears in the right pane of the SQL Formatter window.

Bind variables

You can declare bind variables within the SQL statement, prefixed with the @ symbol or without. The Formatter recognizes bind variables and highlights them in red after formatting the statement.

Copying statements for tuning

To copy the formatted SQL statement to the SQL Editor window, click the Copy to SQL Editor button or select SQL | Copy to SQL Editor from the menu.

CHAPTER 7

SQL Monitor

SQL Monitor provides an easy way to capture, view, and analyze SQL statements currently running on the database server. A set of user-defined criteria determines each monitor task, or collector, used to retrieve the SQL statements you want to view.

To use the SQL Monitor module you must have sa_role privileges. If you do not have sa_role privileges and cannot grant them to yourself, log on as a user with sa_role privileges or consult your system administrator.

Before you start monitoring with the SQL Monitor Server, follow the configuration procedures and set the necessary configuration parameters.

Configuring Sybase Monitor Server

❖ **Configuring SQL Monitor**

1. Sybase Monitor Server must be installed and running. Use Sybase Central to start the Sybase Monitor Server .
2. Configure Sybase Monitor Server on the same machine as the ASE server you want to monitor.
3. Add the Sybase Monitor Server host string to the interfaces file (*sql.ini*) using the dsedit utility. This connects your machine to the Sybase Monitor Server.
4. Log into SQL Expert, using a login name with sa_role privileges to the database server.

5 Set the two parameter values that Sybase Monitor Server requires. These are event buffers per engine and maximum SQL text monitored. These parameters are discussed in the next section .

Configuration parameters

Step 5 in configuring your SQL Monitor requires you to set the two configuration parameters event buffers per engine and maximum SQL text monitored.

For details on configuring the Sybase Monitor Server to capture SQL statements, refer to *Monitor Server's User's Guide*.

Note recommended settings are not set automatically. You must reconfigure parameter values, using isql or Sybase Central, and restart the database server. The recommended settings may not be adequate for large or complicated SQL statements.

---

event buffers per engine

This parameter specifies the number of events per database server that can be monitored simultaneously.

Sybase recommends that you set event buffers per engine to:

```
sp_configure "event buffers per engine", 2000
```

max SQL text monitored

This parameter specifies the maximum size in bytes of the SQL text that can be monitored, per connection.

Sybase recommends setting max SQL text monitored to:

```
sp_configure "max SQL text monitored", 4096
```

Collector Manager window

This window allows you to choose the items you want to monitor, by adding or deleting collectors. “Collectors” collect SQL statements, by performing tasks that retrieve currently running SQL statements from the ASE server.

Opening SQL Monitor

Click on the SQL Monitor button or select Tools | SQL Monitor from the menu. The Collector Manager window appears.

Note The first time you create collectors, the Add Collector dialog box appears. For more information on the Add Collector box, see “Add Collector and Modify Collector windows” on page 62.

The Collector Manager window consists of a grid and a status bar.

The grid at the top of the window stores information on each collector, including:

- Collector Name–the unique Collector name, which the user defines.
- Description–the description of the Collector, which the user defines.
- Status–the current status of the Collector. This column remains blank until you begin the monitoring process.
- No of SQL–the total number of SQL statements collected.
- Problematic SQL–the total number of Problematic SQL statements collected.
- Complex SQL–the total number of Complex SQL statements collected.
- Simple SQL–the total number of Simple SQL statements collected.
- SQL without Plan–the total number of SQL statements collected that do not have a query plan
- Start Time–the time the monitoring process begins.
- End Time–the time the monitoring process ends.

- Server Name–the name of the Monitor Server.

Note See “SQL Classification tab” on page 13 f or definitions of Problematic, Complex and Simple SQL statements.

The status bar at the bottom of the Collector Manager window provides the following information:

- Data Directory
  The path where the Monitor saves all the data files monitoring produces. While monitoring occurs, a progress bar shows monitoring progress.
- Total Collectors
  The total number of Collectors.
- Completed
  The total number of Collectors already monitored.
- Remaining
  The total number of Collectors not yet monitored.

Adding and modifying a collector

To add a collector to the Collector Manager window, click the Add Collectors button or select Collector | Add Collector [Ins] from the menu. The Add Collectors window appears.

To modify a collector, highlight the collector row you want to modify, and click Modify Collector. You can also select Collector | Modify Collector from the menu, or click the right mouse button to display a pop-up menu from which you can select Modify. The collector name and the created date time cannot be modified.

Note If you have already monitored the collector, modifying collector criteria erases any existing information.

Add Collector and Modify Collector windows

The Add Collector and Modify Collector windows contain the same four tabs:

- General Information

- Monitor Process
- SQL Collecting Criteria
- Schedule

Select the tab page that contains the settings and information you want to display, and click OK.

General Information

The General Information tab includes:

- Collector Name–use this field to enter the name of the collector; for example, Collector3
- Description–use this field to enter the description of the collector.
- Created Date Time–this field tells you the date and time you created the collector: for example, 03-07-2001 AM 11:05:18
- Monitor Server Name–this drop-down list lets you define the host string of the Sybase Monitor Server (defined in the *sql.ini* file); for example, ASE_MS.

Monitor Process

The Monitor Process tab includes:

- Whole Server–tells the Monitor to monitor all SQL statements currently running from the ASE server.
- Process ID–tells the Monitor to monitor all SQL statements currently running that carry a specified process ID (SPID). The process ID grid displays information about all the current users and processes of the ASE server , including the SPID, login name, host name, database name and the command the server is currently executing (CMD). Select the process ID you want to monitor. To select all process IDs, click the Select All check box on the left side of the pane.
- Connection Identify–tells the Monitor to monitor all currently running SQL statements with a specified login name, host name, database name, and program name.

SQL Collecting Criteria

The SQL Collecting Criteria tab includes:

- All SQL–tells the Monitor to monitor all SQL statements currently running.
- Selected Type–tells the Monitor to monitor only complex SQL and problematic SQL statements, depending on the boxes you select.
- Table Range–specifies the numerical range of table references in the query plan for a complex SQL statement definition.
- With Full Index Scan–specifies whether SQL statements with full index scan are complex SQL statements.
- With Full Table Scan, tab, monitor; tab: With Full Table Scan, monitorWith Full Table Scan
  This check box specifies whether SQL statements with full table scan are problematic SQL statements.
- With Worktable–specifies whether SQL statements with worktables greater than or equal to the number you define in the drop-down list are problematic SQL statements.

Schedules

The Schedule tab includes:

- Start Time–specifies the time monitoring begins. The default is the current time.
- Until (Time)–specifies the time monitoring ends. The default is 5 minutes after the start time.
- Until (Duration–)specifies the duration of time spent monitoring in hours and minutes. The default is 5 minutes.
- Interval–specifies the interval time in seconds between monitor actions. The default is 3 seconds. Decreasing the interval time may affect database server performance and slow network traffic.

Note Ad hoc monitoring does not use the Start Time, Until Time, and Until Duration fields. Monitoring starts instantly, and you must terminate it by clicking the Abort Monitor button.

Query plan

The SQL Monitor shows the SQL statements it captures with their query plans, using the sp_showplan stored procedure. For more information about stored procedures, see the *Adaptive Server Reference Manual, Volume 3*.If the statement query plan is not available, the Query Plan tab on the Monitored SQL Viewer window remains blank.

Note Sybase does not guarantee that using the sp_showplan stored procedure it uses to retrieve the query plan provides a corresponding SQL statement.

Monitored SQL statement types

SQL statements captured by the SQL Monitor are classified as problematic, complex, simple, and SQL without plan.

Note The parameter settings for problematic, complex and simple SQL statements may differ for the SQL Monitor and SQL Scanner modules. For more information about the SQL Scanner see Chapter 8, “SQL Scanner and Scanned SQL Viewer.”

Problematic SQL statements

Problematic SQL statements must be tuned. To be labeled Problematic, a SQL statement must satisfy one of the following criteria:

- The number of tables referenced in the query plan must exceed the upper limit of the Complex SQL Table Range. The default value is 3.
- SQL statements must have a full table scan if you select the check box “With Full Table Scan” in the Add Collector window. The default value is 1 page.
- SQL statements must have a number of worktables greater than or equal to the value you define, if you select the check box With Worktable in the Add Collector window. The default value is l.

Complex SQL statements

Complex SQL statements are SQL statements which should be tuned.

To be labeled complex, a SQL statement must satisfy one of the following criteria:

- The number of tables referenced in the query plan falls into the complex SQL table range.
- SQL statements have a full index scan if you select "With Full Index Scan" in the Add Collector window.

Simple SQL statements

Simple SQL statements are direct, straightforward SQL statements with minimal probability of improvement.

To be labeled simple, a SQL statement must have the number of tables referenced in the query plan, which is less than the lower limit of the complex SQL table range.

SQL statements without plan

SQL without plan statements are statements for which a query plan cannot be retrieved using the sp_showplan stored procedure. For more information about stored procedures, see the *Adaptive Server Reference Manual, Volume 3*.

Monitoring

To enable the Monitor function, you must mark the collector you want to monitor. To mark a collector, click on the row you want to monitor in the Collector Manager window. A check mark appears on the first column next to the collector name. To unmark the collector, click the marked row again. Only one collector can be marked at a time.

Click on the Monitor button or select Monitor | Monitor to start monitoring. If the start time of the collector you choose has not been reached, SQL Monitor waits until it is time to begin.

During and after monitoring, the Monitor updates information in the Collector Manager window. Monitoring terminates automatically once it reaches the end time you select in the Add Collector window. Duplicated SQL statements are eliminated. This takes a few minutes.

Note If you have already monitored a collector, remonitoring overwrites your existing information.

Find collector

To search for specified text in all collectors, select Collector | Find Collector in the menu bar. Enter the line of text you want to find and click the Find button on the task bar. All Collectors and SQL number that satisfy the search criteria appear in the Find window.

Ad hoc monitoring

Monitoring on a case-by-case basis allows you to analyze the SQL statements currently running without worrying about start, end, and duration time. Right-click on an existing collector to display the pop-up menu and click Ad Hoc Monitor. Monitoring continuies until you click Abort Monitor or select Monitor | Abort Monitor.

Note If you have already monitored the selected collector, ad hoc monitoring overwrites all existing information.

Deleting marked collectors

To delete collectors from the Collector Manager window, select the collectors you want to delete and select Collector | Delete Collector from the menu bar, or press the Delete key on your keyboard.

Aborting monitor

You can abort the monitoring process by clicking the Abort Monitor button on the task bar, or by selecting Monitor | Abort Monitor from the menu bar.

Confirm the abort process by clicking OK. It takes several seconds for all events to close down.

Placing bookmarks in the Collector Manager

Right-click on the collector you want to bookmark, and select Add/Remove Bookmark from the pop-up menu that appears. If the row contains a bookmark already, the row displays in fuchsia. You can place multiple bookmarks in the Collector Manager window.

Using the Monitored SQL Viewer

The Monitored SQL Viewer window enables you to view and analyze monitored SQL statements. The window displays the unformatted monitored SQL statement, query plan, server process ID, login name, capture time, and database name.

The Monitored SQL Viewer window is divided into four sections:

- Drop-down list– displays the current collector name and a list of those collector names that satisfy the view criteria.
- Monitored SQL–displays the monitored SQL statements from a particular collector. The SQL statement is unformatted, and the display depends on how the Monitor retrieves the statement from Monitor Server. The SQL Monitor Server may capture statements that are entirely Transact- SQL and stored procedures.

  You cannot copy these SQL statements directly to the SQL Editor for optimization. Instead, use SQL Scanner to identify SELECT, UPDATE, INSERT, and DELETE SQL statements.
- SQL Information–if the monitored SQL statements' query plan is available, the Monitor retrieves and displays it on the top right pane of the Monitored SQL Viewer window. The Query Plan tab displays the query plan.

- Information–displays the server process ID, login name, capture time, and database name of the SQL statement currently monitored. If the monitored SQL statement is one of the SQL statement types, SQL type information also appears: problematic, complex, simple, and SQL without plan.

Retrieving a query plan

The Viewer attempts to retrieve a statement's query plan from the database server, using the sp_showplan stored procedure. This procedure's parameters include *process ID*, *batch ID*, *context ID* and the *statement number* of the monitored SQL statement. If the query plan information is unavailable, the Query Plan tab remains blank.

Note Sybase does not guarantee that the query plan you retrieve with the sp_showplan stored procedure corresponds to the SQL statement. Sybase recommends that you use the SQL Scanner module to analyze monitored SQL statements.

Opening the Monitored SQL Viewer window

You can view captured SQL statements through the Monitored SQL Viewer window by clicking on the Monitored SQL Viewer button, selecting Collector | Monitored SQL Viewer from the menu, or by double-clicking on the collector you want to view.

Viewing specified SQL statement types

When you first open the Monitored SQL Viewer window, the default view shows all the SQL statements found by the specified collector. However, you can restrict the SQL statements shown. Select specific SQL statement types by selecting the item you want under the View drop-down menu. The Monitored SQL Viewer windows' title bar displays the option you choose. You can view one or more types of SQL statement by selecting or deselecting menu items. The drop-down list contains all the jobs within your chosen SQL type:

- Simple SQL
- Problematic SQL
- Complex SQL

- SQL Without Plan
- All SQL

Collector navigation

There are several ways to navigate through the collectors in the Monitored SQL Viewer window:

- In the Collector Manager window, double-click on the collector you want to view.
- Select your collector from the drop-down list.
- Use the navigation buttons on the toolbar, or select the appropriate menu item.

Finding SQL statements with keywords

The Find SQL option locates SQL statements that contain a specified word for a particular collector. Select SQL | Find SQL to open the Find SQL window. Enter the word you want to search for, and click Find. To continue searching for the same words select SQL | Find Next SQL, or press Ctrl+F3.

Generating reports

SQL Monitor allows you to generate a report of the Monitored SQL Viewer window. The contents of the report depend on your requirements. Select Report | Monitor SQL from the Monitored SQL Viewer window to open the Report Criteria window. Select your criteria in the Report Criteria window, and click OK to generate the report. You can save and print the information in the report by clicking the Print button on the window.

Copying monitored statements to SQL Editor

Once you have identified the SQL statement that needs improvement, you can copy it to the SQL Editor window for optimization. Click the Copy to SQL Editor button or select SQL | Copy to SQL Editor from the menu.

A SQL statement captured by SQL Monitor consists of Transact-SQL statements (for example, DECLARE, BEGIN, and END) or stored procedures (for example, sp_help). You cannot copy directly to SQL Editor for optimization. Sybase recommends that you use the SQL Scanner to identify the SELECT, INSERT, UPDATE, and DELETE SQL statements and then analyze which ones need tuning. .

Identifying SQL statements with SQL Scanner

SQL Monitor offers you a way to identify problematic SQL statements within the database server by analyzing the retrieved query plan. However, Sybase does not guarantee that the query plan retrieved from sp_showplan corresponds to the SQL statements.

CHAPTER 8

SQL Scanner and Scanned SQL Viewer

The SQL Scanner detects potentially problematic SQL statements in source codes and database objects, without executing them.

Typically, SQL statements are reviewed and tuned manually. A database administrator may have the tedious task of browsing through thousands of lines of source code to locate SQL statements and analyze them for performance efficiency.

SQL Scanner can analyze a group of source codes or database objects without user intervention. Once the problematic SQL statements have been identified by SQL Scanner, the database administrator can copy them to the Syntactical SQL Optimizer to determine the best method of rewriting them.

Using the Group Manager and Job Manager windows

The first windows you see when you open SQL Scanner, by clicking the SQL Scanner icon or selecting Tools | SQL Scanner, are the Create Groups window and the Group Manager window. Before you can identify statements to optimize and assign them to specific jobs, you must create groups.

Using the Group Manager window

Before you can display the Job Manager window, you must create a group, through the Group Manager window.

Use the Group Manager window to create, open, delete and rename groups. It also displays a list of existing groups. You must open a job group to display the Job Manager window.

When you open SQL Scanner, you see the Group Manager window. If you have not yet created a group, the Create Group dialog box appears. Create a new group or select a working group, and click Open. The Job Manager window automatically appears when you select a working group.

The Group Manager provides the following functions, as buttons:

- Open–opens the selected group and displays it in the Job Manager window.
- Create–opens the Create Group window and allows you to create a new group.
- Modify–renames or modifies an existing group
- Delete–deletes the selected group and all the files it contains.
- Close–closes the selected group and the Group Manager window.

Note SQL Expert stores all group information as files in the data directory you define in the Preferences window.

When you create or select a working group, the Job Manager window opens. Its two components, the grid and the status bar, supply the information you need to review a specific scanning job.

If you have the Job Manager window open, you can open the Group Manager window to open, create, delete, or modify a group without closing the Job Manager window. Click on the Group Manager button, select Group | Group Manager, or press Ctrl+G.

Using the Job Manager window

The Job Manager window provides a work area for SQL Scanner, where you add different scan jobs. These items, or jobs, can be database objects, abstract plan groups, text/binary files, SQL Monitor collectors, or (SCLD) SQL files. Each job has an associated database name and user name for connecting to the database while scanning.

Note Before you can add jobs, you must first define a group or work area to work from. See "Using the Group Manager window" on page 74.

The Job Manager window consists of two components: the grid and the status bar.

The grid in the Job Manager window comprises most of the window, and displays information on each job in the group.

- File/Database object–displays the job name and its associated database and user.
- Status–displays the status of the job the Scanner is currently addressing. This column remains blank until the job is scanned.
- Valid SQL–displays the number of valid SQL statements found in the source file. Valid SQL statements are syntactically correct statements that are recognized by Sybase SQL Expert and for which Sybase can provide an query plan. Valid SQL statements are classified as simple, complex and problematic SQL statements.
- Problematic SQL–displays the number of problematic SQL statements in the current job.
- Complex SQL–displays the number of complex SQL statements in the current job.
- Simple SQL–displays the number of simple SQL statements in the current job.
- File Size–displays the size of the source file in bytes.
- Started At–displays the start date and time of scanning, in the format MM DD YYYY HH: MM: SS: for example, 03-07-2000 PM 3:00:00
- Processing time–displays the total time required to scan the job.

The Job Manager window status bar, at the bottom of the window, provides the following information:

- Data Directory–the path that shows you where the Scanner saves all the data files scanning produced.
- Group–the name of the open group, which is the group that contains the jobs you are currently scanning.
- Total Jobs–the total number of jobs in the open group.
- Completed–shows the number of jobs already scanned.
- Remaining–shows the number of jobs not yet scanned.

Adding jobs to the Job Manager window

To open the Add Jobs window, click the Open button on the Group Manager window. The Jobs Manager window appears. The Add Jobs icon appears on the toolbar, and the Group menu appears on the menu bar. You can select Group | Add Jobs to display the Add Jobs window, or click the Add Jobs icon.

The Add Jobs window contains these tabs:

- Database Objects–scan all SQL statements in database objects, including views, procedures, triggers, rules, and defaults. Depending on your database access privileges, you can select available database objects by their object name, object type, or user name. Click each branch name to expand or collapse the associated list of items. To add a database object to a group, highlight the user name, object type, or object name and click the > (forward) button. You can also drag the item to the Selected Objects pane on the right. To remove an item from the Selected Objects pane, select it and click the < (backward) button, or drag the item back to the tree diagram in the Database Objects pane on the left.

  Note You must have access privileges to the syscomments system table to scan database objects.

- Abstract Plan Group (for Adaptive Server 12.0 and later)–scan SQL statements that belong to a particular user and abstract plan group. The list of abstract plan groups available varies, depending on whether or not you select “Show only groups with SQL.”

Select the desired abstract plan group and click the > (forward) button, or select and drag the item to the list pane on the right. To remove an item from the list pane, click the < (backward) button, or select and drag that item out of the list pane back to the pane on the left.The Use Database and Set User drop-down lists allow you to switch to a different database or user when scanning.

- Source Codes–SQL Scanner can scan any source code stored as text or a binary files. To add source code files, click the Source Codes tab. To scan, select the file type and click Browse

File types include:

- Text/Binary–source code saved as text or binary format with embedded SQL statements.
- SCLD–files with dynamic SQL statements within quoted strings on a single line.
- COBOL–COBOLsource files.
- SQL Monitor Collectors–give a better understanding of which SQL statement causes a performance problem in the server when you add the collectors to SQL Scanner. SQL Scanner identifies SELECT, INSERT, UPDATE, AND DELETE SQL statements with their query plan and traceon dbcc information if it is available.

  To add SQL Monitor collectors, click the SQL Monitor Collectors tab. The pane on the left displays a list of available collectors. If you have not yet created collectors, this pane is blank. Click the > (forward)

  The Use Database and Set User drop-down lists allow you to switch to a different database or user when scanning.

Placing bookmarks in Job Manager window

Right-click the row you want to bookmark from the pop-up menu that appears, and select Add/Remove Bookmark. If the row already contains a bookmark , the complete row displays in fuchsia. You can place multiple bookmarks in the Job Manager window.

Eliminating duplicate SQL statements

To eliminate duplicate SQL statements, right-click on a job, then select Modify. Select from the pop-up menu that appears. Select “Eliminate duplicate SQL” and click OK . By default, this option is not selected.

Marking and scanning

You must mark a job before the Scanner can scan it. Click the row you want to scan. A check mark appears on the first column of the Job Manager window grid. If no jobs are checked, the Scanner does not start scanning. To deselect a job, click the marked row. The check mark disappears.

To begin scanning, click the Scan button or select Job | Scan from the menu.

During and after scanning, the Job Manager window updates information, replacing the check mark with another mark that shows it has started scanning the marked job. The time the Scanner takes to scan a job depends on your processor speed, the size of the file, and the number of valid and invalid SQL statements.

You can abort the scanning process by clicking the Abort Scan button on the task bar or selecting Job | Abort Scan from the menu bar.

Marking and unmarking all jobs

To mark all jobs in the Job Manager window, select Group | Mark All from the menu, or press Ctrl+M. To unmark all marked jobs, select Group | Unmark All from the menu, or press Ctrl+U. A check mark appears to indicate that the job is marked. The check mark, or any other appended symbol (such as the one that shows a job scan in progress) disappears, to indicate that the job is now unmarked.

Deleting marked jobs

To delete jobs from the Job Manager window, mark the jobs you want to delete and select Group | Delete Marked Jobs. You can also press the Delete key.

Scanning temporary table SQL statements

If scanned SQL statements create or modify a local (#) temporary table during scanning, these SQL statements execute automatically if you select Create Scanner Temp Table in the Preferences window. The Scanner Temp Tables you create are dropped when the job is scanned. If you check Create Scanner Temp Table in the Preferences window, the SQL statement is executed as it is scanned.

A list of the SQL statements that create temporary table SQL statements follows:

- CREATE TABLE
- SELECT INTO
- CREATE INDEX
- UPDATE
- INSERT
- DELETE

Note SQL Scanner does not override a User Defined Temp Table. The source code below provides examples.

Original source code:

```
select EMP_ID, EMP_NAME
 into #a
 from EMPLOYEE

select *
 from #a
```

After scanning:

SQL1:

```
select EMP_ID, EMP_NAME
 into #a
 from EMPLOYEE
```

SQL2:

```
select *
 from #2
```

With Scanner Temp Table:

```
select EMP_ID, EMP_NAME
 into #a
 from EMPLOYEE
```

Aborting a scan

You can abort the scanning process by clicking the Abort Scan button on the task bar or by selecting Job | Abort Scan. Abort Scan may take a few seconds to complete, as time is needed to close down all the necessary processes once the abort process is confirmed.

Finding a job

To search all jobs for a specified text, select Job | Find Job from the menu bar. Enter the text you want to search for and click Find. All jobs and SQL numbers that satisfy your search criteria are listed.

Viewing group summaries

As long as the Job Manager window is open, you can view the group summary. The Group Summary window provides a statistical list of all the jobs and SQL statements scanned within the specified group. Select Report | Group Summary from the menu bar to display the summary information.

Moving and copying jobs

To move or copy jobs in the Job Manager window into another group, mark the jobs you want to move or copy and select Group | Move to Other Group or Group | Copy to Other Group. Then select the group you want to move the job to or copythe job to from the drop-down list. Click OK after you make your selection.

Switching to another database or user

To switch to another database or user without interrupting the scanning process, right-click, then select Change Database or Change User from the pop-up menu.

Note You cannot change the specified database and user if you are scanning a database object.

Supported SQL statements

You usually find two types of SQL statement in source code.

- Embedded SQL statements–SQL statements within the program source code. These statements are constructed when you compile the program. SQL Scanner scans these statements immediately.
- Dynamic SQL statements–SQL statements you construct at runtime. You must capture these statements with the SQL Monitor modules before the Scanner can review them. The only dynamic SQL statements the Scanner can review are Single Command Line Dynamic (SCLD) files.

Scannable files

The following files and database objects are supported.

- database objects–if your user privileges allow access to syscomments system tables, you can scan views, procedures, triggers, rules and defaults.
- abstract plan groups (for Adaptive Server version 12.0 and later)–identify the SQL statements that relate to the abstract plan group.
- SQL Monitor collectors–the collectors you create in SQL Monitor, to ensure the accuracy of the query plans governing monitored and dynamic SQL statements.
- text/binary files–contain embedded SQL statements.

- SCLD SQL Files–contain Single Command Line Dynamic (SCLD) SQL statements, which are dynamic SQL statements that are saved in the source code on one command line and within quoted strings. Java and Visual Basic source codes are examples of such files.

  For example:

  ```
  SQL1:= “SELECT emp_id, ” + variableA + “ FROM
  employee”;
  ```

  SCLD SQL statements are the only dynamic SQL statements the Scanner currently supports.

Scanner statement types

The Scanner reviews SQL statements on the basis of whether they are valid or invalid, and if valid, of what type: simple, complex, or problematic.

All SQL statements the Scanner retrieves are syntactically correct. After the Scanner identifies the SQL statements, it analyzes each statement to retrieve an query plan. If it successfully retrieves the statement’s query plan, it classifies the SQL statement as valid. If the Scanner cannot retrieve a query plan for a statement, the statement is classified as invalid.

Valid SQL statements are simple, complex or problematic. These definitions depend on user-defined parameters you set in the Preferences window. For more information about these criteria see “SQL Classification tab” on page 13.

Note

Problematic SQL statements must be tuned. They are classified as problematic if they satisfy certain criteria, defined in “SQL Classification tab” on page 13.

Complex SQL statements are complicated, and can be improved. Complex SQL statements satisfy criteria defined in “SQL Classification tab” on page 13.

Simple SQL statements are unlikely to be improved. They have fewer than the lower limit of the Number of Table Scanned Operation range (default value 1) referenced in their query plan.

Invalid SQL statements

In some cases, a scanned SQL statement may be declared invalid, for one of the following reasons:

- No privilege to tables or view–the user does not have privileges to the tables or views referenced, even though the syntax of the SQL statement is correct. You can declare a database and user to scan for SQL statements in source code, but if you switch to an account without the necessary privileges the scanned SQL statements may be classified as invalid.
- Dynamic SQL statements (not SCLD SQL statements)–SQL Scanner can identify only SCLD SQL statements.

About the Scanned SQL Viewer window

The Scanned SQL Viewer window displays the formatted scanned SQL statement, its query plan, abstract plan and dbcc traceon information, and general information about the statement. If the statement is valid, this information includes the classification type of the SQL statement and any conversion you apply. If the statement is invalid, an error message appears.

The Scanned SQL Viewer window is divided into four sections:

- Drop-down list– displays the job description of the selected job, and a list of job descriptions that satisfy the view criteria.
- Scanned SQL–displays the scanned SQL statement for a particular scan job. If the SQL statement is valid, the scanned SQL statement complies with the SQL Expert's indentation algorithm. If the statement is invalid, the scanned SQL statement is unformatted.
- SQL Information– displays the following tabs:

  Query plan– retrieves the query plan for the SQL statement. The query plan is the combination of steps that the RDBMS optimizer selects.

  Trace on–displays the trace output. The trace on tab shows the index and table joins selection. dbcc traceon (3604, 302, 310)

Scanner Temp Table–appears only if the scanned SQL statement requires a temporary table. The tab displays the temporary table SQL statement that is assumed to have created the temporary table that the scanned SQL statement uses.

Note The Trace On tab page displays only if you have selected the dbcc traceon (3604, 302,310) option in the Preferences window and have sa_role privileges.

- Information–displays information relating to the scanned SQL statement. This may include the SQL statement type: problematic, complex or invalid SQL statement.

  If SQL Expert has added conversions to the scanned SQL statement to generate an query plan, the conversion information appears in the Information section of the Scanned SQL Viewer window.

Using the Scanned SQL Viewer

The Scanned SQL Viewer window allows you to view a particular type of SQL statement, navigate through the jobs, find a statement with a keyword, and perform other necessary tasks.

Opening the Scanned SQL Viewer

After you scan the job and find the specified SQL statements, you can view them through the Scanned SQL Viewer window by clicking on the Scanned SQL Viewer button, or by selecting Job | Scanned SQL Viewer from the menu. You can also double-click on the job you want to view.

Viewing a specific type of SQL statement

When you first open the Scanned SQL Viewer window, it displays by default all SQL statements found for the selected job. However, you can restrict the SQL statements shown by selecting the type of SQL statement. Go to the View menu and select the appropriate menu item. The title bar in the Scanned SQL Viewer window displays the option you choose. You can view one or more types of SQL statement by selecting or deselecting menu items. The drop-down list contains all jobs using the view criteria you choose.

- All SQL
- Simple SQL
- Problematic SQL
- Complex SQL
- Invalid SQL

To navigate through the jobs in the Scanned SQL Viewer window:

- In the Job Manager window, double-click on the job you want to view.
- Select the job from the drop-down list.
- Use the navigation buttons on the toolbar or select the corresponding menu item:

To navigate through the SQL statements in the Scanned SQL Viewer window:

- In the Scanned SQL Viewer window, click on the appropriate tab.
- Use the navigation buttons on the toolbar or select the corresponding item from the menu.

Finding statements with a keyword

The Find SQL option enables you to locate the SQL statements that contain a specified word for a particular job. Select SQL | Find SQL to open the Find SQL window, enter the word you want to search for, and click Find. To continue searching for the same word, select SQL | Find Next SQL, or press Ctrl+F3.

Generating a report

You can generate a report of the Scanned SQL Viewer window. The contents of the report depend on your requirements. Select Report | Scanned SQL on the menu bar to open the Report Criteria window. Click OK to generate the report. You can save and print the information in the report after the report is generated.

Copying to SQL Editor

After you identify the SQL statement that needs tuning from the Scanned SQL Viewer window, copy the scanned SQL statement to the SQL Editor window for optimization. Select the SQL statement and then click the Copy to SQL Editor button, or select SQL | Copy to SQL Editor from the menu.

SQL Scanner conversions

You can turn a scanned, invalid SQL statement into a valid standalone SQL statement by applying one of several possible conversions. If a conversion has been applied to the SQL statement, a message appears on the Scanned SQL Viewer window under the Information panel.

Note If you apply a conversion to a scanned SQL statement, you may need to reverse the changes when you paste it back into the original source code after optimization.

Trigger conversion

Two logical tables store deleted and inserted records as triggers, but deleted and inserted logical tables cannot be referenced outside the trigger body. Therefore, to simulate SQL statement execution, you can create two temporary tables to simulate the inserted and deleted tables.

Example:

Scanned SQL statement:

```
INSERT INTO EMP_SMALL (EMP_ID,
```

```
            EMP_NAME,
            EMP_SALARY)
SELECT A.EMP_ID,
    A.EMP_NAME,
    B.EMP_SALARY

FROM EMPLOYEE A,
    Inserted B
WHERE A.EMP_ID = B.EMP_ID
```

After conversion:

```
SELECT *
 INTO #inserted_sim
 FROM dbo.EMPLOYEE
 WHERE 1 = 2

INSERT INTO EMP_SMALL (EMP_ID,
            EMP_NAME,
            EMP_SALARY)
SELECT A.EMP_ID,
    A.EMP_NAME,
    B.EMP_SALARY
FROM EMPLOYEE A,
    #inserted_sim B
WHERE A.EMP_ID = B.EMP_ID
```

External Parameter conversion

Some source codes use a question mark (?) to define external parameters. Therefore, to enable unique referencing the Scanner changes the name of the variable so that it becomes unique within the SQL statement.

Example:

Scanned SQL statement:

```
SELECT EMP_ID
 FROM EMPLOYEE
 WHERE EMP_ID = ?
  AND EMP_NAME = ?
```

After conversion:

```
SELECT EMP_ID
 FROM EMPLOYEE
 WHERE EMP_ID = ?1
 AND EMP_NAME = ?2
```

Local Variable conversion

This conversion works for SCLD SQL files only; SQL Scanner does not support other dynamic statements. If the Scanner detects a local variable it encloses the variable name with an ampersand (&).

Example:

Scanned SQL statement:

```
“SELECT “ + VEMPID + ” FROM EMPLOYEE WHERE EMP_ID > 100”
```

After conversion:

```
SELECT &[VEMPID]
 FROM EMPLOYEE
 WHERE EMP_ID > 100
```

Cursor Query Plan conversion

This converts SQL statements found inside a cursor declaration. The Scanner does not change the SQL statement, but it embeds retrieval of the query plan in a cursor declaration.

Scanned SQL statement:

```
SELECT *
  FROM EMPLOYEE
FOR READ ONLY
```

Into Clause conversion

Some front-end tools use the INTO clause of a SQL statement for variable assignment, which violates the syntax of the INTO clause. Therefore, the Scanner inserts a comment into an INTO clause in the SQL statement with more than one variable.

Scanned SQL statement:

```
select EMP_ID, EMP_NAME
 into a, b
from EMPLOYEE
```

After conversion:

```
select EMP_ID,
 EMP_NAME /* into a, b */ /* Commented by
             SQL Expert */
 from EMPLOYEE
```

APPENDIX A

SQL Expert Tutorial

This chapter can be used as a starting point for using the SQL Expert. It is not a complete step-by-step tutorial and does not cover all functionalities. Each feature is described from the first step, independent of the steps described in a previous section.

Using the Syntactical SQL Optimizer

This tutorial enables you to:

- Tune a SQL statement
- Retrieve the query plan of a SQL statement
- Examine the results of the SQL statement chosen
- Tune a SQL statement using the abstract plan
- Tune a SQL statement that uses temporary tables

Note Support for tuning SQL statements using abstract plans and temporary tables is limited to Adaptive Server version 12.0 and later.

Tuning a SQL Statement

Usually, the statements you tune are problematic or, if you have time, complex.

❖ **Tuning a SQL statement**

1. Open the SQL Editor window by clicking the SQL Editor button or selecting Tools | SQL Editor.
2. Enter the SQL statement you want to tune on the left pane of the window.

3 Click Optimize or select SQL | Optimize.

4 After optimization, three windows appear:

- Optimization Details
  This window displays the total number of semantically equivalent SQL statements investigated, the number of SQL statements with alternative query plans, and a warning message if the number of SQL transformations has reached the defined quota values.
- SQL Run Time
  This window displays runtime information. Sybase Cost is the estimated cumulative cost. The smaller the cost the lower the estimated cost. However, the Sybase Cost is merely

  a recommended testing order, and does not always provide a true indication of performance.
- Optimized SQL Viewer
  This window displays the semantically equivalent SQL statements and their query plan, response time, and elapsed time information.

5 Click OK on the Optimization Details window.

6 In the Optimized SQL Viewer window, look at the tabs on the bottom pane, labeled SQL1, SQL2, and so forth. Click these tabs to see the semantically equivalent SQL statements.

7 In the Optimized SQL Viewer window, look at the pane on the right (SQL Information) to see the query plan for the SQL statement.

8 In the SQL Run Time window, the Sybase Cost estimates how the SQL statement performs. You should test run each statement, however, to see how it actually performs.

9 To test all the SQL statements, click Batch Run, or select SQL | Batch Run.

- In the Batch Run Criteria window, select the SQL statements to run in a batch. (The default selection is all SQL statements.) Review the settings on the Termination Criteria tab or the Run Time Mode & Repeat Test tabs.
- If you need both the elapsed time (the time taken to retrieve all records) and the response time (the time taken to retrieve the first record), run Batch Run twice. The first time you execute Batch Run, you can test the response time, and the second time you can change the runtime mode to test for elapsed time.

10 Click OK.

11 After running all the SQL statements, the Batch Run Details window appears with the details of each SQL statement.

12 Click OK. The runtime results appear on the SQL Run Time window. Select the SQL statement that best suits your application needs.

13 From the Optimized SQL Viewer window, you can copy the tuned SQL statement back into your original source code.

Retrieving the query plan for a SQL statement

The query plan is a specified combination of steps that the SQL Optimizer chooses to execute the SQL statement.

❖ **Retrieving the query plan**

1 Open the SQL Editor window by clicking the SQL Editor button, or selecting Tools | SQL Editor.

2 In the left pane, enter the SQL statement whose query plan you want to see.

3 Click the Query Plan button or select SQL | Query Plan.

4 After the SQL Editor makes syntax checks, the query plan of the SQL statement appears on the right pane of the SQL Editor window.

5 If you need more information, Plan Help provides on-line help for keywords in the query plan.

- Check that the right pane(SQL Information) of the SQL Editor window has a specified query plan.
- Click the Plan Help button or select Help | Plan Help. The cursor changes to a question mark.
- Click the appropriate branch of the query plan to display online help relating to the keyword you select.

Examining the result of a SQL statement

You can execute a single SQL statement and retrieve the result from either the SQL Editor window or the Optimized SQL Viewer window.

❖ **Examining the result from the SQL Editor window**

1. Open the SQL Editor window by clicking the SQL Editor button or by selecting Tools | SQL Editor.
2. In the left pane of the window, enter the SQL statement for which you want the run result.
3. Click the Run Result button or select SQL | Run Result. If you enter UPDATE, INSERT, or DELETE SQL statements you are asked to commit or rollback.
4. The SQL Result window displays the result of the SQL statement.
5. Click OK to close the window.

❖ **Examining the result from the Optimized SQL Viewer window**

1. Open the SQL Editor window by clicking the SQL Editor button or selecting Tools | SQL Editor.
2. In the left pane of the window, enter the SQL statement for which you want the run result.
3. Click the Optimize button or select SQL | Optimize.
4. From the Optimized SQL Viewer window, select the SQL statement whose results you want to retrieve by clicking the appropriate tab at the bottom of the window (for instance, SQL1).
5. Click the Run Result button or select SQL | Run Result. Remember that all records influenced by UPDATE, INSERT, and DELETE SQL statements are rolled back.
6. The SQL Result window displays the retrieved result of the SQL statement.
7. Click OK to close the window.

Tuning a SQL statement using abstract plan

Abstract plan is a new feature, introduced with Adaptive Server version 12.0. This feature enables you to influence the optimization of a SQL statement without modifying the SQL statement text. This is useful for protecting the performance of the SQL statement from database changes. It is also useful for optimizing SQL statements from third-party applications, where you do not have access to the source code.

Note Before beginning this procedure, you must select "Dump abstract plan" in the Optimization tab of the Preferences window, and then type in an abstract plan group.

❖ **Tuning with abstract plan**

1 Open the SQL Editor window by clicking the SQL Editor button or selecting Tools | SQL Editor.

2 In the left pane, enter the SQL statement for which you want the run result. Select "Optimize using abstract plan only"at the bottom of the SQL Editor window.

3 Click the Optimize button or select SQL | Optimize.

4 After optimization, three windows appear:

- Optimization Details
  This window displays the total number of semantically equivalent SQL statements investigated, the number of SQL statements with alternative query plans, and a warning message if the number of SQL transformations has reached the defined quota values.
- SQL Run Time
  This window displays runtime information. Sybase Cost is the estimated cumulative cost. The smaller the cost the lower the estimated cost. However, the Sybase cost is only a recommended testing order, and does not always provide a true indication of performance.
- Optimized SQL Viewer
  This window displays the semantically equivalent SQL statements and their query plan, response time, and elapsed time information.

5 Click OK to close the Optimization Details window.

6 Look at the tabs on the bottom pane of the Optimized SQL Viewer window, labeled SQL1, SQL2, SQL3, and so forth. (In order to see these tabs you must have optimized SQL statements.) By clicking the appropriate tabs, you can see the source SQL statement with an alternative abstract plan, within the PLAN clause.

7 The query plan for the SQL statement appears in the right pane of the Optimizer SQL Viewer window.

8 In the SQL Run Time window, the Sybase Cost estimates how the SQL statement performs. You should test run each statement, however, to see how it actually performs.

9 To test run all the SQL statements, click Batch Runr or select SQL | Batch Run.

- In the Batch Run Criteria window, select the SQL statements to run in a batch. (The default selection is all SQL statements.) Review the settings on the Termination Criteria tab, or the Run Time Mode & Repeat Test tabs.
- If you need both the elapsed time (the time taken to retrieve all records) and the response time (the time taken to retrieve the first record), run Batch Run twice. The first time you execute Batch Run you can test the response time,tab and the second time you can change the runtime mode to test for elapsed time.

10 Click OK.

11 After you run all the SQL statements, the Batch Run Details window appears, showing the details of each SQL statement. Click OK.

12 The runtime results appear on the SQL Run Time window. Select the SQL statement that best suits your application needs.

13 From the Optimized SQL Viewer window, select the optimized SQL statement. Click the Save Abstract Plan button or select SQL | Save Abstract Plan. Select the abstract plan group and check that the abstract plan is correct. Select SAVE. The abstract plan is now saved to the database in the specified abstract plan group, so that the next time you execute the same SQL statement, the same abstract plan determines the query plan.

Tuning a SQL statement with temporary tables

Because the local temporary table (#) is session-related, you must create temporary tables before you can tune the SQL statement.

❖ **Tuning with temp tables**

1. To create temporary tables, click the User-defined Temp Table button or select File | User-Defined Temp Table. On the Creation tab, enter the SQL code for temporary tables and click Execute.
2. Open the SQL Editor window by clicking the SQL Editor button or selecting Tools | SQL Editor.
3. Enter the SQL statement in the left pane (Source SQL) of the window.
4. Click the Optimize button or select SQL | Optimize.
5. After optimization, three windows appear:
   - Optimization Details
     This window displays the total number of semantically equivalent SQL statements investigated, the number of SQL statements with alternative query plans, and a warning message if the number of SQL transformations has reached the defined quota values.
   - SQL Run Time
     This window displays runtime information. Sybase Cost is the estimated cumulative cost. The smaller the cost the lower the estimated cost. However, the Sybase cost is only a recommended testing order, and does not always provide a true indication of performance.
   - Optimized SQL Viewer
     This window displays the semantically equivalent SQL statements and their query plan, response time, and elapsed time information.
6. Click OK on the Optimization Details window.
7. Look at the tabs on the bottom pane of the Optimized SQL Viewer window, labeled SQL1, SQL2, SQL3, and so forth. By clicking the appropriate tabs, you can see the semantically equivalent SQL statements.
8. The query plan for the SQL statement displays in the optimized SQL Viewer window.
9. To test run all the SQL statements, click the Batch Run button, or select SQL | Batch Run.

- In the Batch Run Criteria window, select the SQL statements to run in a batch. (The default selection is all SQL statements.) Review the settings on the Termination Criteria tab or the Run Time Mode & Repeat Test tabs.
- If you need both the elapsed time (the time taken to retrieve all records) and the response time (the time taken to retrieve the first record), run Batch Run twice. The first time you execute Batch Run you can test the response time, and the second time you can change the runtime mode to test for elapsed time.

10 Click OK.

11 After you run all the SQL statements, the Batch Run Details window appears with details on each SQL statement. Click OK.

12 Runtime results appear on the SQL Run Time window. Select the SQL statement with the runtime result that best suits the needs of your application.

13 From the Optimized SQL Viewer window, you can copy the tuned SQL statement back into your original source code.

Standardizing code with the SQL Formatter

❖ **Standardizing the format of a SQL statement**

1 Click the SQL Formatter button or select Tools | SQL Formatter.

2 On the left pane, enter the SQL statement you want to format.

3 Click the Format button, or select SQL | Format.

4 After the SQL Formatter checks the statement syntax, the formatted SQL statement appears on the right pane of the window. The Formatter highlights comments, bind variables, incorrect fields, table names, and optimizer forces in different colors.

Viewing database objects with the Database Explorer

The left pane (Database Objects) of the window displays the database objects belonging to the database you are working in.

Click the Database Explorer button or select Tools | Database Explorer.

1 The left pane (Database Objects) of the window displays the database objects belonging to the database you are working in.

2 Expand the Databases branch to see a list of the objects that belong to the specified database

3 Expand the branches to view object details. Information about the database object you select appears on the right pane (Object Information) of the window.

Note Access to database objects depends on your database privileges.

Retrieving and identifying SQL statements

You can use the SQL Scanner to identify problematic and complex SQL statements without executing them, and SQL Monitor to capture and identify problematic and complex SQL statements while they are running.

❖ **Using SQL Monitor to retrieve running statements**

1 Click the SQL Monitor button, or select Tools | SQL Monitor.

2 The Collector Manager window appears, followed by the Add Collector window if you have not created a collector before. If the Add Collector window does not appear, click the Add Collector button or select Collector | Add Collector.

3 Enter a collector name and the Monitor Server host string. Check that the other settings satisfy your requirements. Define a monitoring end time on the Schedule tab on the Add Collector window.

4 Click OK.

5 Click the Monitor button or select Monitor | Monitor to begin capturing running statements.

- Details appear in the Collector Manager window grid as the information accumulates.
- The monitoring process stops at the end time you define on the Add Collector window, on the Schedule tab.

6 To view the retrieved SQL statements, select the newly added collector row and click the Monitored SQL Viewer button, or select Collector | Monitored SQL Viewer.

- Click the tabs, SQL1, SQL2, and so forth, at the bottom of the Monitored SQL Viewer window, to see the captured SQL statements.
- The query plan for each statement appears in the top right pane (SQL Information), and that the bottom right pane (Information) displays monitored SQL information that may help you identify the origin of the SQL statement.

❖ **Using SQL Monitor to tune SQL statements**

1 Click the SQL Monitor button or select Tools | SQL Monitor.

2 To view retrieved statements, select the collector you want to view. Click the Monitored SQL Viewer button or select Collector | Monitored SQL Viewer. The Monitored SQL Viewer window displays all the SQL statements monitored by default.

3 View problematic and complex statements using View | Problematic SQL and View | Complex SQL.

4 Select a SQL statement (not a Transact-SQL statement) and click the Copy to SQL Editor button or select SQL | Copy to SQL Editor. The SQL statement automatically loads into the SQL Editor window.

5 Click the Optimize button or select SQL | Optimize.

6 After optimization, three windows appear:

- Optimization Details
  This window displays the total number of semantically equivalent SQL statements investigated, the number of SQL statements with alternative query plans, and a warning message if the number of SQL transformations has reached the defined quota values.

- SQL Run Time
  This window displays runtime information. Sybase Cost is the estimated cumulative cost. The smaller the cost the lower the estimated cost. However, the Sybase Cost is only a recommended testing order, and does not always provide a true indication of performance.
- Optimized SQL Viewer
  This window displays the semantically equivalent SQL statements and their query plan, response time, and elapsed time information.

7 Click OK in the Optimization Details window.

8 Look at the tabs on the bottom pane of the Optimized SQL Viewer window, labeled SQL1, SQL2, SQL3, and so forth. By clicking the appropriate tabs, you can see the semantically equivalent SQL statement.

9 In the Optimized SQL Viewer window, look at the right pane (SQL Information) to see the query plan for the SQL statement.

10 To test all the SQL statements, click the Batch Run button, or select SQL | Batch Run.

- In the Batch Run Criteria window, select the SQL statements to run in a batch. (The default selection is all SQL statements.) Review the settings on the Termination Criteria tab or the Run Time Mode & Repeat Test tabs.
- If you need both the elapsed time (the time taken to retrieve all records) and the response time (the time taken to retrieve the first record), run Batch Run twice. The first time you execute Batch Run you can test the response time, and the second time you can change the runtime mode to test for elapsed time.

11 Click OK.

12 After you run all the SQL statements, the Batch Run Details window appears with details on each SQL statement. Click OK.

13 Runtime results appear on the SQL Run Time window. Select the SQL statement with the runtime result that best suits the needs of your application.

14 From the Optimized SQL Viewer window, you can copy the tuned SQL statement back into your original source code.

❖ **Using SQL Scanner to scan SQL statements from database objects**

1 Click the SQL Scanner button or select Tools | SQL Scanner.

Before scanning database objects or source code files, you must create a group in which you store the items you want to scan. If you are using the SQL Scanner for the first time, the Create Group window appears; if the Create Group window does not appear, click the Create button on the Group Manager window.

2 Enter a new group name. Click OK.

3 Check that your new group is highlighted in the drop-down list box; highlight it if necessary.

4 Click OK. The selected group opens in the Job Manager window.

5 Click the Add Jobs button or select Group | Add Jobs.

6 In the Add Jobs window, click the Database Objects tab.

7 Expand the user branch in the left pane drop-down list box (Database Objects) of the Database Objects window.

8 Highlight your user name and click the > (forward) button. The string

```
<user name> -> /*all*/ -> /*all*/
```

appears in the right pane.

9 Click OK in the Add Jobs window. A list of database objects appears on the Job Manager window grid, ready for scanning. Your database privileges determine whether or not you can scan all the selected database objects.

10 Click the Scan button or select Job | Scan to begin scanning. Details appear on the Job Manager window grid as the scanning process completes each job, showing you how many SQL statements are included in each database object.

11 To view the scanned SQL statements in a database object, highlight the object and click the Scanned SQL Viewer button, or select Job | Scanned SQL Viewer.

- The first SQL statement the Scanner finds in the database object appears in the left pane (Scanned SQL). Click the appropriate tabs at the bottom of the window to view other SQL statements in the database object.
- The query plan for each statement appears in the right pane (SQL Information).

❖ **Using SQL Scanner to scan SQL statements from source codes**

1 Click the SQL Scanner button, or select tools | SQL Scanner.

Before scanning database objects or source code files, you must create a group in which you store the items you want to scan. If you are using the SQL Scanner for the first time, the Create Group window appears; if the Create Group window does not appear, click the Create button on the Group Manager window.

2 Enter a new group name. Click OK.

3 Check that your new group is highlighted in the drop-down list box, and highlight it if you need to. Click to open.

4 Click the Add Jobs button or select Group | Add Jobs.

5 Select the type of source code file by clicking one of these tabs at the top of the window: Text/Binary Files, COBOL Files, or SCLD. SQL Files. Click Browse.

6 Select the files you want to scan. Click Open.

7 Click OK on the Add Jobs window. A list of source files appears on the Job Manager window grid, ready for scanning.

8 Click the Scan button or select Job | Scan to begin scanning. Details appear in the Job Manager window grid as each job is scanned.

9 To view the scanned SQL statements, highlight the object and click the Scanned SQL Viewer button or select Job | Scanned SQL Viewer .

- The first SQL statement the Scanner finds in the file appears in the left pane (Scanned SQL). Click the appropriate tabs at the bottom of the window to view other SQL statements in the database object.
- The query plan, abstract plan, and Scanner Temp Table (if you use a temporary table) for each statement appears in the top right pane (SQL Information).

❖ **Using SQL Scanner to tune scanned SQL statements**

1 Click the SQL Scanner button or select Tools | SQL Scanner.

Before scanning database objects or source code files, you must create a group in which you store the items you want to scan. If you are using the SQL Scanner for the first time, the Create Group window appears; if the Create Group window does not appear, click the Create button on the Group Manager window.

2 The specified group opens, displaying details from previously entered jobs.

3 Highlight the database object or source file whose SQL statements you want to see. Click the Scanned SQL Viewer button or select Job | Scanned SQL Viewer.

4 The left pane (Scanned SQL) of the window displays the first SQL statement in the database object or source file.

5 The name of the source file or database object appears at the top of the window in the drop-down list.

- To view only the problematic and complex statements, select View | Problematic SQL and View | Complex SQL in the menu bar.
- You can view the other scanned SQL statements in the left pane, by clicking the appropriate tabs at the bottom of the pane.

6 Select a SQL statement you want to improve. Click the Copy to SQL Editor button or select SQL | Copy to SQL Editor. The SQL statement loads automatically into the SQL Editor window.

7 Click the Optimize button or select SQL | Optimize.

8 After optimization, three windows appear:

- Optimization Details
  This window displays the total number of semantically equivalent SQL statements investigated, the number of SQL statements with alternative query plans, and a warning message if the number of SQL transformations has reached the defined quota values.
- SQL Run Time
  This window displays runtime information. Sybase Cost is the estimated cumulative cost. The smaller the cost the lower the estimated cost. However, the Sybase cost is only a recommended testing order, and does not always provide a true indication of performance.
- Optimized SQL Viewer
  This window displays the semantically equivalent SQL statements and their query plan, response time, and elapsed time information.

9 The query plan for the SQL statement appears in the optimized SQL Viewer window.

10 To test all the SQL statements, click the Batch Run button, or select SQL | Batch Run.

- In the Batch Run Criteria window, select the SQL statements to run in a batch. (The default selection is all SQL statements.) Review the settings on the Termination Criteria tab or the Run Time Mode & Repeat Test tabs.
- If you need both the elapsed time (the time taken to retrieve all records) and the response time (the time taken to retrieve the first record), run Batch Run twice. The first time you execute the Batch Run you can test the response time, and the second time you can change the runtime mode to test for elapsed time.

11 Click OK.

12 After you run all the SQL statements, the Batch Run Details window appears with details on each SQL statement. Click OK.

13 Runtime results appear on the SQL Run Time window. Select the SQL statement with the runtime result that best suits the needs of your application.

14 From the Optimized SQL Viewer window, you can copy the tuned SQL statement back into your original source code.

Synchronizing the data dictionary

Using the Synchronize Data Dictionary feature ensures that changes to the database are directly reflected in SQL Expert. Synchronize Data Dictionary does not break the connection with the database, but it updates new database information to SQL Expert memory.

Select Database | Synchronize Data Dictionary.

Glossary

This glossary describes SQL and SQL Expert terms used in this book. For a description of Adaptive Server and general SQL terms, refer to the *Adaptive Server Glossary*.

abstract plan Describes the query plan for a query using a language created for that purpose. This language contains operators to specify the choices and actions that the optimizer can generate.

Activity Log Records optimization and execution activities during daily processing.

artificial intelligence A computerized system that simulates human intelligence.

bind variables Variables whose values can be changed. Before executing a statement the database management system performs a binding action to substitute all bind variables with values provided by users, making it possible to execute the same SQL statement with different values.

capture time The time a SQL statement is captured, in the format MM DD YYYY HH MM SS.

collector A task that performs an element of statement optimization

comment indicator Special symbol used to identify comments within code.

complex SQL statement A statement whose numsber of table operations is between 2 and 99, according to the setting in the Preference window.

cursor declaration A SELECT statement defined under the DECLARE CURSOR statement.

database object Component of a database: for instance, tables and columns.

data dictionary Holds specific database information, such as tables, indexes, and data volumes.

data directory Stores the data files created while monitoring and scanning.

driving path The sequence in which tables are accessed during retrieval of a SQL statement.

elapsed time Time between the beginning and the end of a statement optimization.

feedback searching engine	Transforms source SQL statement and continues rewriting until it reaches a set of user-defined quotas.
force	Directs the SQL Expert Optimizer toward a particular query plan.
Formatter	Standardizes format of SQL statements to ensure consistency within an application.
host string	A string that indicates the type of network protocol necessary for connection and the destination Adaptive Server.
housekeeping functions	Functions within the Abstract Plan Group Manager that drop, purge, copy, select and modify a specific abstract plan.
index force	Specifies whether to enforce the use of a particular index to access a table.
job	Item scanned by the SQL Scanner, each with an associated database name and user name for connecting to the database while scanning.
job group	A group of jobs that you must create before running any specific job.
logical read	Retrieving data from cache memory.
monitor	Locates and analyzes SQL statements at run time, according to user-defined standards of performance levels.
optimization quota	Restricts the SQL transformation produced during the optimization process.
optimize	Tune a SQL statement
optimizer	An engine that estimates the cost of each permutation of table accesses in terms of CPU and I/O cost.
parallel force	Specifies the degree of parallelism used to access a table if a parallel query is enabled in the database server.
parsing	Analyzing and validating a SQL statement.
physical read	Retrieving data from disk.
problematic SQL statement	A statement that requires optimization.
query plan	A series of steps the Adaptive Server Query Optimizer provides for optimizing a statement.
response time	The time required to optimize the first SQL statement.
run time	Time required to run source and optimized SQL statements.

Run Time and Batch Run	A feature that tests both source and optimized SQL statements.
scanning options	Instructions to the scanning algorithm that specify which SQL statements to scan.
SCLD	Single Command Line Dynamic SQL statement, the only dynamic statements the SQL Expert optimizes.
simple SQL statement	A statement that probably does not need to be optimized.
source SQL statement	The original SQL statement, before tuning.
subquery cache	An area of cache in which the result set of a subquery is stored.
Sybase Cost	A figure Sybase estimates to indicate the performance of a SQL statement.
table join	A SQL statement that retrieves data from two or more tables on the same query level.
temporary table	A table generated for a specific SQL statement, that is automatically removed after the statement runs.
times of improvement	Times of improvement for response time of SQL statement compared with the response time of the source SQL statement.
trace on	Trace output provided by a server, displaying reasons for the SQL statement to be resolved in a specific way.
tuning	Rewriting a SQL statement for greater efficiency of execution.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W