Oracle® Enterprise Manager

Database Tuning with the Oracle Tuning Pack

Release 2.0

February 1999

Part No. A67827-01

ORACLE

Database Tuning with the Oracle Tuning Pack

Part No. A67827-01

Release 2.0

Contents

Part III Getting Started with Oracle SQL Analyze

3 Getting Started with Oracle SQL Analyze

4 Starting a Tuning Session

5 Gathering and Analyzing Information

6 Tuning SQL Statements

11 Creating and Working with Tuning Sessions

12 Collecting the Data

Send Us Your Comments

Database Tuning with the Oracle Tuning Pack, Release 2.0

Part No. A67827-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- electronic mail - nedc_doc@us.oracle.com
- FAX - 603-897-3317. Attn: Oracle Enterprise Manager Documentation
- postal service
  Oracle Corporation
  Oracle Enterprise Manager Documentation
  One Oracle Drive
  Nashua, NH 03062
  U.S.A

If you would like a reply, please give your name, address, and telephone number below.

If you have problems with the software, please contact your local Oracle World Wide Support Center.

Preface

The Oracle Tuning Pack provides advanced tools that focus on tuning the highest impact database performance areas, such as: application SQL, indexing strategies, instance parameters controlling I/O, SGA performance, and object sizing, placement, and reorganization. The tools included in this pack are designed to work together to accomplish many database tuning tasks. The applications included in the Oracle Tuning Pack are: Oracle Auto-Analyze, Oracle SQL Analyze, Oracle Expert, Oracle Index Tuning Wizard, and Oracle Tablespace Manager.

This preface covers the following topics:

Purpose

The purpose of this manual is to introduce you to the Oracle Tuning Pack applications. The manual describes how you can use each of the applications as part of an overall database tuning process.

Although this manual provides some guidelines for tuning your database, it does not attempt to be a definitive source for making tuning decisions. For more complete information about database and SQL application tuning, see the books listed in "Related Publications" and "Other References."

Audience

This manual is intended for database administrators, developers of Oracle-based applications, and all those responsible for the maintenance and performance of an Oracle database.

How This Manual is Organized

This manual contains the following six parts:

Part 1 **Introduction to the Oracle Tuning Pack**

This section briefly describes the individual pack applications and provides tips on choosing the right application to perform your database tuning tasks.

Part 2 **Getting Started with Oracle Auto-Analyze**

This section provides an overview of the Oracle Auto-Analyze wizard. This wizard helps you create jobs that run on a schedule within a user specified maintenance window and intelligently updates the dictionary statistics for any tables which would benefit.

Part 3 **Getting Started with Oracle SQL Analyze**

This section provides an overview of Oracle SQL Analyze. Oracle SQL Analyze helps you optimize the performance of SQL statements run against your databases.

Part 4 **Introduction to Oracle Expert**

This section provides an overview of Oracle Expert. Oracle Expert is used to automate performance tuning and generate tuning recommendations.

Part 5 **Introduction to Oracle Index Tuning Wizard**

This section provides an overview of the Oracle Index Tuning wizard. This wizard identifies tables with inefficient indexes and makes recommendations which will improve access to those tables.

Part 6 **Using Oracle Tablespace Manager**

This section provides an overview of Oracle Tablespace Manager. Oracle Tablespace Manager is used to automate tablespace management and improve performance.

Conventions

The following conventions are used in this manual:

Convention	Meaning
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
=>	Indicates that you choose a menu option. For example, File=>Exit means select the Exit option from the File menu.
[ ]	A key name enclosed in brackets indicates that you press that key. For example, [F1] means the function key labeled "F1."

Related Publications

For further details about the information provided in this manual, see the following manuals in the Oracle Server documentation set:

- *Oracle Server Concepts, Volume 1*
- *Oracle Server Concepts, Volume 2*
- *Oracle Server Reference*
- *Oracle Server SQL Reference, Volume 1*
- *Oracle Server SQL Reference, Volume 2*
- *Oracle Server Tuning*

Oracle Enterprise Manager Documentation

The *Oracle Enterprise Manager Database Tuning with the Oracle Tuning Pack* manual is one of several Oracle Enterprise Manager documents.

Oracle Enterprise Manager base documentation

- The *Oracle Enterprise Manager Readme* provides important notes regarding the online documentation, updates to the software, and other late-breaking information.
- The *Oracle Enterprise Manager Installation CD-ROM Insert* provides information about installing Oracle Enterprise Manager.

- The *Oracle Enterprise Manager Administrator's Guide* explains how to use Oracle Enterprise Manager, Oracle's systems management console, common services, and integrated platform tool.
- The *Oracle Enterprise Manager Concepts Guide* provides an overview of the Oracle Enterprise Manager.
- The *Oracle Enterprise Manager Configuration Guide* explains how to configure Oracle Enterprise Manager.
- The *Oracle Enterprise Manager Application Developer's Guide* describes the programming external interfaces to the Oracle Enterprise Manager console.
- The *Oracle Enterprise Manager Messages Manual* describes the Oracle Enterprise Manager error messages and methods for diagnosing the messages.

Oracle Enterprise Manager Change Management Pack documentation

- The *Oracle Enterprise Manager Change Management Pack Readme* provides important notes regarding the Change Management Pack online documentation, updates to the software, and other late-breaking information.
- The *Oracle Enterprise Manager Getting Started with Oracle Change Management Pack* manual provides an overview of the concepts and features of the Oracle Change Management Pack applications.

Oracle Enterprise Manager Diagnostics Pack documentation

- The *Oracle Enterprise Manager Diagnostics Pack Readme* provides important notes regarding the Diagnostics Pack online documentation, updates to the software, and other late-breaking information.
- The *Oracle Enterprise Manager Performance Monitoring and Planning Guide* provides an overview of the concepts and features of the Oracle Performance Manager, Oracle Capacity Planner, and Oracle TopSessions applications. It also describes the Oracle Advanced Events for managing database, listener, and service types.
- The *Oracle Enterprise Manager Oracle Trace User's Guide* explains how to use the Oracle Trace application to capture and use historical data to monitor Oracle databases.
- The *Oracle Enterprise Manager Oracle Trace Developer's Guide* explains how to instrument your application with Oracle Trace routines.

Oracle Enterprise Manager Tuning Pack documentation

- The *Oracle Enterprise Manager Tuning Pack Readme* provides important notes regarding the Tuning Pack online documentation, updates to the software, and other late-breaking information.
- The *Oracle Enterprise Manager Database Tuning with the Oracle Tuning Pack* provides an overview of the concepts and features of each of the applications included in the Oracle Tuning Pack. The applications include: Oracle Auto-Analyze, Oracle SQL Analyze, Oracle Expert, Oracle Index Tuning Wizard, and the Oracle Tablespace Manager. A description of how these applications can work together to tune an Oracle database is also provided.

Other References

The following books are further resources for learning more about Oracle database tuning. (Note that the books listed here are not specifically recommended by Oracle, nor does Oracle take responsibility for any information provided by these sources.)

Advanced Oracle Tuning and Administration, by Eyal Aronoff, Kevin Loney, and Noorali Sonawalla. Oracle Press series, Osborne McGraw-Hill, 1997.

High Performance Oracle8 SQL Programming and Tuning, by Pete Cassidy. The Coriolis Group, 1998.

High-Performance Oracle Database Applications : Performance and Tuning Techniques for Getting the Most from Your Oracle Database, by Donald K. Burleson. The Coriolis Group, 1996.

High Performance Oracle 8 Tuning, by Donald K. Burleson. The Coriolis Group, 1997.

Oracle and Unix Performance Tuning, by Ahmed Alomari. Prentice Hall Computer Books, 1997.

Oracle: The Complete Reference, third edition, by George Koch and Keven Loney. Oracle Press series, Osborne McGraw-Hill, 1997. *(Oracle 7.3)*

Oracle: The Complete Reference, Electronic Edition, by George Koch and Keven Loney. Oracle Press series, Osborne McGraw-Hill, 1997. *(Oracle 7.3)*

Oracle8 Tuning, by Michael Abbey, Michael J. Corey, Daniel J. Dechichio, and Ian Abramson. Oracle Press series, Osborne McGraw-Hill, 1997.

Oracle Performance Tuning (Nutshell Handbook), second edition, by Mark Gurry and Peter Corrigan. O'Reilly & Associates, 1996.

Oracle Performance Tuning and Optimization, by Edward Whalen. Sams Publishing, 1996.

Oracle SQL High-Performance Tuning, by Guy Harrison. Prentice Hall Computer Books, 1997.

Part I

Oracle Tuning Pack Introduction

This section gives a brief description of the various applications that comprise the Oracle Enterprise Manager Tuning Pack, and discusses some of the ways you can use these integrated applications to tune the performance of your databases.

This part contains the following chapter:

1

Introduction to the Oracle Tuning Pack

The Oracle Tuning Pack enables users to proactively tune their database environment. The applications within the Oracle Tuning Pack can be used to:

- Identify and resolve performance problems
- Quickly identify the cause of a problem that has been reported and get advice on the best way to correct it
- Maintain existing performance
- Avoid performance problems by practicing proper maintenance operations
- Proactively identify tuning problems before they occur
- Provide tools and methodologies to establish and maintain database performance

Oracle Tuning Pack Applications

The Oracle Tuning Pack contains five applications to help users tune all aspects of their database environment; from identifying and correcting problem SQL statements to adjusting the instance parameters for the database. Tuning pack applications include:

- Oracle Auto-Analyze Wizard

The Auto-Analyze wizard provides automated updating of database statistics for Oracle8*i*. Auto-Analyze runs on a schedule within a user specified maintenance window and intelligently updates the dictionary statistics for any tables which would benefit.

- Oracle SQL Analyze

Oracle SQL Analyze can be used to identify and correct problem SQL statements. SQL Analyze allows the user to sort SQL statements by various performance metrics, and acquire detailed tuning information for any SQL statements. Wizards are included which guide the user through a SQL optimization methodology or add hints to existing SQL statements.

- Oracle Expert

Oracle Expert automates the database tuning process. Oracle Expert provides a methodology that helps the user to collect, evaluate, verify, and implement database tuning changes. Tuning areas covered by Oracle Expert include instance parameter tuning, database structure and placement, index tuning, and SQL statement reuse evaluation.

- Oracle Index Tuning Wizard

The Index Tuning Wizard provides an easy solution to the Index Tuning problem. It automatically identifies tables which would benefit from index changes, presents its findings to the user for verification, and allows the user to implement the index tuning recommendations.

- Oracle Tablespace Manager

Oracle Tablespace Manager identifies and corrects structural problems within the database, such as fragmentation problems and indexes that need to be rebuilt. A space reorganization wizard is provided to help the user correct any problems which are detected. Tablespace Manager also contains a graphical display to help the user to visualize the segment placement within their tablespaces.

Choosing the right application

The Oracle Tuning Pack supports both reactive and proactive tuning. Reactive tuning occurs when a database performance problem is reported that needs immediate correction. Proactive tuning adjusts the database environment before performance degradation becomes significant enough to be noticed by a user.

The first step in using the Oracle Tuning Pack involves determining which resource to tune. If you suspect that inefficient SQL is the cause of the problem, SQL Analyze can be used to identify the most resource intensive SQL statements, and help you to write them more efficiently. If you suspect a problem with your database resources, such as inefficient use of memory of data storage, Oracle Expert or Tablespace Manager can help optimize resource usage.

The Index Tuning and Auto-Analyze wizards are task focused, and solve specific tuning problems. They can be used for both proactive and reactive tuning. The

Index Tuning wizard will identify tables with inefficient indexes and make recommendations that will improve access to those tables. The Auto-Analyze wizard optimizes updating the data dictionary statistics that are used by the optimizer to determine the best data access method.

A new feature in the Oracle Tuning Pack 2.0 release is the "SQL History". The purpose of the SQL History is to gather and provide the relevant SQL statements that execute within the database environment to any Oracle Tuning Pack application that makes index tuning recommendations.

Oracle Expert, SQL Analyze, and the Index Tuning wizard share SQL statements collected in the SQL History. This ensures consistency in index tuning recommendations between the different Oracle Tuning Pack applications. The SQL History also saves production database resources, since SQL statements do not need to be collected separately for each application.

Additional details on how to use each Oracle Tuning Pack application can be found in the application-specific sections of this manual.

Part II

Getting Started with Oracle Auto-Analyze

This section provides an overview of the Oracle Auto-Analyze wizard. This wizard helps you create jobs that run on a schedule within a user specified maintenance window and intelligently updates the dictionary statistics for any tables which would benefit.

This part contains the following chapter:

2

Getting Started with Oracle Auto-Analyze

One of the most important factors in successful database tuning and performance is using the most current dictionary statistics available. Oracle databases use the ANALYZE command and STAT DBMS_STATS (for Oracle8*i* databases) to update statistics on the storage characteristics of tables, indexes, and clusters, which are then stored in the data dictionary. These statistics are used by the cost-based optimizer to determine the most efficient execution plan, and are used by Oracle SQL Analyze, Oracle Expert, and other Oracle Enterprise Manager applications to identify current and potential performance problems.

To keep your statistics current, you need to perform regular analyses of your databases. Updating the data dictionary statistics, however, may temporarily lock rows in the database, use up large amounts of temporary storage space, and can take more time than is available during regularly scheduled maintenance. It also takes a certain level of expertise and organization: you need to know which tables to analyze first, which tables should not be analyzed at all (such as SYS and SYSTEM), and you must keep track of tables that have been added or deleted to the database. There are times when it is simply impractical to collect statistics for a production database.

Oracle Auto-Analyze helps you work around this problem by letting you schedule data collection jobs within regularly scheduled maintenance periods (maintenance windows). This lets the Auto-Analyze job update the data dictionary statistics when the most resources are available and the least number of users will be affected. It also prioritizes the database objects, so if the time needed to analyze all the schema exceeds the limits of the maintenance window, the most "important" objects are analyzed first.

How Oracle Auto-Analyze Works

Oracle Auto-Analyze uses the parameters you set in a four-panel wizard (see "Using Oracle Auto-Analyze" on page 2-2) to schedule an Oracle Enterprise Manager Job that analyzes the selected schema within your databases.

To schedule the job, Oracle Auto-Analyze takes the list of schemas to analyze and prioritizes them based on the size and rate of change since the last analysis. It calculates the amount of time allotted to it during the maintenance window, and analyzes as many schema as it can during that time period, according to priority. Even if there is not enough time to analyze all of the objects listed, Oracle Auto-Analyze will first try to analyze the objects it determines to be the most important. Oracle Auto-Analyze also remembers where it left off after the last job, and takes that into consideration when prioritizing the tables the next time the job is run.

Using Oracle Auto-Analyze

Oracle recommends that you analyze your database on a regular basis and after any major changes to the data in the database if you are using the cost-based optimizer.

To take advantage of Auto-Analyze's features, the node on which the database is located must be running an Oracle Agent.

To run Auto-Analyze, you must configure the target databases on the server side. This configuration creates a repository on the managed node, where data is saved and the job history is maintained.

Two methods for setting up Auto-Analyze are documented:

- Setting Up Auto-Analyze Using the Job System
- Setting Up Auto-Analyze Manually

Note: Oracle Auto-Analyze is only available with Oracle8*i* and later databases.

Setting Up Auto-Analyze Using the Job System

To set up Auto-Analyze using Oracle Enterprise Manager's Job System, follow the instructions below.

1. Copy VbuaTask.jar to a `$ORACLE_HOME\network\agent` on each 8.1.5 target to be configured.

   $ORACLE_HOME is the agent's Oracle home.

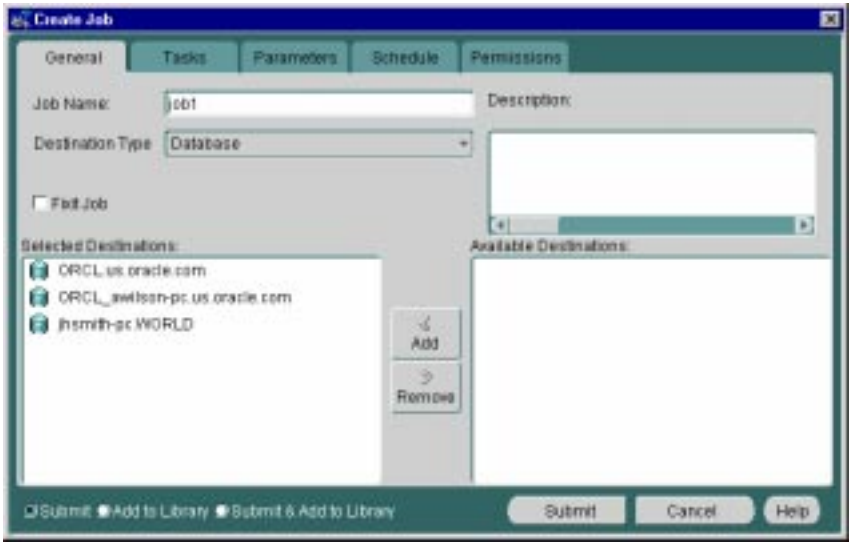
2. Discover all of the 8.1.5 targets to be configured using Oracle Enterprise Manager.

3. Create Job1.

   a. Select Create Job from the Console's Job menu.

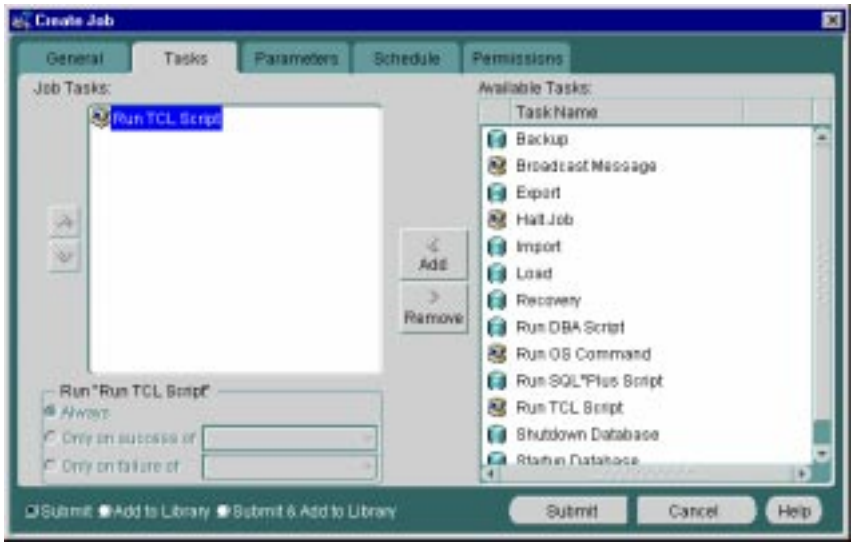
   b. In the General page, choose the 8.1.5 targets to be configured as destinations of the job.

Figure 2–1 General Page



c. In the Tasks page, select "Run TCL Script" as your Job Task from the list of Available Tasks.

Figure 2–2 Task Page

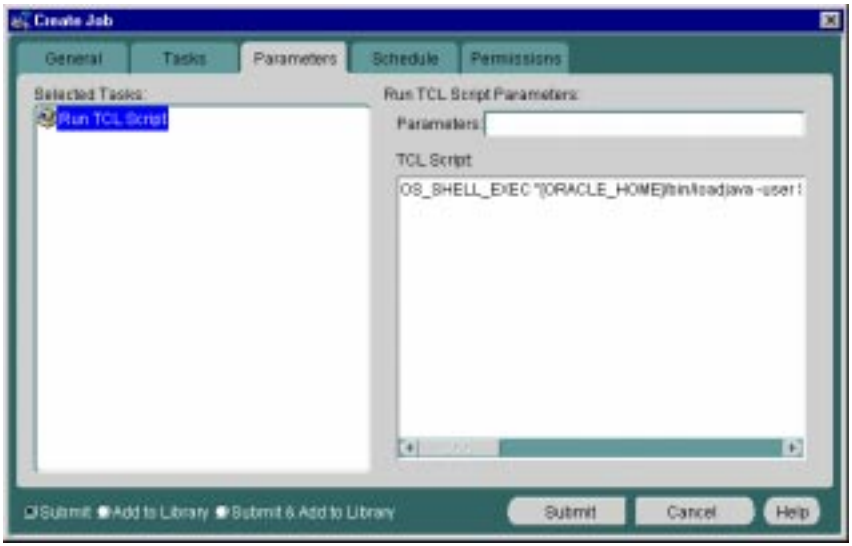


d. In the Parameters page, provide the following command as the TCL Script:

```
OS_SHELL_EXEC "[ORACLE_HOME]/bin/loadjava -user $SMP_USER/$SMP_PASSWORD
-resolve -force [AGENT_ORACLE_HOME]/network/agent/VbuaTask.jar"
```

Note: The command you enter as the TCL Script must be a single line. Also, the forward slashes in the TCL Script command must remain forward slashes regardless of the operating system.

Figure 2–3 Parameters Page



4. Create Job2.

   a. Select Create Job from the Console's Job menu.

   b. In the General page, choose the 8.1.5 targets to be configured as destinations of the job.

   c. In the Tasks page, select "Run SQL*Plus Script" as your Job Task from the list of Available Tasks.

   d. In the Parameters page, paste the contents of the `vbuacre.sql` into the Script Text text box. The `vbuacre.sql` file is installed with the Tuning Pack into `<ORACLE_HOME>\sysman\admin`.

Note: This description is relative to the system Oracle Enterprise Manager is installed on. The above syntax is appropriate for UNIX. For NT, backslashes replace forward slashes. This note about backslashes does not apply to the TCL Script command.

---

5. Submit the first job and after it succeeds submit the second job.

Setting Up Auto-Analyze Manually

To set up Auto-Analyze manually, follow the instructions below.

> **Note:** It is recommended that you set up Auto-Analyze using the Oracle Enterprise Manager's Job System, which is a more efficient method if you are setting up on multiple targets.

1. Move the `VbuaTask.jar` file from the Enterprise Manager Console client node to the target database's `$ORACLE_HOME\network\agent` directory:

   `VbuaTask.jar` is located in the client node's `$ORACLE_HOME\jar` directory.

   `vbuacre.sql` and `vbuadrop.sql` are located in the client node's `$ORACLE_HOME\sysman\admin` directory.

2. Create and run a loadjava TCL job using the Enterprise Manager Job System or run the job manually on a target database.

   - For information on creating and running a TCL job, refer to the Jobs chapter of the *Oracle Enterprise Manager Administrator's Guide*.

     You can save the job to the Job Library so that you can rerun it on other target databases.

     The username/password must match the preferred credentials.

   - To run the job manually on a target database, at an operating system prompt, enter:

     ```
     loadjava -user $SMP_USER/$SMP_PASSWORD -resolve -force [AGENT_ORACLE_
     HOME]/network/agent/VbuaTask.jar
     ```

     The command string is case-sensitive.

     The username/password must match the preferred credentials.

3. Run the `vbuacre.sql` script to create the repository on the target database.

   You can run the script in SQL*Plus (connected as a user with preferred credentials) or as a SQL*Plus job using the Enterprise Manager Job System.

Starting Oracle Auto-Analyze

To start Oracle Auto-Analyze, select **Tools =>Oracle Tuning Pack=>Oracle Auto-Analyze** from the Oracle Enterprise Manager console menu, or select **Oracle Auto-Analyze** from the Tuning Pack drawer. A dialog appears, introducing the wizard and guiding you on to the next step.

Follow the instructions provided in the Oracle Auto-Analyze wizard panels. They will guide you through the following steps:

- Welcome Page, introducing you to Oracle Auto-Analyze
- Maintenance Window Schedule, in which you define the time periods when the analysis is to be performed
- Parameters Page, where you set the maximum amount of time to be taken by the analysis, and specify the schema to be analyzed.
- Job Information Page, where you specify the name and description of the job to be submitted to Oracle Enterprise Manager, and schedule the submission.

When you are finished, the job is saved and submitted to Oracle Enterprise Manager at the specified time. When it is submitted, the job is listed in the Event pane of the Oracle Enterprise Manager console.

Welcome Page

The welcome screen introduces Oracle Auto-Analyze. Select the **Next** button to continue.

Maintenance Window Schedule

An *Oracle Auto-Analyze maintenance window* is the period of time during which the Auto-Analyze job will run.The maintenance window allows you to select the best time to schedule an Auto-Analyze job, based on when resources are available and when gathering statistics from your database will least interfere with normal database use. The Maintenance Window Schedule panel lets you specify whether a maintenance window will begin immediately or will occur on a scheduled basis.

To set the maintenance window for Auto-Analyze jobs, make the following selections.

Run Job

Select the radio button that indicates how often you want Oracle Auto-Analyze to run. You can set the jobs to run:

- Immediately
- Once (one time only)
- At regular intervals, at an hourly or daily rate
- On a specific day of a week
- On a specific date or set of dates each month

Notice that when you make your selection, the appearance of the lower right pane changes. (See Figure 2–4.)

Setting the Start Execution and Stop Execution times determines the begin and end dates between which the maintenance windows occur.

> **Note:** You will determine the duration of the maintenance window in the next page.

Start Execution

Set the date and time at which the job intervals begin.

End Execution

Set the date and time at which Oracle Auto-Analyze will stop scheduling Auto-Analyze jobs.

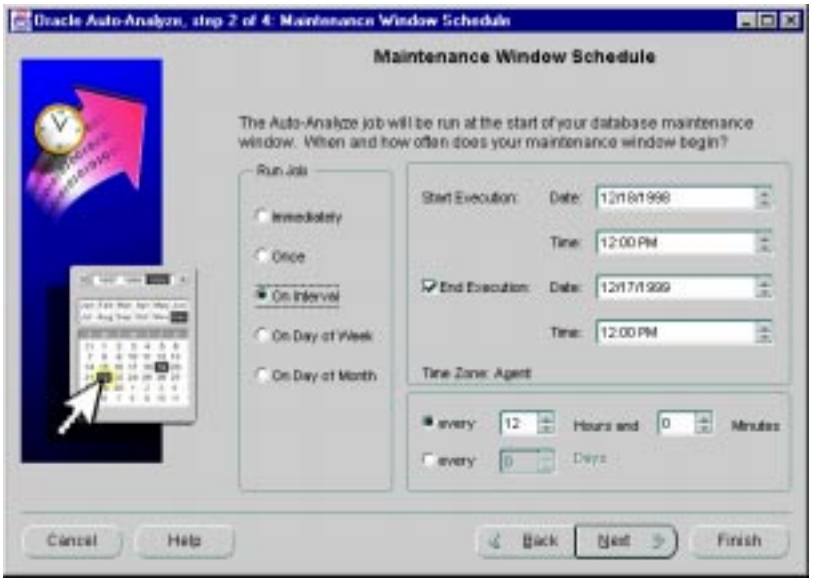
Use Which Time Zone

Because Oracle Enterprise Manager allows you to manage databases from any location, the client and the Agent may be located in completely different time zones. This setting lets you choose which point of reference Oracle Auto-Analyze will use to determine the time to begin and end the interval period.

You can choose from:

- **Agent:** The time where the Agent is located
- **Client:** The time where the client is located
- **GMT:** Greenwich Mean Time

Figure 2–4 Maintenance WIndow Schedule



Example

If you make the following selections:

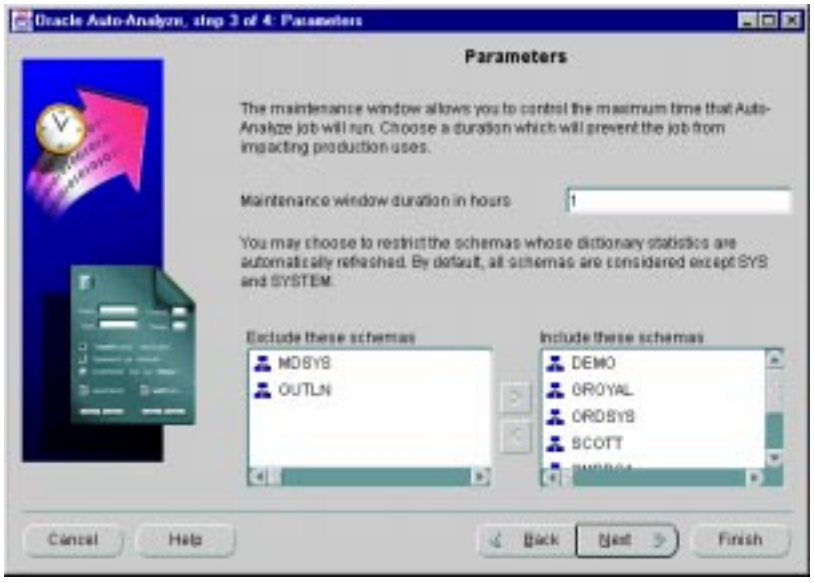
- On Day of Month
- Start Execution: 1/1/1999 and 1:00 AM
- End Execution: 12/31/1999 and 1:00 AM
- GMT
- check the boxes next to 15 and 30

Oracle Auto-Analyze will schedule Auto-Analyze jobs to occur regularly on the 15th and 30th of each month at 1:00 AM, Greenwich Mean Time, until 12/31/99. Each of the Auto-Analyze jobs will run as long as necessary, with the exception that they can not exceed the duration of the maintenance window specified in the next page.

Parameters Page

The Parameters Page lets you set the maximum time that the Oracle Auto-Analyze job may run and determine the schema to be analyzed.

Figure 2–5 Parameters Page



Maintenance Window Duration In Hours

Determines the longest period of time the Auto-Analyze job may run. If the job does not complete within the maintenance window duration period, Oracle Auto-Analyze will re-assess the state of the database statistics and will re-prioritize the order of the tables to be analyzed. Tables that were not completed in the previous job will be assigned higher priority because their statistics are older.

Exclude or Include These Schemas

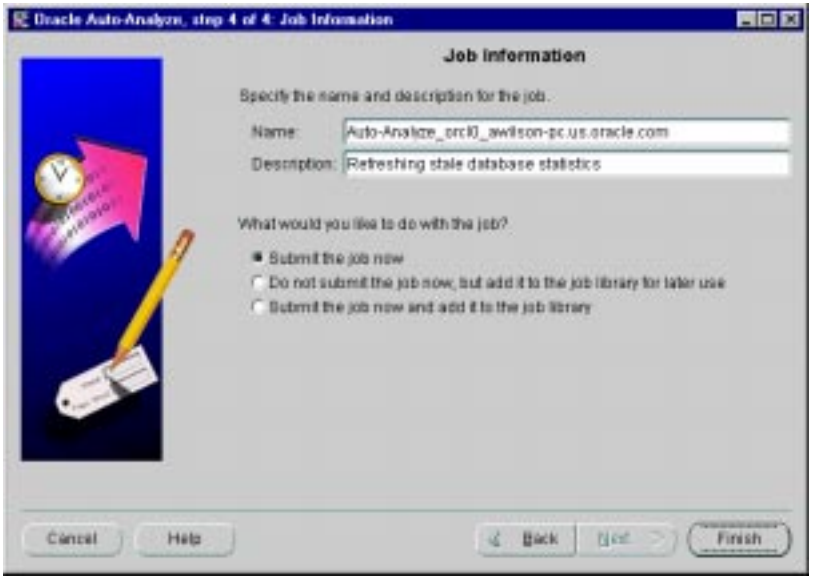
The default value is to include all available schema for analysis, except for SYS and SYSTEM. You can select tables to exclude from the analysis by selecting them from the right pane and clicking the arrow key.

Note: SYS and SYSTEM are automatically excluded from the schema. Schema that are to be optimized by the Rule-Based Optimizer should also be excluded.

Job Information Page

The Job Information Page lets you specify the name and description of the Oracle Auto-Analyze job about to be submitted to Oracle Enterprise Manager.

Figure 2–6 Job Information Page



Name

Specify a unique name for the job.

Description

Type in an identifying description of the job.

What would you like to do with the job?

You can make one of three choices:

- **Submit the job now:** The job will be submitted to Oracle Enterprise Manager and displayed in the console Job pane.
- **Do not submit the job now, but add it to the job library for later use:** Adds the job to the Oracle Enterprise Manager Job Library. You can access them later by opening the Job Library dialog from the Job menu in the console.

- **Submit the job now and add it to the job library:** Submit the job and add it to the library at the same time.

Click **Finish** to complete the Oracle Auto-Analyze session.

Reading the Job Report

If you submitted the job, it will be listed in the Oracle Enterprise Manager Job pane. You can use Oracle Enterprise Manager to monitor the status of the job and the databases the job is acting upon.

From the Job History, you can print a report that provides the following information about the Auto-Analyze job:

- Date the report was generated
- Total number of tables selected for analysis
- Tables analyzed during this particular execution
- The length of the maintenance window
- The schemas that were excluded from the analysis

Furthermore, the report provides the following statistics about each analyzed schema:

Schema Name
The name of the schema selected for analysis.

Table Name
The name of the table within the schema that was selected for analysis.

Priority
The priority of the table, a calculation based on the size and rate of change of the table since the last Auto-Analyze job.

Start Analyze Time
The time the analysis of the table began.

End Analyze Time
The time the analysis of the table was completed.

Row Count
The number of rows in the table.

Observed Rate of Change
The percent of change in the table since the last time it was analyzed.

Last Count Time
The last time the rows were counted by an Auto-Analyze job.

Discovery Status
Whether the table is within a discovered database.

Analyze Status
Whether or not the table was analyzed during this job.

Part III

Getting Started with Oracle SQL Analyze

Oracle SQL Analyze allows you to sort SQL statements by various performance metrics, and acquire detailed tuning information for any statements you choose. The SQL Hint and Tuning wizard can add hints to existing SQL statements and guide you through a SQL optimization methodology.

This part contains the following chapters:

3

Getting Started with Oracle SQL Analyze

This chapter contains the following topics:

Introduction

One of the primary benefits of the **SQL** language is its flexibility; you can achieve the same results by taking any of a number of different approaches. Although each approach might return the same results, performance varies dramatically depending on the database environment, structure of indexes and the access paths chosen by the **Oracle optimizer**.

While efficient SQL statements can help maintain peak database performance, inefficient statements can drag performance down. In many cases, tuning your SQL statements can increase overall performance by 100% or more.

Tuning SQL, however, has not been easy in the past. It involves collecting and analyzing information, and requires expert knowledge and experience. Tuning a SQL statement requires:

- awareness of the current environment and data
- knowledge of all schema objects
- an understanding of the Oracle Optimizer
- an in-depth knowledge of SQL

Oracle SQL Analyze provides you with the tools to collect information about the database environment and schema objects, analyze SQL performance, identify and compare different optimizer approaches, and edit your statements for optimal performance—in some cases, automatically.

Benefits of Oracle SQL Analyze

- Provides **TopSQL** functionality to identify the SQL statements that consume the most resources.
- Provides access to SQL statements that have been run against the database in the past, its **SQL history**.
- Executes SQL under various **optimizer** modes and presents **explain plans** and execution statistics for easy comparison.
- Walks you through the explain plan, providing the order of execution and an explanation of the operations.
- Steps you through an analysis of a SQL statement's potential join orders and methods, and provides alternative SQL for improved performance.

- Automatically checks SQL statements for basic SQL design "rules-of-thumb" violations and generates alternative SQL that corrects them.
- Presents relevant **object properties** to help identify and correct problems that will impact SQL performance.
- Provides easy access to initialization parameter settings that have a direct impact on SQL performance.
- Helps you add **hints** to your statement with the Hints Wizard.
- Saves SQL statements, execution plans, and performance statistics in a repository for future use.

What's New in Oracle SQL Analyze 2.0

Oracle SQL Analyze 2.0 introduces a number of new features that enhance the information you can use to tune your databases, and help automate the tuning process:

- You can now log into the Oracle Management Server to access shared information, or start Oracle SQL Analyze as a a standalone application.
- SQL History lets you examine past SQL statements and share information with Oracle Expert.
- New explain plan rules enhance your understanding of SQL statement processing and point out some areas for improvement.
- Support for remote queries illustrate queries that extend over distributed databases.
- Automated index tuning recommendations creates recommendations and implementation scripts that you can apply to your database.

SQL Tuning as Part of an Overall Tuning Process

SQL tuning is, of course, only one part of a comprehensive tuning process. As described in *Oracle Server Tuning*, there are several other areas you will want to consider tuning. The tuning method prescribed in that guide suggests tuning in the following sequence:

1. Tune the business rules
2. Tune the data design
3. Tune the application design

4. Tune the logical structure of the database
5. **Tune the SQL**
6. Tune the access paths
7. Tune the memory
8. Tune the I/O and physical structure
9. Tune the resource contention
10. Tune the underlying platforms

Decisions you make in one step may influence subsequent steps. For example, in Step 5 you may rewrite some of your SQL statements. These SQL statements may have significant bearing on parsing and caching issues addressed in step 7. Also, disk I/O, which is tuned in step 8, depends on the size of the buffer cache, which is tuned in step 7. At any time in the process, you may need to loop back from any step to any previous step. This tuning process is described in full in the *Oracle Server Tuning* guide.

In this guide, we will be concerned primarily with tuning SQL statements. But as you will see, the logical structure of the database, the access paths, the memory and the I/O, and physical structure all have a bearing on the effectiveness of your SQL statements.

Oracle SQL Analyze helps you with these issues by providing information about the database structure and letting you modify some initialization parameters to test the SQL statement against different conditions and database environments.

SQL Tuning Issues

Poorly performing SQL arises in applications for a number of reasons:

- **SQL is easier to write than to analyze**. Although SQL is a relatively easy language to learn, its non-procedural nature tends to obscure performance-related issues. As a result, it's much harder to write efficient SQL than it is to write functionally correct SQL.
- **Collecting data and analyzing performance can be difficult and time-consuming.** Before the advent of Oracle SQL Analyze, you would need to run several different SQL scripts to obtain statistics describing the various database objects such as indexes and views, the explain plan, and the execution performance statistics for a specific SQL statement. And picking the poor performer out of all the statements run against your database could be a

particularly daunting task. Oracle SQL Analyze collects the relevant data for you, and helps you identify statements by the resources they consume.

- **Programmers assume the optimizer will make the best decisions.** The Oracle optimizer is the part of the Oracle server which tries to determine the most efficient way to execute your SQL statements. Although it can choose between hundreds of possible approaches much faster than any individual SQL programmer, the programmer might have essential information about the application's nature or environment that the optimizer lacks. The optimizer is an excellent aid, but it cannot make as good a decision as an experienced SQL programmer.

  SQL Analyze lets you adjust environmental information and to compare different optimizer modes and execution plans, helping you determine the most efficient way to execute your SQL statements.

Methods of SQL Analysis and Tuning

As a SQL tuner, you need to be able to gather and analyze environmental data and performance statistics in order to locate problem areas. The following sections describe the information you can gather and the methods Oracle SQL Analyze makes available for tuning your statements.

Analyzing Explain Plans

The *explain plan* allows you to evaluate the steps in an execution path for a statement without actually executing the statement. The explain plan shows you the following:

- the relative **cost** of executing the statement (if you use the cost-based optimizer)
- the execution path chosen by the optimizer
- which indexes are being used
- which join methods are being used
- the order in which joins are performed

You can use Oracle SQL Analyze to generate and walk through explain plans for each of the available optimizer modes (see the next section). Oracle SQL Analyze creates a graphical view of the explain plan and the **compact view**, which illustrates in more detail how joins are performed.

Generating and "walking through" explain plans is discussed "Walking Through Explain Plans" on page 3-31**.**

Controlling Optimizer Modes

The task of the Oracle optimizer is to find the most efficient method for executing a SQL statement. The optimizer has four primary modes of operation: **Rule**, **Cost First**, **Cost All**, and **Choose**. The mode you select directs the strategy the optimizer:

- *Rule* evaluates possible execution paths and rates the alternative execution paths based on a series of syntactical rules.
- *Cost First* executes the statement in a way that most efficiently retrieves the first row of data.
- *Cost All* executes the statement in a way that least impacts overall performance.
- *Choose* invokes the Cost All mode if any tables have been analyzed, and the Rule-Based mode if no tables have been analyzed.

These modes are explained further in Understanding Hints on page 4-2. You can set a default optimizer goal by specifying the OPTIMIZER_MODE parameter in your database's init.ora file. Additionally, you can set the optimizer for a specific SQL statement by adding hints to it.

But how do you know which optimizer mode will be most efficient for your statement? Oracle SQL analyze lets you test each of these execution strategies against a SQL statement and provides cost information and performance statistics that help you to determine which mode is best.

Adding Hints

Within a query, you can specify hints that direct the Cost-based optimizer in its processing of the query.

Hints affect the following:

- Execution paths

  As described above, hints can be used to determine the optimizer mode.

- Data access methods

  Hints can force the optimizer to use specific scan methods while executing a statement. For example, it can direct the optimizer to use specific index scans instead of performing full table scans.

- Join Methodology

  Hints can affect the way the optimizer chooses to join rows.

- Parallel execution

  Hints can be used to enhance parallel operations, which can lead to potentially significant cost reduction.

  Oracle SQL Analyze provides a Hint Wizard that helps you add syntactically correct hints to your statement.

Applying Rules-of-Thumb

There are certain syntax variations that are known by Oracle experts to have a negative impact on performance. Oracle SQL Analyze can evaluate statements against a conservative set of rules, identifying less efficient coding and providing an alternative statement when possible. You can evaluate these rules automatically, using the SQL Analyze Tuning Wizard.

Applying Join Methodology

Most SQL queries involve selecting data from multiple tables. In these operations, data from several tables is "joined" from multiple tables in order to produce the desired results set. The type of join method used and the order of table joining is driven by several factors, such as the presence of indexes and the cardinality of columns involved in the selection process.

You can control join methodology through the use of SQL hints. This can be valuable for specific queries where the developer is aware of details that may not be available to the optimizer, such as:

- the performance goal of the query (throughput vs. response time)
- outdated or non-existing statistics for certain objects
- bind variables that may effect a filter condition.

Using hints to control the join method and order for specific queries can be complex and risky. To assist the developer with this, SQL Analyze provides an automated methodology that an be used to evaluate alternative join strategies where appropriate. Oracle SQL Analyze will estimate the object statistics and collect typical values for bind variables that can be used to fully evaluate the optimal join order. If an alternative join order can be used, it will rewrite the statement with the necessary hints or reordering of objects.

Analyzing Object Properties

The performance of the SQL statement is also affected by the space usage of the objects accessed. Factors such as the existence of chained rows in a table can increase the number of I/Os required to retrieve the data set. Oracle SQL Analyze lets you examine the details for tables, indexes, clusters, and views.

The SQL Tuning Process

This section suggests a methodology for using Oracle SQL Analyze to identify problem statements and tune them for greater efficiency. For a deeper understanding of the concepts involved, see Chapter 4, "Tuning SQL Statements".

Step One: Start a Tuning Session

There are several way to begin a tuning session, depending on the status of the SQL you want to tune.

- To identify problem statements that have been run or are about to be run on the database, you'll want to begin by analyzing the statements gathered by TopSQL or stored in the SQL History. After you have determined the statements that need tuning, you can go on to Step Two, "Gathering Information."
- To create a new statement, you can enter a new SQL statement or import or copy an existing SQL statement into Oracle SQL Analyze.
- You can open a SQL file and edit the statements within the file.
- You can go back to SQL statements from a previous tuning session.
- You go back and edit a previous tuning session.

Step Two: Gathering Information

After you have chosen a statement to tune, you need to understand more about the database environment in which the SQL statement is being executed, and more about the performance of the statement.

- To learn more about the tuning environment, take a look at the initialization parameters.
- To learn more about the SQL statement's performance, generate an explain plan. Walking through the explain plan will show you how the statement is executed. You may also want to compare the explain plan with another plan. Within the explain plan, you can gather information about:
  - Performance statistics
  - The properties of Objects such as tables, clusters, indexes, and views
  - The join order selected by the optimizer
  - Parallelism

- Distribution

Step Three: Tuning the Statement

After reviewing the statistics, it's time to tune the statements. You can:

- Edit the statement manually.
- Use the Hint Wizard to add hints that specify:
  - the optimization approach for a SQL statement
  - the goal of the cost-based approach for a SQL statement
  - the access path for a table accessed by the statement
  - the join order for a join statement
  - a join operation in a join statement
- Use the Tuning Wizard to help you tune for optimum performance. The Tuning Wizard helps you:
  - add hints to the statement
  - apply SQL syntax guidelines (rules-of-thumb)
  - analyze and optimize the join methodology
  - measure the effectiveness of your tuning efforts by comparing the tuned statement against the original
- Get tuning recommendations. Oracle SQL Analyze will provide recommendations and a SQL script that you can use to implement the suggested changes.

Step Four: Verify Your Results

You can use the same methods you used for gathering information to verify that the performance of your statement has been improved:

- execute the new statements and compare the results
- generate new explain plans and compare them
- review the object properties to ensure they are accurate.

4

Starting a Tuning Session

This chapter contains the following topics:

Starting a Tuning Session

To begin your tuning session, you must have the following:

- the SQLADMIN or DBA role assigned to you, as described in the next section.
- "Create Table" privileges

Oracle SQL Analyze can be started from Tuning Pack drawer on the Oracle Enterprise Manager console, from the menu, or from the System Prompt.

To begin Oracle SQL Analyze from the menu, select **Tools=>Tuning Pack=>Oracle SQL Analyze**.

To begin from the System prompt, type `vmq`.

Assigning the SQLADMIN Role

In order to run Oracle SQL Analyze, you must assign the SQLADMIN role to the user that will run Oracle SQL Analyze.

> **Note:** The permissions contained within this role are also in the DBA role. Therefore, DBAs do not need the SQLADMIN role assigned to them.

The VMQROLE.SQL script has been provided to help automate the process of creating the SQLAMDIN role. It is located in the $ORACLE_HOME\SYSMAN\ADMIN directory.

1. From the Oracle Enterprise Manager program group, double-click on the SQL Worksheet icon to start the tool.
2. Use the Login Information dialog box to connect to the database you want to run SQL Analyze on. Login as SYS.
3. From the Worksheet menu, run the VMQROLE.SQL script to create the SQLADMIN role for the managed database.
4. In the SQL Worksheet bottom pane, assign the SQLADMIN role to the user by typing:

   ```
   GRANT SQLADMIN to <user>
   ```

5. Exit SQL Worksheet.

Creating and Working with Tuning Sessions

Throughout the tuning process, you will be working in a main window divided into a number of separate panes. The types of panes and their locations within the main window depend upon the type of action you are performing. For example, when you are generating and analyzing explain plans, the window will be divided into the Navigator window, the SQL Text window, and the Details window.

The following sections describe how the Oracle SQL Analyze interface is used for different operations.

Oracle SQL Analyze Repository

Oracle SQL Analyze stores the information for tuning sessions in the Enterprise Manager repository. The following information is saved when you select **Save to Repository** from the **File** menu:

- information necessary to reconstruct the Navigator tree, including initialization parameter information, and SQL and explain plan object names
- SQL statements
- all explain plans and related statistics

The list of discovered nodes in the Navigator window reflects the information provided by Oracle Enterprise Manager.

Note: Oracle SQL Analyze may display additional nodes whose connections have been dropped, if there are tuning sessions connected to the disconnected node.

Object properties and their estimates are not saved in the repository.

Migrating From Release 1.5.5 to 1.6.0

Oracle SQL Analyze 1.5.5 gave you the option of saving your SQL tuning information in a *workspace*. A SQL Analyze workspace consists of Oracle tables that are used by SQL Analyze to store information about your SQL tuning. If you elected to create a SQL Analyze workspace, then these tables were created in the schema of the SQL Analyze database user connection. You can create a workspace for each SQL Analyze database connection, therefore you may have created multiple workspaces in SQL Analyze 1.5.5.

In Release 2.0, the SQL Analyze repository is created in the Oracle Enterprise Manager repository. See the *Oracle Enterprise Manager Configuration Guide* for information on creating a repository.

If you created an Oracle SQL Analyze outside of your Oracle Enterprise Manager 1.5.5 repository and you want to save this workspace information for use in release 2.0, you will need to move the tables from the SQL Analyze workspace schema into the Oracle Enterprise Manager 1.5.5 repository schema. Note that if you have multiple 1.5.5 workspaces you can only migrate one of your workspaces. This migration must be done before you begin using SQL Analyze 2.0.

To move the tables, use the Oracle Server Export/Import Utilities as described in Oracle Utilities and your platform-specific documentation. The tables that must be moved into your Oracle Enterprise Manager repository are:

VMQ_SQL_DATABASE

VMQ_SQL_DATABASE_PARAMS

VMQ_SQL_SESSION

VMQ_SQL_SESSION_PARAMS

VMQ_SQL_OBJECT

VMQ_SQL_TEXT

VMQ_SQL_IMPORT_STATS

VMQ_SQL_PLAN_RULE

VMQ_SQL_PLAN_COST_FIRST

VMQ_SQL_PLAN_COST_ALL

VMQ_SQL_STATS_RULE

VMQ_SQL_STATS_COST_FIRST

VMQ_SQL_STATS_COST_ALL

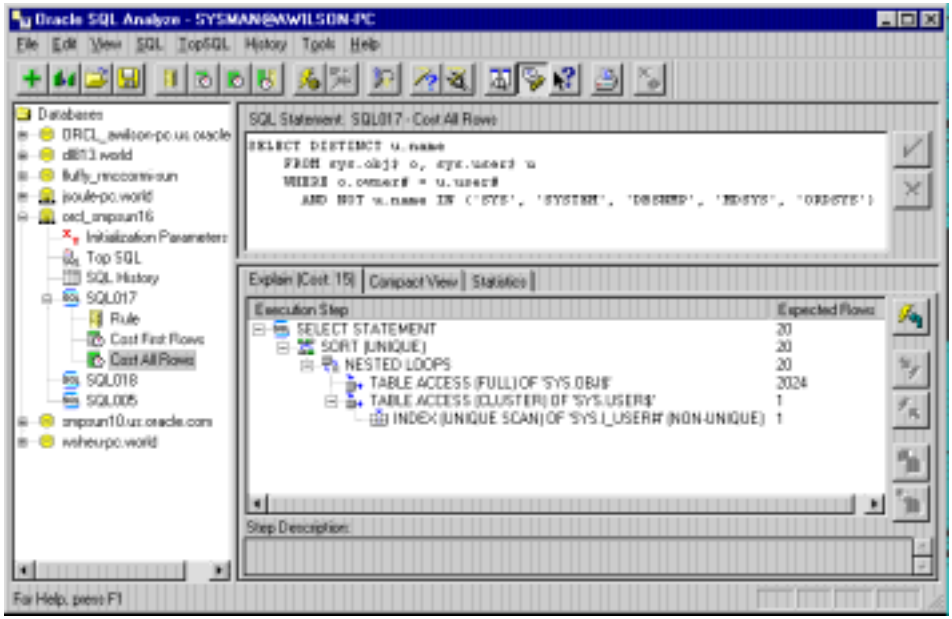
Oracle SQL Analyze Main Window

The Main window is the basic work area for SQL statement creation and tuning. It displays the databases, sessions, and SQL statements you are working with, as well as the explain plans for the SQL statements.

As shown in Figure 4–1, the Main window is usually divided into three panes:

- The Navigator window displays a tree view listing available nodes, objects, and SQL statements.
- The SQL Text window lists the selected SQL statement. It also displays the list of SQL statements called by TopSQL or the SQL History.
- The Details window displays information about the selected statement, including explain plans, object properties, and performance statistics.

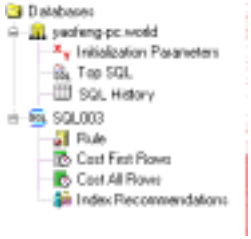
Figure 4–1 Main Window



The Navigator Window

The Navigator pane provides access to the database services you can use to tune your SQL statements against. The Navigator pane is always displayed, and is shown in Figure 4–2.

Figure 4–2 Navigator Window



Clicking on the "+" symbol reveals the Initialization Parameters, TopSQL, SQL History, SQL Statement, Explain Plan, and Tuning Recommendation objects related to that database.

The top level is the database node, just as it is represented in Oracle Enterprise Manager.

Note: Services that have been disconnected from Oracle Enterprise Manager will still be displayed in the Navigator tree if there are SQL statement objects associated with them. To delete a disconnected service, select **File=>Remove Database Service**.

Clicking on the TopSQL object activates the TopSQL filtering operation, which allows you to sort SQL statements stored in V$SQLAREA according to the resources they consume.

Selecting the SQL History object calls the SQL History, which allows you to sort SQL statements stored in the SQL History repository.

Clicking on the Initialization Parameters object displays instance parameters you can edit to simulate different database environments.

The SQL Statement Object contains a single specific syntactical version of a SQL statement. Clicking on this object displays the statement in the SQL Text window.

The Explain Plan object contains a single explain plan generated for the SQL statement. Oracle SQL Analyze lets you generate Rule-based explain plans for all SQL statements, and Cost-based explain plans for SQL statements that have been analyzed with the ANALYZE command.

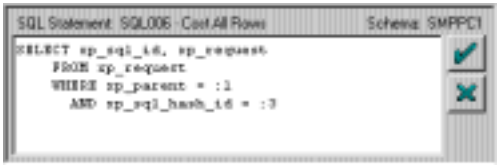
The Tuning Recommendations object is created if you generate index tuning recommendations for the database.

SQL Text Pane

The SQL Text pane, as shown in Figure 4–3, displays the SQL statement currently being analyzed.

In this window, you can edit the statement or select a database view to examine.

Figure 4–3 SQL Text Window



The statement name, type of explain plan that is currently displayed in the Details pane, and the owner of the current schema are listed at the top of the SQL Text pane.

Note: Identifying the appropriate schema owner is very important for accurate analysis of the statement. If a statement has been executed by more than one user, Oracle SQL Analyze may prompt you to select an owner for the schema when you try to get the explain plan or perform any other tuning operation on this statement.

The two buttons to the right of the text pane allow you to confirm or reject edits made to the SQL statement and revert back to the last saved version of the statement.

To confirm and save an edit, select the **Apply** button (marked by a check mark).

To reject an edit and revert back to the last saved version, select the **Revert** button ("X").

When you select either **Apply** or **Revert**, Oracle SQL Analyze will refresh all child objects of the SQL object, such as the explain plan or index tuning recommendations. If you select **Cancel**, the child objects will be marked as invalid until they are refreshed.

When you are using TopSQL or SQL History, the SQL Text window displays multiple SQL statements, sorted by order of resource consumption, as shown in Figure 4–4. You can drag these statements over to the Navigator window to create SQL Statement objects.

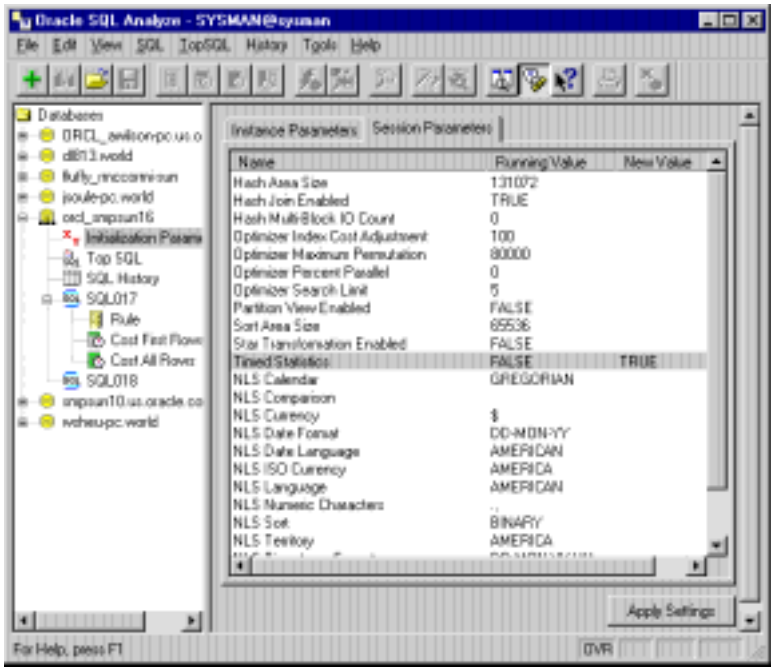
Figure 4–4 SQL Text Pane When Using TopSQL or SQL History

Details Pane

The Details pane displays information about the subject of your analysis. Its size appearance differs depending upon the operation being performed.

If, for example, you are creating explain plans based on the statement in the text window, the Details pane occupies the space to the left of the Navigator pane and below the SQL Text window as shown in Figure 4–1. If you are examining initialization parameters of a database, the Details pane occupies all the space to the right of the Navigator pane, as shown in Figure 4–5.

Figure 4–5 Details Window Showing Initialization Parameters



Selecting a Statement for Tuning

Although there are several ways to begin a SQL tuning session, the most likely scenario is that you want to identify existing SQL statements that are creating bottlenecks in your system. Statements that have the most potential to improve performance, if tuned, are those that:

- consume the greatest overall resources
- consume the greatest resource per rows (or per run)
- are executed most frequently

You can use the TopSQL function to sort the statements located in the V$SQLAREA view (those statements that have been run or are ready to be run against the database) according to their resource consumption. TopSQL is described in the next section, "Selecting Statements with TopSQL."

You can also examine statements stored in the SQL History repository. You can select statements from the SQL History the same way you select statements

fromTopSQL. For more information about the SQL History, see "Using SQL History" on page 5-31.

Other ways to select a statement for tuning include:

- entering a new SQL statement, as discussed on page 4-14.
- importing a statement from a SQL file, as discussed on page 4-15.
- opening a previously used tuning session, as discussed on page 4-15.

Selecting Statements with TopSQL

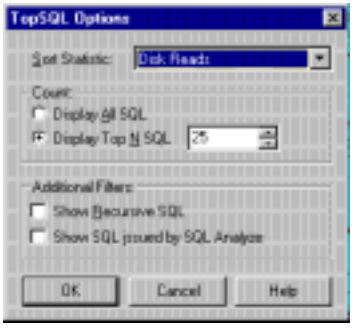
TopSQL lets you examine SQL statements that have been used on the database by the resources they consume. Using the statistics in this list, you can determine which statements consume the most resources and then select them for tuning.

TopSQL takes its statistics from the V$SQLAREA view. The V$SQLAREA view lists statistics on shared SQL areas, and provides statistics on SQL statements that are in memory, parsed, and ready for execution or which might have been executed already.

To start a TopSQL analysis:

1. Click on the TopSQL object in the Navigator window, as shown in Figure 4–2. A SQL options dialog opens, as shown in Figure 4–6.

Figure 4–6 TopSQL Options Dialog



2. Select the resource in the **Sort Statistic** field whose consumption you want to measure.
3. In the Count area, select the number of SQL statements you want displayed.
4. In the Additional Filters area, select the types of SQL you want displayed.

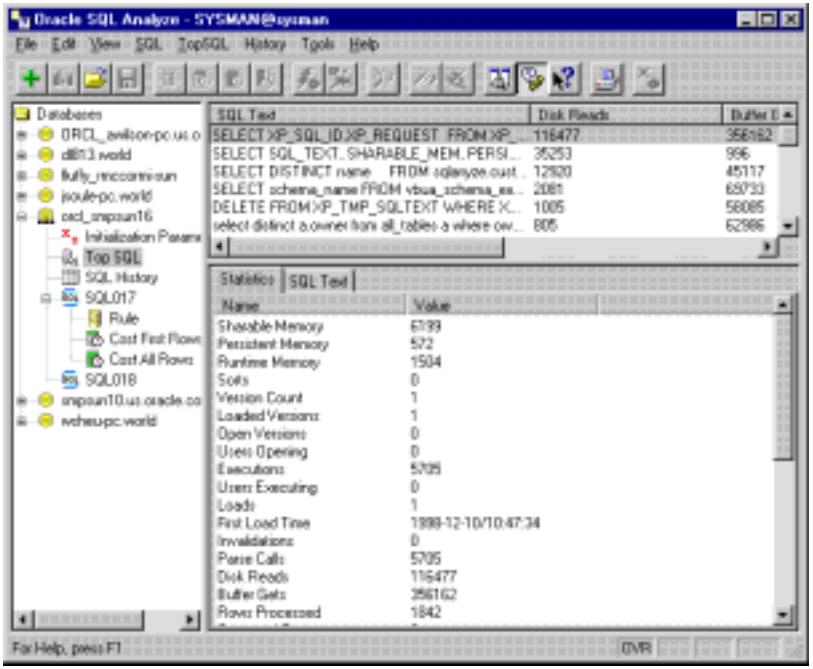
5. Select **OK**.

   The statements are displayed in the TopSQL View in order of their resource consumption.

6. Select a SQL statement you want to tune, drag it over to the Navigation window, and drop it on the database node.

7. A new SQL object is created. You can now tune this statement in Oracle SQL Analyze.

Figure 4–7 TopSQL View



TopSQL lets you sort the statements based on their use of the following database resources which are most likely to impact performance:

Buffer Cache Hit Ratio

The rate at which Oracle finds the data blocks it needs already in memory. The closer the hit ratio is to 100%, the better your system will perform.

Buffer Gets
The number of buffer gets for all cursors. A measurement of CPU usage, excessive buffer gets may indicate that this statement needs to be examined more closely.

Executions
The number of executions carried out for the statement.

CPU Usage Per Execution
This statistic shows the average number of buffer gets per execution. Excessive buffer gets may indicate that this statement needs to be examined more closely.

CPU Usage Per Row
The number of buffer gets per rows processed.

Disk Reads
The number of disk reads for all cursors.

Disk Reads per Execution
This statistic takes the number of times the statement has been executed and calculates the number of disk reads per execution. Excessive disk reads may indicate that this statement needs to be examined more closely.

Executions
The number of times the statement was executed since it was brought into the library cache.

Parse Calls
The number of parse calls for all cursors.

Parse Calls Per Execution
The number of times the SQL statement was parsed per execution. Ideally, a SQL statement should be parsed once and executed multiple times, but some front-end applications re-parse every time they execute. The ratio can range from 0 to 1. A ratio is close to 0 is good. A ratio close to or equal to 1 indicates unnecessary parse calls.

Rows Processed
The total number of rows the parsed SQL statement returns. Depending upon the purpose of the statement, you may find the rows processed higher or lower than expected, indicating a need to examine this statement further.

Sorts
The number of sorts that was done for all cursors. An excessive number of sorts may indicate inefficient use of indexes or syntax that needs to be optimized.

Note: The most important factors which affect performance are (in order): Disk Reads, Buffer Gets, Sorts, and Executions. These statistics are shown in the Main Window's upper pane for all SQL statements.

Along with those listed above, the following statistics are displayed in the lower pane:

- Sharable Memory
- Persistent Memory
- Runtime Memory
- Sorts
- Version Count
- Loaded Versions
- Open Versions
- Users Opening
- Executions
- Users Executing
- Loads
- First Load Time
- Invalidations
- Parse Calls
- Disk Reads
- Buffer Gets
- Rows Processed
- Command Type

- Optimizer Mode
- Parsing User ID
- Parsing Schema ID
- Kept Versions
- Address
- Hash Value
- Module
- Module hash
- Action
- Action Hash
- Serializable Aborts
- Buffer Cache Hit Ratio
- CPU Usage Per Execution
- CPU Usage Per Row
- Disk Reads Per Execution
- Parse Calls Per Execution

You can read its complete list of V$SQLAREA statistics to gain a full understanding of the performance of a statement, then decide which statements require tuning, and what performance problems need to be addressed. These statistics are discussed more completely in the *Oracle Server Reference* and *Oracle Server Tuning*.

Entering a New Statement

To enter a new SQL statement, select **SQL=>Create New SQL**. Then enter the new statement in the SQL Statement pane located in the upper right portion of the Main pane.

Importing Statements From a SQL File

To import or copy an existing SQL statement into Oracle SQL Analyze:

From a SQL script

Select **File => Open SQL** to open a SQL script. A dialog will open, letting you select the desired SQL script.

From TopSQL or SQL History

Drag the desired SQL statement from the SQL Text pane onto a session or SQL object.

Opening a Previously Used Tuning Session

To open a previously used tuning session, click on the desired SQL statement object or explain plan object.

Printing

Oracle SQL Analyze prints SQL statements, explain plans, and statistical data for the statements and plans.

To print a SQL statement and its performance statistics, select the SQL object in the Navigator window, then select **File=>Print**.

To print a SQL statement, its explain plan, and the performance statistics for that explain plan, select the explain plan object in the Navigator window, then select **File =>Print**.

To print the list of TopSQL statements displayed in the SQL text pane, select **TopSQL=>Print**.

To print the list of SQL History statements displayed in the SQL Text pane, select **History=>Print**.

Saving

To save a SQL statement into a file, select **File => Save SQL As**.

To save the current tuning session, select **File => Save to Repository**.

5

Gathering and Analyzing Information

This chapter contains the following topics:

Understanding Statistical Information

Oracle SQL Analyze provides a wealth of information that is vital to your tuning efforts.

Information about the database environment, such as "Optimizer Mode" or "Sort Area Size," will influence the decisions the Oracle Optimizer makes when it generates an explain plan for a statement and how efficiently the operations are performed when the statement is executed. Oracle SQL Analyze shows you the values of these parameters in two locations:

- The *Instance-based Initialization Parameters View* shows you parameters that cannot be changed within Oracle SQL Analyze.
- The *Session-based Initialization Parameters View* shows you parameters that can edited to simulate different database environments for the purposes of testing different tuning scenarios.

Instance-based Initialization parameters are further discussed on page 5-2.
Session-based Initialization parameters are discussed on page 5-7.

You can also examine many of the logical constructs within your database: views, tables, indexes, and clusters. These objects are created to help manage information and make data access more efficient, but only if used correctly. The Object Properties provided by Oracle SQL Analyze will help you determine whether the information they arrange is still in accordance with the way the database is used, and whether the optimizer is taking full advantage of these objects or bypassing them. The available object properties and their meaning are discussed on page 5-16.

Performance statistics, of course, are the ultimate measure of SQL statement effectiveness. Oracle SQL Analyze lets you execute statements using different explain plans and then compare their performance. Explain plans, performance statistics and their analysis are discussed on page 5-30.

Analyzing the Database Environment

Every time a database is started, a system global area (SGA) is allocated and Oracle background processes are started. The system global area is an area of memory used for database information shared by the database users. The combination of the background processes and memory buffers is called an *Oracle instance.*

Oracle SQL Analyze lets you examine instance- and session-based initialization parameters, as described in the following sections.

instance-based Parameters

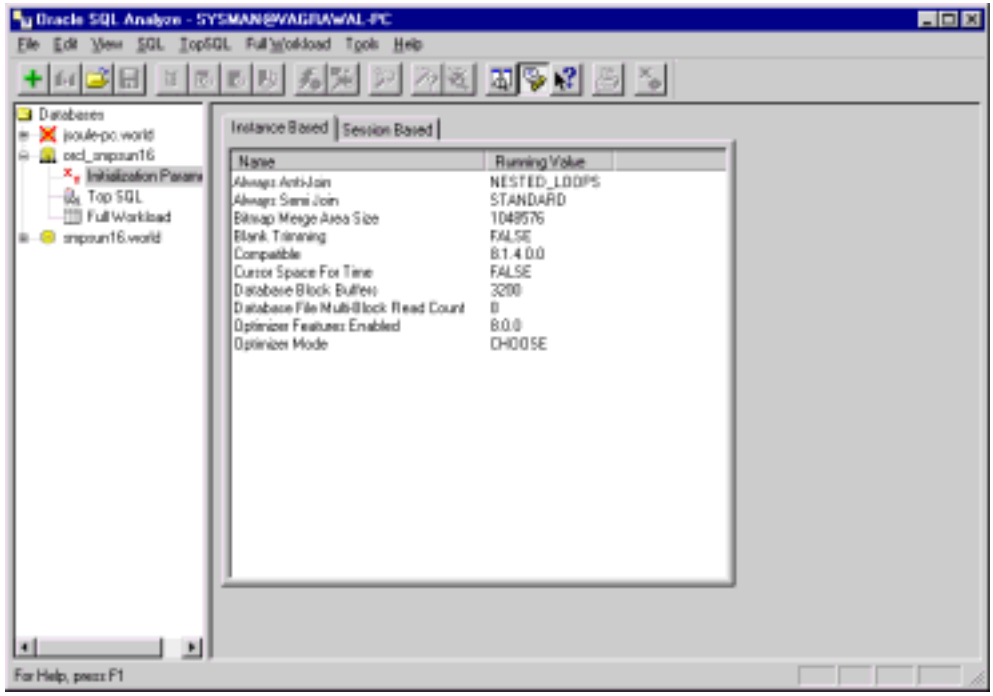
The instance-based initialization parameters displayed in the Database Parameters dialog affect memory and disk performance. You cannot edit these values from within Oracle SQL Analyze, but you should consider the effects these parameters have, and consider using Oracle Expert to tune them at a later date.

Note: The following information about initialization parameters and their tuning implications is meant as an introduction only. Please consult *Oracle Server Administration*, *Oracle Server Tuning*, and the *Oracle Server Reference* for more detailed information.

Opening the Database Parameters View

To open the Initialization Parameters view, click on the database node in the Navigation window, as shown.

Figure 5–1 Instance-based Parameters View



The Details Window on the right will display the following statistics:

Always Anti-Join

Description Sets the type of anti-join that the Oracle Server uses: NESTED_LOOPS, MERGE, or HASH. The system checks to verify that it is legal to perform an anti-join, and if it is, processes the subquery depending on the value of this parameter. The default setting is NESTED_LOOPS.

Tuning Considerations Always Anti-Join is useful in cost-based optimizations to make the most efficient use of parallel processing of the NOT IN clause.

HASH processes the NOT IN clause most efficiently. It causes the NOT IN operator to be evaluated in parallel using a parallel hash anti-join. Without this parameter set to HASH, NOT IN is evaluated as a (sequential) correlated subquery.

NESTED LOOPS processes the NOT IN clause least efficiently.

This parameter often needs to be set in data warehousing applications.

Bitmap Merge Area Size

Description Specifies the amount of memory used to merge bitmaps retrieved from a range scan of the index.

Tuning Considerations The default value is one megabyte. A larger value often improves performance because it will cause the optimizer to bitmap indexes more often.

Blank Trimming

Description Specifies the data assignment semantics of character datatypes.

Tuning Considerations A value of TRUE allows the data assignment of a source character string / variable to a destination character column / variable even though the source length is longer than the destination length. In this case, however, the additional length over the destination length is all blanks. A value of FALSE disallows the data assignment if the source length is longer than the destination length and reverts to SQL92 Entry Level semantics.

Compatible

Description Specifies the release with which the Oracle Server must maintain compatibility. The default value is the earliest release with which compatibility can be guaranteed.

Tuning Considerations You can use this parameter to immediately take advantage of the maintenance improvements of a new release in your production systems without testing the new functionality in your environment. This parameter also allows you to use a new release while at the same time guaranteeing backward compatibility with earlier releases, in case it becomes necessary to revert to the earlier release.

If you set this parameter to an earlier release, however, you may restrict or disable some of the features of the current release. To be sure that you are getting the full benefit of the latest performance features, make sure this parameter is set equal to the current release.

Cursor Space for Time

Description Specifies when a shared SQL area can be deallocated from the library cache to make room for a new SQL statement. The default value of this parameter is FALSE.

Tuning Considerations If the value is FALSE, a shared SQL area can be deallocated from the library cache regardless of whether application cursors associated with its SQL statement are open. In this case, Oracle must verify that a shared SQL area containing the SQL statement is in the library cache.

Setting the value of the parameter to TRUE saves Oracle a small amount of time and may slightly improve the performance of execution calls. If the value of this parameter is TRUE, a shared SQL area can be deallocated only when all application cursors associated with its statement are closed. In this case, Oracle need not verify that a shared SQL area is in the cache, because the shared SQL area can never be deallocated while an application cursor associated with it is open.

Do not set the value to TRUE if:

- you have found library cache misses on execution calls. Such library cache misses indicate that the shared pool is not large enough to hold the shared SQL areas of all concurrently open cursors.
- the amount of memory available to each user for private SQL areas is scarce. If the private SQL areas for all concurrently open cursors fills the user's available memory so that there is no space to allocate a private SQL area for a new SQL statement, the statement cannot be parsed and Oracle returns an error indicating that there is not enough memory.

For more information, see *Oracle8 Concepts*.

Database Block Buffers

Description This parameter is used to define the number of buffers in the buffer cache in the System Global Area (SGA). Each individual buffer pool is created from this total amount with the remainder allocated to the default buffer pool.

Tuning Considerations The number of buffers affects the performance of the cache. Larger cache sizes reduce the number of disk writes of modified data. However, a large cache may take up too much memory and induce memory paging or swapping.

The Database Block Buffers parameter, together with the Database Block Size parameter, determines the total size of the buffer cache. Effective use of the buffer cache can greatly reduce the I/O load on the database. Since Database Block Size can be specified only when the database is first created, use Database Block Buffers to control the size of the buffer cache.

For more information, see *Oracle Concepts*. Also, see your operating system-specific Oracle documentation for the default value.

Database Buffer Cache

Description Database buffers of the SGA store the most recently used blocks of database data; the set of database buffers in an instance is the database buffer cache. The buffer cache contains modified as well as unmodified blocks. Because the most recently (and often the most frequently) used data is kept in memory, less disk I/O is necessary and performance is improved.

Tuning Considerations The first time an Oracle user process accesses a piece of data, the process must copy the data from disk to the buffer cache before accessing it. This is called a *cache miss*. When a process accesses a piece of data that is already in the cache, the process can read the data directly from memory. This is called a *cache hit*. Accessing data through a cache hit is faster than data access through a cache miss.

The size of the cache affects the likelihood that a request for data will result in a cache hit. If the cache is large, it is more likely to contain the data that is requested. Increasing the size of a cache increases the percentage of data requests that result in cache hits. However, too large a cache can induce excessive swapping and paging.

Database File Multi-Block Read Count

Description This parameter is used for multi-block I/O and specifies the maximum number of blocks read in one I/O operation during a sequential scan.

Tuning Considerations The total number of I/Os needed to perform a full table scan depends on factors such as these:

- the size of the table
- the multi-block read count
- whether parallel query is being utilized for the operation

In general, a large value gives cheaper table scan cost and favors table scans over indexes. The default value is 8. OLTP and batch environments typically have values for this parameter in the range of 4 to 16. DSS database environments tend to get the most benefit from maximizing the value for this parameter.

Optimizer Mode

Description This parameter sets the mode of the optimizer at instance startup: rule-based, cost-based optimized for throughput, cost-based optimized for response time, or choice-based on presence of statistics.

Tuning Considerations This parameter specifies the default behavior of the optimizer. In most cases, cost-based optimization will yield better results than rule-based optimization. Oracle SQL Analyze lets you test statements using all four optimizer choices, if the statement has been analyzed by the ANALYZE SQL command. You can override the default parameter using hints.

For more information about the optimizer, see *Oracle Concepts* and *Oracle Tuning*.

Session-based Initialization Parameters

The session-based initialization parameters affect memory and disk performance. You can edit these values from within Oracle SQL Analyze to simulate a different environment, or to test the effect these parameters have on your database performance.

Showing Initialization Parameters

To show these parameters, select the desired Initialization Parameters object from the Navigator window. The details window will display the initialization session parameters, their current running values, and the new value you set for it (if any).

Editing Initialization Parameters

To edit an initialization parameter setting:

1. Select the desired session object from the tree menu.

   A window opens in the right window displaying database parameters, their current running values, and the new value you set for it (if any).

2. Double-click on the value you want to change.

   A dialog window opens. The appearance of the dialog window depends on whether the value to be changed is numerical or Boolean.

3. If the value to be changed is numeric, enter the new value in the **Value** field.

   If the value is Boolean (TRUE, FALSE, or AUTO), select the appropriate radio button.

> **Note:** The changes you make affect only the currently selected tuning session, not the database itself.

The following initialization parameters are available:

Hash Area Size

Description Maximum amount of memory, in bytes, to be used for hash joins.

Tuning Considerations A larger value causes hash join costs to be cheaper, causing the optimizer to choose more hash joins. If it is too large, the system may run out of memory. If this parameter is not set, its value defaults to twice the value of the Sort Area Size parameter.

The recommended value is approximately half of the square root of S, where S is the size (in megabytes) of the smaller of the inputs to the join operation. The value should not be less than 1 megabyte.

Hash Join Enabled

Description Enables or disables the hash join feature.

Tuning Considerations This parameter specifies whether the optimizer should consider using a hash join as a join method. When set to FALSE, hash join is turned off; that is, it is not available as a join method that the optimizer can consider choosing. When set to TRUE, the optimizer will compare the cost of a hash join to other types of joins, and choose it if it gives the best cost. This parameter should always be set to TRUE for data warehousing applications.

Hash Multi-Block I/O Count

Description Specifies how many sequential blocks a hash join reads and writes in I/O.

Tuning Considerations This parameter strongly affects performance, because it controls the number of partitions into which the input is divided. A larger value causes hash join costs to be cheaper, giving more hash joins.

This parameter rarely needs to be changed. If you do change this parameter, try to make sure that the following formula remains true:

```
R/M <= Po2(M/C)
```

where R = size of(left input to the join), M=(Hash area size)*0.9, Po2(n)=largest power of 2 that is smaller than n, and C = (Hash Multi-block I/O Count)*(Database Block Size).

Optimizer Index Cost Adjustment

Description Lets you adjust the costing of index access paths in the cost-based optimizer.

Tuning Considerations This parameter makes the optimizer more or less prone to selecting an index access path over a full table scan. The default for this parameter is 100 percent, which makes the optimizer cost index access paths at the regular cost. Any other value will make the optimizer cost the access path at that percentage of the regular cost. For example, setting it to 50 percent will make the index access path look half as expensive as normal. The legal range of values for this parameter is 1 to 10000 percent. This parameter can be used to tune the

performance of a system where it is felt that the optimizer chooses too few or too many index access paths. The adjustment does not apply to user-defined cost functions for domain indexes.

Optimizer Maximum Permutation

Description Lets you limit the amount of work the optimizer spends on optimizing queries with large joins.

Tuning Considerations By restricting the number of permutations of the tables the optimizer will consider, you can ensure that the parse time for the query stays within acceptable limits. However, in doing so, there is a slight risk that the optimizer will overlook a good plan it would otherwise have found. The default value for this parameter is 80000, which corresponds to the old behavior. Setting this parameter to a value less than 1000 should ensure parse times of a few seconds or less.

Optimizer Percent Parallel

Description Determines how aggressively the optimizer will attempt to parallelize a given execution plan. The default of 0 means that the optimizer chooses the best serial plan. A value of 100 means that the optimizer uses each object's degree of parallelism in computing the cost of a full table scan.

Tuning Considerations Low values favor indexes, high values favor table scans.

Optimizer Search Limit

Description The maximum number of tables in the FOM clause for which all possible join permutations will be considered.

Tuning Considerations This parameter specifies the search limit for the optimizer. Its recommended value is: 100 / number_of_concurrent_users

Partition View Enabled

Description Enables partition views.

Tuning Considerations This parameter often needs to be set in a data warehousing applications. If Partition View Enabled is set to TRUE, the optimizer prunes (or

skips) unnecessary table accesses in a partition view. This parameter also changes the way the cost-based optimizer computes statistics on a partition view from statistics on underlying tables.

Partition View Enabled

Description Enables partition views.

Tuning Considerations If Partition View Enabled is set to TRUE, the optimizer prunes (or skips) unnecessary table accesses in a partition view.. This parameter also changes the way the cost-based optimizer computes statistics on a partition view from statistics on underlying tables.

Sort Area Size

Description Specifies the maximum amount, in bytes, of Program Global Area (PGA) memory used for a sort.

Tuning Considerations If your system has a lot of memory, you can benefit from setting Sort Area Size to a large value. This can dramatically increase the performance of hash operations and large sorts, because the entire operation is more likely to be performed in memory.

If the sort area is too small, data is divided into smaller pieces and each piece, or *run*, is sorted individually. An excessive amount of I/O is then required to merge the runs back together in to a single sort. If the sort area size is very small, there will be many runs to merge, and multiple passes may be necessary. The amount of I/O increases as the sort area size decreases.

If the sort area is too large, the operating system paging rate will be excessive. The cumulative sort area adds up fast, because each parallel server can allocate this amount of memory for each sort.

If memory is a concern for your system, you may want to limit the amount of memory allocated for sorts and hashing operations. Instead, increase the size of the buffer cache so that data blocks from temporary sort segments can be cached in the buffer cache.

Star Transformation Enabled

Description Determines whether a cost-based query transformation will be applied to star queries.

Tuning Considerations If set to TRUE, the optimizer will consider performing a cost-based query transformation on the star query. If set to FALSE, the transformation will not be applied. For more information, see *Oracle8i Concepts*.

Timed Statistics

Description Sets the statistics related to time to zero, or sets them to be recorded.

Tuning Considerations If this value is FALSE, the statistics related to time are always zero and the server can avoid the overhead of requesting the time from the operating system. A FALSE setting will also disable the Query Progress Monitor. A TRUE setting enables the Query Progress Monitor and provides timing statistics for server operations.

Sort Direct Writes

Description This parameter controls whether sort data will bypass the buffer cache to write intermediate sort results to disk.

Tuning Considerations Sort Direct Writes can improve sort performance if memory and temporary space are available on your system.

When set to TRUE, additional buffers are allocated from memory during each sort, making sort costs lower and encouraging the optimizer to use more sort joins.

When set to the default of AUTO, and the value of the Sort Area Size is greater than ten times the block size, memory is allocated from the sort area.

When set to FALSE, the sorts that write to disk write through the buffer cache.

For more information, see *Oracle Tuning*.

Note: TFor more information about the following NLS parameters, see *Oracle8i National Language Support Guide*. Also see the *Oracle8i Administrator's Guide*. Note that changing certain parameters will affect other parameters. Most notably, changing NLS Territory affects several other parameters that will automatically change their values as a result.

NLS Calendar

Description Specifies which calendar system Oracle uses. This parameter can have one of the following values: Arabic Hijrah, English Hijrah, Gregorian, Japanese Imperial, Persian, ROC Official (Republic of China), or Thai Buddha.

Tuning Considerations None.

NLS Comparison

Description NLS Comparison can be used to indicate that the comparisons must be linguistic according to the NLS Sort session parameter.

Tuning Considerations Lets you avoid the cumbersome process of using NLS Sort in SQL statements. Normally, comparison in the WHERE clause is binary. To use linguistic comparison, the NLS Sort function must be used. Sometimes this can be tedious, especially when the linguistic sort needed has already been specified in the NLS Sort session parameter.

NLS Currency

Description Specifies the string to use as the local currency symbol for the L number format element. The default value of this parameter is determined by NLS Territory.

Tuning Considerations None.

NLS Date Format

Description Specifies the default date format to use with the TO_CHAR and TO_ DATE functions. The default value of this parameter is determined by NLS

Territory. The value of this parameter can be any valid date format mask, for example, MM/DD/YYYY.

Tuning Considerations None.

NLS Date Language

Description Specifies the language to use for the spelling of day and month names and date abbreviations (AM, PM, AD, BC). The default value of this parameter is the language specified by NLS Language.

Tuning Considerations None.

NLS ISO Currency

Description Specifies the string to use as the international currency symbol for the C number format element. The default value of this parameter is determined by NLS Territory.

Tuning Considerations None.

NLS Language

Description Specifies the default language of the database. This language is used for messages, the day and month names, the symbols for AD, BC, AM, and PM, and the default sorting mechanism. Examples of supported languages are American, French, and Japanese. This parameter determines the default values of the parameters NLS Date Language and NLS Sort.

Tuning Considerations None.

NLS Numeric Characters

Description Specifies the characters to use as the group separator and decimal and overrides those defined implicitly by NLS Territory. The group separator is the character that separates integer groups (that is, the thousands, millions, billions, and so on). The decimal separates the integer portion of a number from the decimal portion.

Any character can be the decimal or group separator. The two characters specified must be single-byte, and both characters must be different from each other each other. The characters cannot be any numeric character or any of the following characters: plus (+), hyphen (-), less than sign (<), greater than sign (>). The default value of this parameter is determined by NLS Territory.

Tuning Considerations None.

NLS Sort

Description Specifies the collating sequence for ORDER BY queries.

Tuning Considerations If the value is BINARY, then the collating sequence for ORDER BY queries is based on the numeric value of characters (a binary sort that requires less system overhead). If the value is a named linguistic sort, sorting is based on the order of the defined linguistic sort. Most languages supported by the NLS_LANGUAGE parameter also support a linguistic sort with the same name.

Setting NLS Sort to anything other than BINARY causes a sort to use a full table scan, regardless of the path chosen by the optimizer. BINARY is the exception because indexes are built according to a binary order of keys. Thus the optimizer can use an index to satisfy the ORDER BY clause when NLS Sort is set to BINARY. If NLS_SORT is set to any linguistic sort, the optimizer must include a full table scan and a full sort into the execution plan.

The default value of this parameter depends on the value of the NLS_LANGUAGE parameter.

For more information on this parameter, see the *Oracle Administrator's Guide.*

NLS Territory

Description Specifies the name of the territory whose conventions are to be followed for day and week numbering. Also specifies the default date format, the default decimal character and group separator, and the default ISO and local currency symbols. Supported territories include America, France, Japan, and so on.

Tuning Considerations This parameter determines the default values for the following parameters: NLS Currency, NLS ISO Currency, NLS Date Format, and NLS Numeric Characters.

NLS Union Currency

Description This parameter can be used to override the default dual currency symbol defined in the territory. When starting a new session without setting NLS Union Currency, the default dual currency symbol defined in the territory of your current language environment will be used. When you set this parameter, you will start up a session with its value as the dual currency symbol.

Tuning Considerations None.

Analyzing Logical Structure

Tables, views, indexes, clusters; these objects are created to better manage and provide faster, more efficient access to the data in your database. These same objects, if not monitored, can grow to consume excessive memory, become inefficient if they are not modeled to reflect current user behavior, or may even become obsolete and unused.

Oracle SQL Analyze provides Object Properties information that can help you determine whether an index or cluster is efficient, or whether you might want to consider editing or re-creating some of these logical structures.

Viewing Object Properties

SQL Analyze shows you the details of any table, index, cluster, or view used by an explain plan.

You can view these objects from an explain plan in the following way:

1. Select an entry in the explain plan that concerns the object. You can determine this by finding the object's name in the Object Name column.
2. Right click on the entry.

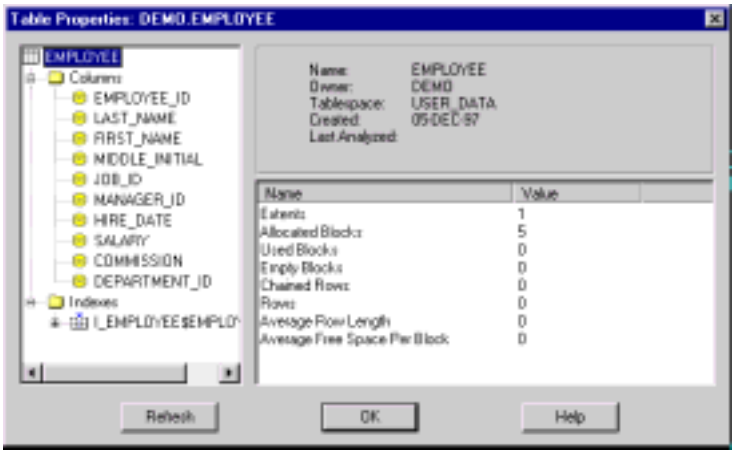
   A menu appears with an **Object Properties** choice.
3. Select **Object Properties**.

   The Object Properties dialog window opens, as shown on page 5-17.

In the Object Properties dialog, you can select table, index, cluster, and view statistics, if applicable. The meaning of the statistics is presented in the following sections.

Note: The following information about object properties and their tuning implications is meant as an introduction only. Please consult *Oracle Server Administration, Oracle Server Concepts, Oracle Server Tuning, Oracle Server Reference,* and the *Oracle Server Application Developers Guide* for more detailed information.

Figure 5–2 Object Properties Dialog



Viewing General Properties

The general details provided about the object are:

- the object's name
- owner
- tablespace location
- creation date
- the date the object was last ANALYZED. If the statistics are not current, this will impact the effectiveness of the cost-based optimizer.
- the object's type. If a Cluster object is selected, the type is INDEX or HASH. If an Index object is selected, the type is UNIQUE or NON-UNIQUE.

If an Index object is selected, the dialog displays the following for the index identified in the **Name** field:

Name
The key column on which the index is based.

Position
Column position in the index.

Type
Datatype of the key column.

Nullable
Whether a value in the column requires a value, or can be left blank (NULL).

Table Properties

Tables are the basic units of data storage in an Oracle database. When you create a table, Oracle allocates to the table's data segment an initial extent of a specified number of data blocks. When these blocks become full, database performance declines. Therefore, it's important to monitor the statistics listed in the following to ensure that your database is using its existing space efficiently, and that your database has the space it needs.

If you determine that block space allocated to tables is a problem, you will want to consider PCTFREE and PCTUSED parameters to:

- increase the performance of writing and retrieving a data or index segment
- decrease the amount of unused space in data blocks
- decrease the amount of row chaining between data blocks

For hints on managing tables, see the *Oracle Server Application Developer's Guide.*

To display Table properties, open the Object Properties dialog as described in "Viewing Object Properties."

You can review the following statistics in the Table Properties page:

Extents

Description A specific number of contiguous data blocks allocated for storing a specific type of information.

Tuning Considerations Proper sizing of extents is a key factor in managing the performance of full table scans. If the extents are not properly sized, the number and size of the extents can greatly increase the amount of work performed by the database during a full table scan. You can judge the size of an extent by the number of data blocks per extent.

Database objects frequently read via large scans—either full table scans or large index range scans—should be stored in a small number of extents. Keeping the number of extents small makes it more likely that the next data to be read is physically near the data currently being read.

To eliminate the potential impact of multiple extents on performance, you need to make sure that the size of each extent is a multiple of the number of blocks read during each multiblock read (see "Database File Multiblock Read Count" in the section on Database Parameters). In many systems, 64 Kb or 128 Kb is read during each read. Therefore, size your extents so that each is an even multiple of 64KB or 128 KB.

Allocated Blocks

Description When you create a table, Oracle allocates to the table's data segment an initial extent of a specified number of data blocks. Although no rows have been inserted yet, the Oracle blocks that correspond to the initial extent are reserved or allocated for that table's rows.

Tuning Considerations Indicates size of extents.

Used Blocks

Description Blocks that have already been allocated to a table.

Tuning Considerations Indicates size of extents.

Empty Blocks

Description Blocks that Oracle considers available to be used in a table's rows.

Tuning Considerations Indicates size of extents.

Chained Rows

Description If a row is too large to fit into one data block when it is first inserted, Oracle stores the data for the row in a chain of data blocks (one or more) reserved for that segment. Row chaining most often occurs with large rows, such as rows that contain a column of datatype LONG or LONG RAW.

Tuning Considerations Chained rows reduce I/O performance associated with these rows.

Rows

Description The value is the number of rows in the table.

Tuning Considerations Indicates size of table, number of rows that will need to be scanned in a Full Table Scan.

Average Row Length

Description The average length of a row in the table, in bytes.

Tuning Considerations Processing costs increase if updates to rows or index entries cause rows to grow and span blocks (chained rows).

Average Free Space Per Block

Description The average free space of all blocks on a free list. A *free list* is a list of database blocks that have been allocated for the segment's extents and have free space greater than the PCTFREE setting.

Tuning Considerations The more densely packed the rows are within your tables' blocks, the fewer blocks you will need to read. Each database block has a header/trailer area, an area used by the rows' data, and free space. To improve the row density within your blocks, you need to consider all three of these areas during your space management efforts.

Cluster Properties

Clusters store related rows of different tables together in the same data blocks. You get two primary benefits from this:

- Disk I/O is reduced and access time improves for joins of clustered tables.
- In a cluster, a cluster key value (that is, the related value) is only stored once, no matter how many rows of different tables contain the value. Therefore, less storage may be required to store related table data in a cluster than is necessary in non-clustered table format.

To identify data that would be better stored in clustered form, look for tables that are related through referential integrity constraints and tables that are frequently accessed together using a join. If you cluster tables on the columns used to join table data, you reduce the number of data blocks that must be accessed to process the query; all the rows needed for a join on a cluster key are in the same block.

Conversely, if the all the rows for a cluster key cannot fit in a single block, they may cause join statements to consume more resources.

Also note that clusters can reduce the performance of DML statements (INSERTs, UPDATEs, and DELETEs) as compared to storing a table separately with its own index. These disadvantages relate to the use of space and the number of blocks that must be visited to scan a table. Because multiple tables share each block, more blocks must be used to store a clustered table than if that same table were stored non-clustered. You should decide about using clusters with these trade-offs in mind.

For information on managing clusters, see *Oracle Server Concepts* and the *Oracle Server Application Developers Guide*.

To display Cluster Details:

1. Open the Object Properties dialog as described in "Viewing Object Properties."
2. Select a cluster object.

The Cluster Statistics dialog displays the following statistics:

Extents

Definition A specific number of contiguous data blocks allocated for storing a specific type of information.

Tuning Considerations See description of table extents on page 5-18

Allocated Blocks

Description Blocks allocated to contain cluster keys and their associated rows.

Tuning Considerations Indicates the size of the clusters.

Average Blocks Per Cluster Key

Description A value reached by dividing the number of blocks in the table by the number of hash keys.

Tuning Considerations By default, Oracle stores only one cluster key and its associated rows in each data block of the cluster's data segment. If all the rows for a given cluster key value cannot fit in one block, the blocks are chained together to speed access to all the values within a given key. Too few rows per cluster key value can waste space and result in negligible performance gains. Too many rows can cause the optimizer to perform excessive searches to find rows for that key.

Empty Blocks

Description Allocated blocks that are still available for storing cluster keys and their associated rows.

Tuning Considerations Indicates size of extents.

Distinct Hash Values

Description Number of distinct hash values, which are based on specific cluster key values.

Tuning Considerations Hash clusters are used to store individual tables or a group of clustered tables that are static and often queried by equality queries. Hashing is an optional way of storing table data to improve the performance of data retrieval. To use hashing, you create a hash cluster and load tables into the cluster.

Hashing is most advantageous when you have the following conditions:

- Most queries are equality queries on the cluster key:

  ```
  SELECT . . . WHERE cluster_key = . . . ;
  ```

In such cases, the cluster key in the equality condition is hashed, and the corresponding hash key is usually found with a single read. In comparison, for an indexed table the key value must first be found in the index (usually several reads), and then the row is read from the table (another read).

- The tables in the hash cluster are primarily static in size so that you can determine the number of rows and amount of space required for the tables in the cluster. If tables in a hash cluster require more space than the initial allocation for the cluster, performance degradation can be substantial because overflow blocks are required.

Column Statistics

Selecting the Column Statistics radio button displays the following column details:

Column Name

The common column or columns that are shared by the tables in the cluster.

Distinct Values

Number of unique values in the column.

Density

Number of times a distinct value appears in the column, divided by the number of distinct values.

Index Properties

Indexes are used in Oracle to provide quick access to rows in a table. Indexes provide faster access to data for operations that return a small portion of a table's rows. While indexes can save considerable time, the SQL engine must maintain all indexes defined against a table whether they are used or not, presenting a significant CPU and I/O demand on any intensive I/O application. Therefore, you should drop indexes that are not used.

A method to determine whether an index is good is to create it, analyze it, and use SQL Analyze to run explain plans on your query to see if the optimizer uses it. If it does, keep the index unless it is very expensive to maintain. Another method is to compare the optimizer cost of the plans with and without the index.

It should be noted, however, that indexes sometimes have uses that are not immediately apparent from a survey of statement execution plans. In particular, Oracle8 uses "pins" (nontransactional locks) on foreign key indexes to avoid the need for share locks on the parent table when enforcing foreign key constraints. In many applications this foreign key index never (or rarely) supports a query.

For guidelines on creating and managing indexes, see *Oracle Server Concepts*, the *Oracle Server Application Developer's Guide*, and *Oracle Server Tuning*.

To display Index Details:

1. Open the Object Properties dialog as described in "Viewing Object Properties."
2. Select an index object.

The Index Properties dialog displays the following statistics:

Extents

Description A specific number of contiguous data blocks allocated for storing a specific type of information.

Tuning Considerations See description of table extents on page 5-18.

Allocated Blocks

Description Blocks allocated to contain index keys and their associated rows.

Tuning Considerations Indicates size of index.

Tree Depth

Description Depth of the B-Tree index.

Tuning Considerations: If this value is greater than 4 (the B-tree index "branches out" four levels or more), consider dropping and re-creating this index.

Leaf Blocks

Description Number of leaf blocks, the lowest level index blocks in a B-tree index, in the current index.

Tuning Considerations: The lowest level index blocks (leaf blocks) contain every indexed data value and a corresponding ROWID used to locate an actual row. This value indicates the size and selectivity of the index.

Distinct Keys

Description Number of distinct indexed values.

Tuning Considerations If this value is low, a bitmap index may be more effective than a B*-tree index for accessing the data.

Average Leaf Blocks Per Key

Description Average number of leaf blocks in which each distinct value in the index appears. This statistic is rounded to the nearest integer.

Tuning Considerations Indicates selectivity of the index. The higher the value, the more rows it will select in a query. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is always 1.

Average Data Blocks Per Key

Description Average number of data blocks in the table that are pointed to by a distinct value in the index. This statistic is the average number of blocks that contain rows that contain a given value for the indexed columns. It is rounded to the nearest integer.

Tuning Considerations Indicates selectivity of the index. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is always 1.

Clustering Factor

Description Represents the amount of order of the rows in the table based on the values of the index.

Tuning Considerations: If the value of the index is near the number of blocks, then the table is very well ordered. In such a case, the index entries in a single leaf block tend to point to rows in the same data blocks. If its value is near the number of rows, then the table is very randomly ordered. In such a case, it is unlikely that the index entries in the same leaf block point to rows in the same data blocks.

Examining Views

Views derive their data from the tables on which they are based, referred to as the *base tables* of the views. Base tables can in turn be tables or can themselves be views. Like tables, views can be queried, updated, inserted into, and deleted from, with restrictions. All operations performed on a view actually affect the base tables of the view.

At times, it is useful to understand the selection criteria behind the view.

To examine a view:

1. Select a view from the explain plan.
2. Open the Object Properties dialog as described in "Viewing Object Properties."
3. Select a View object.
4. The SQL that creates the view is shown in the View Properties dialog.

Understanding the Oracle Optimizer

Optimization is the process of choosing the most efficient way to execute a SQL statement. This is an important step in the processing of any data manipulation language (DML) statement: SELECT, INSERT, UPDATE, or DELETE. Many different ways to execute a SQL statement often exist, for example, by varying the order in which tables or indexes are accessed. The procedure Oracle uses to execute a statement can greatly affect how quickly the statement executes.

A part of Oracle called the *optimizer* chooses what it believes to be the most efficient way. The optimizer evaluates a number of factors to select among alternative access paths. The access paths the optimizer chooses can be displayed by generating *explain plans.*

Although it is beyond the scope of this guide to fully explain the optimizer and how it selects access paths, the section following will explain some of the basic concepts that will help you better understand explain plans and how to identify inefficiencies in them.

For a more detailed explanation of the Oracle optimizer, see *Oracle Server Concepts.*

Cost-Based and Rule-Based Optimization

To choose an execution plan for a SQL statement, the optimizer uses one of two approaches: cost-based or rule-based.

The Cost-Based Approach

Using the cost-based approach, the optimizer determines which execution plan is most efficient by considering available access paths and factoring in information based on statistics in the data dictionary for the schema objects (tables, clusters, or indexes) accessed by the statement. The cost-based approach also considers hints, or optimization suggestions placed in a Comment in the statement.

Conceptually, the cost-based approach consists of these steps:

1. The optimizer generates a set of potential execution plans for the statement based on its available access paths and hints.
2. The optimizer estimates the cost of each execution plan based on the data distribution and storage characteristics statistics for the tables, clusters, and indexes in the data dictionary.

   The *cost* is an estimated value proportional to the expected resource use needed to execute the statement using the execution plan. The optimizer calculates the cost based on the estimated computer resources, including (but not limited to) I/O, CPU time, and memory, that are required to execute the statement using the plan.

   Serial execution plans with greater costs take more time to execute than those with smaller costs. When using a parallel execution plan, however, resource use is not directly related to elapsed time.
3. The optimizer compares the costs of the execution plans and chooses the one with the smallest cost.

Goal of the Cost-Based Approach

By default, the goal of the cost-based approach is the best throughput, or minimal resource use necessary to process all rows accessed by the statement.

Oracle can also optimize a statement with the goal of best response time, or minimal resource use necessary to process the first row accessed by a SQL statement.

Statistics for the Cost-Based Approach

The cost-based approach uses statistics to estimate the cost of each execution plan. These statistics quantify the data distribution and storage characteristics of tables, columns, indexes, and partitions. You can generate these statistics using the ANALYZE command. The optimizer uses these statistics to estimate how much I/O, CPU time, and memory are required to execute a SQL statement using a particular execution plan.

When to Use the Cost-Based Approach

Oracle SQL Analyze allows you to test both rule-based and cost-based approaches. However, it is useful to note the following guidelines for selecting either of the cost-based approaches. In general, you should use the cost-based approach for all new applications; the rule-based approach is provided for applications that were written before cost-based optimization was available. Cost-based optimization can be used for both relational data and object types.

The following features can only use cost-based optimization:

- partitioned tables
- partition views
- index-organized tables
- reverse key indexes
- bitmap indexes
- parallel query and parallel DML
- star transformation
- star join

The cost-based approach generally chooses an execution plan that is as good as or better than the plan chosen by the rule-based approach, especially for large queries with multiple joins or multiple indexes. The cost-based approach also improves productivity by eliminating the need to tune your SQL statements yourself. Finally, many Oracle performance features are available only through the cost-based approach.

Cost-based optimization must be used for efficient star query performance. Similarly, it must be used with hash joins and histograms. Cost-based optimization is always used with parallel query and with partitioned tables. You must use the ANALYZE command in order to keep the statistics current.

The Rule-Based Approach

Using the rule-based approach, the optimizer chooses an execution plan based on the access paths available and the ranks of these access paths. You can use rule-based optimization to access both relational data and object types.

Oracle's ranking of the access paths is heuristic. If there is more than one way to execute a SQL statement, the rule-based approach always uses the operation with the lower rank. Usually, operations of lower rank execute faster than those associated with constructs of higher rank.

Access Methods

This section describes basic methods by which Oracle can access data.

Full Table Scans

A full table scan retrieves rows from a table. To perform a full table scan, Oracle reads all rows in the table, examining each row to determine whether it satisfies the statement's WHERE clause. Oracle reads every data block allocated to the table sequentially, so a full table scan can be performed very efficiently using multiblock reads. Oracle reads each data block only once.

Table Access by ROWID

A table access by ROWID also retrieves rows from a table. The ROWID of a row specifies the datafile and data block containing the row and the location of the row in that block. Locating a row by its ROWID is the fastest way for Oracle to find a single row.

To access a table by ROWID, Oracle first obtains the ROWIDs of the selected rows, either from the statement's WHERE clause or through an index scan of one or more of the table's indexes. Oracle then locates each selected row in the table based on its ROWID.

Cluster Scans

From a table stored in an indexed cluster, a cluster scan retrieves rows that have the same cluster key value. In an indexed cluster, all rows with the same cluster key value are stored in the same data blocks. To perform a cluster scan, Oracle first obtains the ROWID of one of the selected rows by scanning the cluster index. Oracle then locates the rows based on this ROWID.

Hash Scans

Oracle can use a hash scan to locate rows in a hash cluster based on a hash value. In a hash cluster, all rows with the same hash value are stored in the same data blocks. To perform a hash scan, Oracle first obtains the hash value by applying a hash

function to a cluster key value specified by the statement. Oracle then scans the data blocks containing rows with that hash value.

Index Scans

An index scan retrieves data from an index based on the value of one or more columns of the index. To perform an index scan, Oracle searches the index for the indexed column values accessed by the statement. If the statement accesses only columns of the index, Oracle reads the indexed column values directly from the index, rather than from the table. The index contains not only the indexed value, but also the ROWIDs of rows in the table having that value. Therefore, if the statement accesses other columns in addition to the indexed columns, Oracle can find the rows in the table with a table access by ROWID or a cluster scan.

See *Oracle Server Concepts* for a list of index scan types.

Understanding Performance Statistics

Oracle SQL Analyze gives you several ways to monitor and examine the performance of your SQL code:

- Use TopSQL to analyze and sort statements that have already been run, by the resources they consume.
- Generate different explain plans to learn how the optimizer executes a statement. Within an explain plan, you can also examine object details and execution statistics.
- Generate a Compact View to study the join methodologies selected by the optimizer.
- Execute the statement from within Oracle SQL Analyze and examine the statistics.
- Compare explain plans and execution statistics against each other.

The rest of this chapter discusses how to view performance statistics through Oracle SQL Analyze, including an explanation of explain plans and how Oracle SQL analyze can help you navigate through them more easily.

TopSQL Statistics

As discussed in ""Selecting a Statement for Tuning"," TopSQL is an integrated function of Oracle SQL Analyze that lets you measure the resources a SQL statement consumes. Using these statistics, you can determine which statements consume the most resources and select them for tuning.

A TopSQL object exists for each database session represented in the Navigator window. TopSQL displays statistics from the V$SQLAREA view that show you the resources consumed by a statement and help you identify performance problems.

The performance statistics it shows are discussed in Chapter 4, "Starting a Tuning Session".

Using SQL History

The SQL History is a repository of SQL statements gathered from the database SQL cache (using the V$SQLAREA and V$SQLTEXT views).

Shared by Oracle SQL Analyze, Oracle Expert, and the Oracle Tuning wizard, the SQL History can be updated from Oracle SQL Analyze or Oracle Expert. If you update on a regular basis from either program, the time it takes to gather statistics for an analysis or tuning session can be greatly reduced.

SQL History is used for tuning operations that require a larger view of the SQL activity for the database, such as index tuning. As a set of SQL statements collected over a period of time, SQL History gives you a more accurate representation of the database and its use.

SQL History menu commands are the same as the TopSQL commands, with two differences:

- **History =>Update** adds the latest database statistics to the SQL History.
- **History=>Replace** replaces the SQL History with the current statistics. This is useful when you make extensive changes to the database and the past history is no longer relevant to current conditions.

Understanding Explain Plans

A SQL statement that retrieves data from many tables can use many variations of table join methods, join orders, and access paths to produce the same set of results. The Oracle optimizer must figure out the optimal path for these operations based on a multitude of factors, such as, but not limited to:

- available indexes

- the order of tables and columns in the SQL statement
- statistics on the cardinality of objects referenced in the statement
- hints

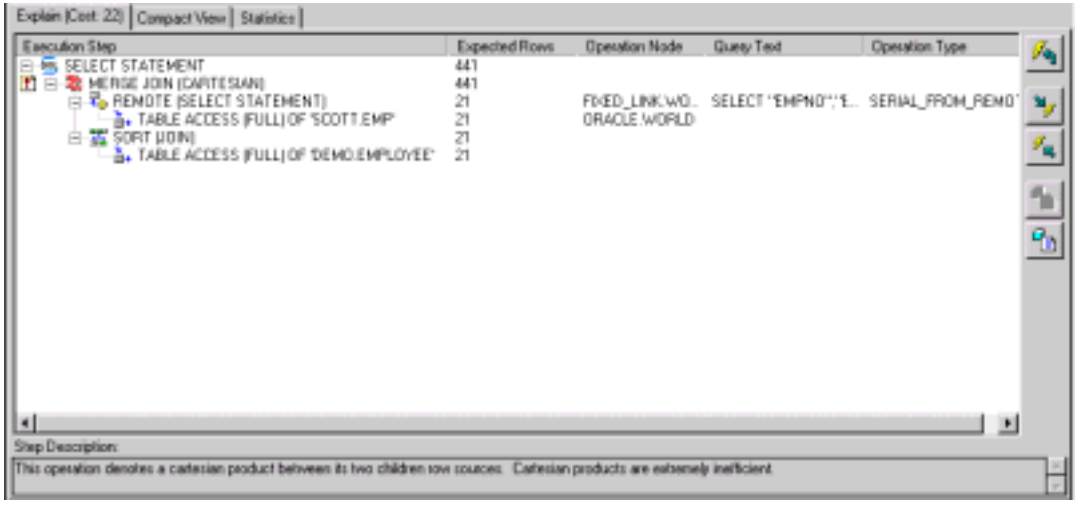
These factors will vary depending on the approach used by the optimizer, whether it is rule-based, cost-based for response time, or cost-based for throughput. The SQL statement's execution path can be displayed through an explain plan, which provides a list of the operations involved in the statement's execution. By examining the execution plan, you can see exactly how Oracle executes your SQL statement.

Oracle SQL Analyze provides a facility for easily generating explain plans that can be used to assess how a given SQL statement will perform under different optimizer modes. You can run a SQL node under each optimizer mode to produce an explain plan for the statement and the execution "cost"—a measurement of several factors, including the amount of computer resources (I/O and CPU consumption, for example), and the time to complete the execution of the statement— if a cost-based optimizer is used.

Generating an Explain Plan

To generate an explain plan, select the desired optimization path from the **SQL => Explain** menu choice. An explain plan appears in the Details window, as shown in Figure 5–3, and an explain plan object is added in the Navigator window, connected to the related SQL statement.

Figure 5–3 Explain Plan



You can select four different optimization paths through which to view an explain plan for a SQL statement:

- Rule (Rule-based optimization)
- Cost first rows (Cost-based optimization for response time)
- Cost all rows (Cost-based optimization for throughput)
- Choose (Optimizer's choice)

You can learn to follow the sequence of an explain plan by reading the section below, and get more information on explain plans in *Oracle Server Tuning*. An easier approach, however, is to let Oracle SQL Analyze Oracle guide you as you "step through" a plan.

Reading An Explain Plan

To read an explain plan, you need to know where the processing will begin, and then follow the path, as described below. In the following SQL statement:

```
SELECT "name", product_id, amount_in_stock, state
FROM inventory, product, warehouse
WHERE product.id = inventory.product_id
AND amount_in_stock > 500
AND warehouse.id = inventory.warehouse_id;
```

is represented by the following explain plan using the rule-based optimizer:

```
SELECT STATEMENT
   NESTED LOOPS
NESTED LOOPS
   TABLE ACCESS (BY ROWID) OF 'INVENTORY'
      INDEX (RANGE SCAN) OF 'AMOUNT_IN_STOCK_PK' (NON-UNIQUE)
   TABLE ACCESS (BY ROWID) OF 'WAREHOUSE'
      INDEX (UNIQUE SCAN) of 'WAREHOUSE_ID_PK' (UNIQUE)
   TABLE ACCESS (BY ROWID) of 'PRODUCT'
      INDEX (UNIQUE SCAN) of 'PRODUCT_ID_PK' (UNIQUE)
```

The execution path uses INVENTORY as the driving table and the execution path is as follows:

1. Oracle performs a range scan of the AMOUNT_IN_STOCK_PK index for that table.
2. After retrieving multiple ROWIDS from the index, it uses these values to retrieve rows from the INVENTORY table.
3. Oracle uses the WAREHOUSE_ID_PK index to retrieve the ROWIDS,
4. Step 3 allows Oracle to then access the WAREHOUSE table by ROWID.
5. Oracle then performs a NESTED LOOPS join of the sets returned from the two tables.
6. Operations involving the PRODUCT table are performed.
7. The final operation is a NESTED LOOPS join of the set from the Product table and the set resulting from the join on the INVENTORY and WAREHOUSE tables.

As you can see, operations are performed in an order that is inverse to their positioning in the plan.

Stepping Through Explain Plans

You can "step through" the plan and immediately understand how the statement will execute and what step each operation is performing. Each operation is highlighted in order of execution. You can control the pace of the walkthrough, and can back-up or start over at any point. As an operation is highlighted, a description of the operation is displayed in the "step description" window directly below the Explain Plan display. You can also choose to view details for any objects involved in a particular operation.

Once you have an explain plan open for viewing, you can use Oracle SQL Analyze to step through the plan in the order the steps would be executed. Oracle SQL Analyze describes the steps in SQL execution terms.

To step through an explain plan:

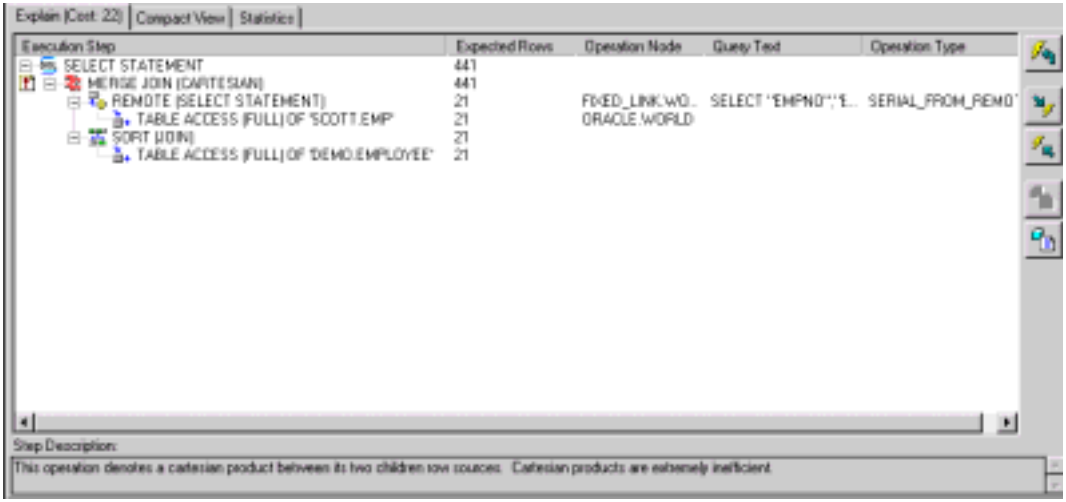
1. Select **View=>Step Description** to enable the Step Description frame.

   The Step Description frame borders the bottom and right side of the Details window. The right side contains four navigation buttons. The bottom contains a Step Description box. The Step Description frame is shown in Figure 5–4

2. The walkthrough navigation buttons allow you to navigate through the explain plan and call up object details for a selected object, such as a table, cluster, or index.

As you step through the explain plan, a description of the selected step appears at the bottom of the Details Window.

Figure 5–4 Explain Plan with Step Description



Explain Plan Details

Explain plans display the following columns:

Execution Step

The operation performed by the optimizer.

Expected Rows
The number of rows accessed by that step in the execution plan.

Explain plans for Parallel Query contain the following additional columns:

Operation Node
Describes the order in which the output from the operation is consumed.

Operation Type
Describes the type of operation being performed.

Query Text
Describes the query used by the query server.

Explain plans for Oracle8 partitions add the following three rows:

Partition Start
The start partition of a range of accessed partitions.

Partition Stop
The stop partition of a range of accessed partitions.

Partition ID
The ID of the PARTITION step that computed the pair of Partition Start and Partition Stop values.

For a full explanation of these columns and their implications, see *Oracle Server Tuning*.

Explain Plan Properties

As Oracle SQL Analyze generates an explain plan for your SQL statement, it applies certain Explain Plan Rules-of-Thumb and identifies steps in the explain plan where problems may occur. These steps are flagged by an exclamation mark. You can get the details for these Rules-of-Thumb by selecting the identified row in the explain plan and selecting the Plan Step Properties button. See "Stepping Through Explain Plans" for more details.

The Rules-of-Thumb Oracle SQL applies are:

Index merges
Index merges are often an indicator that a concatenated index could help speed a query. The rule of thumb indicator is found on any Explain Plan AND_EQUAL object, which indicates an index merge.

Cartesian products

Cartesian products are usually an indicator of a logical mistake in the SQL statement. In general, they are only intended to assist in Star Query optimization. This rule-of-thumb indicator is found on any Explain Plan MERGE JOIN (CARTESIAN) objects.

Full Table scans as non-driving tables in nested loop joins

There may be an opportunity for speeding up the access to the non-driving table when it is joined in the nested loops fashion via a full table scan lookup. Often an appropriatae concatenation of columns from the non-driving table is all thatís required to perform the lookup. The rule -of-thumb indicator is found on any Explain Plan TABLE ACCESS (FULL) object which is a child of a NESTED LOOPS object and is not the first in the chain of tables being joined.

Parallel query bottlenecks

A parallel query bottleneck is an object in a Parallel Query Option plan which indicates that the output of a serial operation in a previous object is being made parallelized in this one. The rule-of-thumb indicator will be found on the Explain Plan object which has the auxiliary data showing a PARALLEL_FROM_SERIAL parallelization has happened at this object.

Remote Queries

Remote queries that extend over distributed databases in a system are noted in the explain plans by an icon and the word REMOTE, as shown in Figure 5–4.

Stepping Through Compact Views

A second type of explain plan is the *compact view*. The compact view displays explain plans with an emphasis on the join methodologies used in the current explain plan. Joined tables are shown as peers, rather than as children, which allows you to see more clearly which tables have been joined, and what method was used to join them.

A sample compact view is shown in Figure 5–5.

Compact views show the following columns:

Execution Step

Shows the explain plan, reorganized to emphasize joins.

Join Method

Displays the type of join used for each joined table in the explain plan.

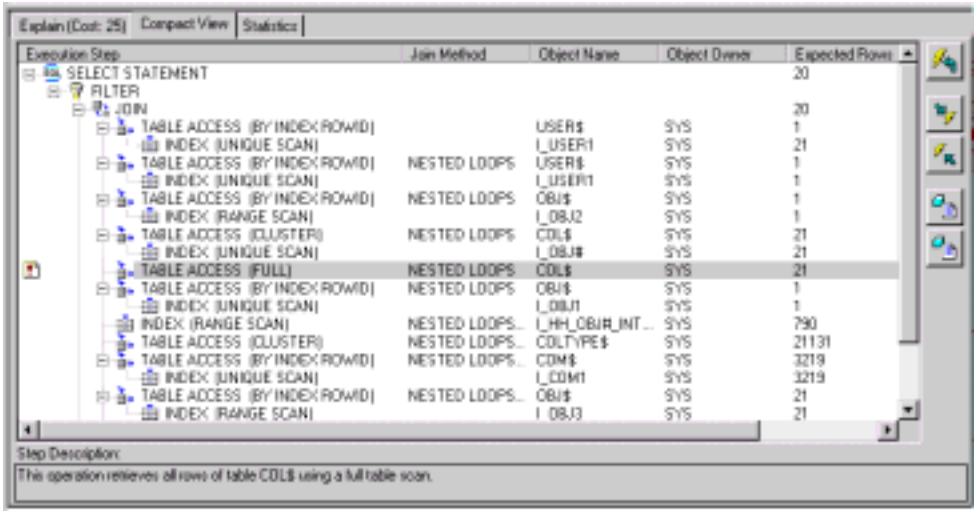
Object Name
Lists tables and indexes associated with the join.

Object Owner
The name of the session to which the object belongs.

Expected Rows
The number of rows the step is expected to fetch when the statement is executed.

Like standard explain plans, you can also walk through the compact view, study the object details, and review the execution statistics.

Figure 5–5 Compact View



Viewing Execution Statistics

Execution statistics provide information about the performance of a SQL statement as it accesses data from a database.

To view execution statistics:

1. Generate an explain plan for the selected SQL statement.
2. Select **SQL=>Execute**. It might take some time for the SQL statement to execute.
3. Select the Statistics page from the Details window.

You might want to execute the statement several times to develop an average set of statistics that might be more representational of the actual performance of the SQL statement and its explain plan.

Comparing SQL Statements and Explain Plans

Comparing explain plans is a powerful tool to analyze the improvements you have made while tuning your SQL statements. Oracle SQL Analyze lets you create a split view in which you can open two different SQL statements and compare results.

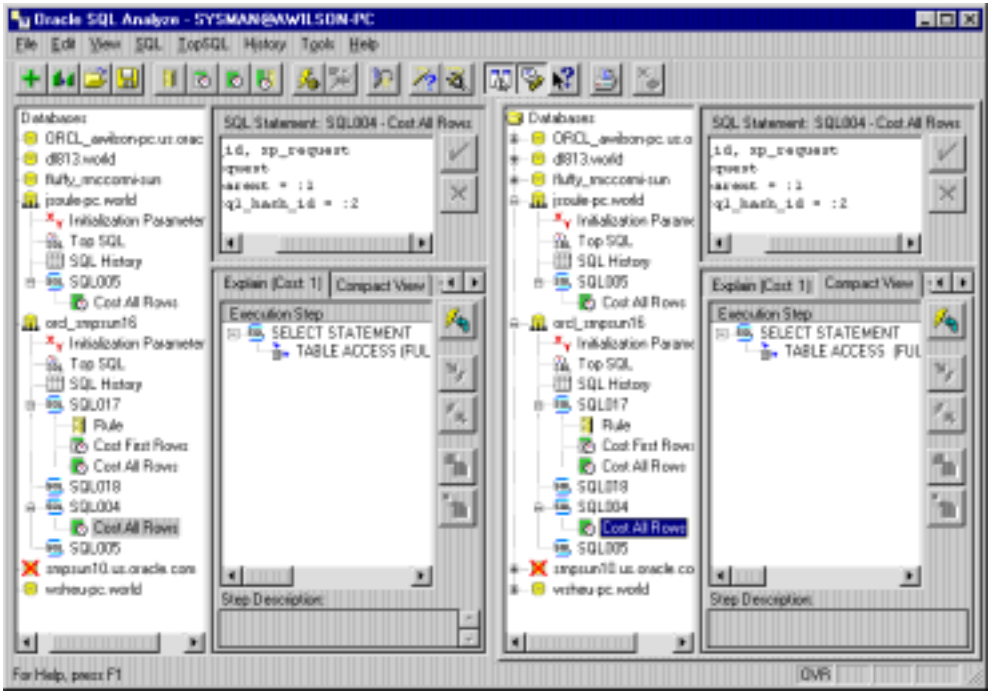
To compare SQL statements and explain plans, select **View=>Comparison** from the menu.

The main window splits into two panes, each with identical navigation and main windows as shown in Figure 5–6.

You can then navigate in either pane to display and compare alternative SQL statements, explain plans, and performance statistics.

To return to a single main window, select **View=>Comparison** again.

Figure 5–6 Comparison View



6

Tuning SQL Statements

This chapter contains the following topics:

Tuning SQL Statements

Oracle SQL Analyze is a flexible tool that allows you to approach tuning from different perspectives.

For example, you may have selected a statement through TopSQL, analyzed its performance and determined the appropriate measures to tune the statement. Whether that method is to edit the syntax manually or add hints through the Hint Wizard, you've determined exactly what you need to do.

Or you might have a statement stored in a file that you know needs tuning. Rather than examine the performance statistics, you would rather process the statement through the Tuning Wizard and have the Wizard tune your statement automatically.

This chapter first describes manual editing and methodologies, and then explains how to use the Hint and Tuning Wizards to automate the tuning process.

Editing Statements Manually

You can edit statements manually by entering text into the SQL Text window. Although Oracle SQL Analyze will not check your syntax while you are typing, you can still test the statement by generating explain plans, executing the statement, and comparing its results to previous versions of the statement, or to other statements you have analyzed.

You will not be able to edit a statement if either of the following is true:

- The statement was dragged over from the TopSQL text window.
- You have already generated an explain plan for the statement.

If either of these cases is true, select **SQL=>Create Like** to create an editable copy of the statement. A SQL statement object is created in the navigation window for the new statement. Then proceed to edit the new statement.

Understanding Index Tuning Recommendations

Effective indexing can improve the performance of your SQL statements by reducing the need for full table scans. Oracle SQL Analyze can generate recommendations that will help you improve index effectiveness, and generate scripts you can use to carry out the recommendations.

Note: The Index recommendations feature is designed to tune indexes that will be used by the Oracle cost-based optimizer. It can not tune indexes used with the rule-based optimizer.

The index tuning evaluation can take several minutes depending on the number of tables being evaluated and the number of SQL statements in the SQL History. You can use Oracle SQL Analyze for other SQL tuning operations while this operation is being performed.

If Oracle SQL Analyze determines that index changes are necessary, it will provide a tree list of index recommendations. The recommendations are organized by table.

You can view the details for a recommendation by either double-clicking on the recommendation or highlighting it and using the right mouse button to access Recommendation Details. The Recommendation Details provide some important information, such as:

- A description of the index, its type and status.
- The type of workload: OLTP, Data Warehousing or Multipurpose.

- The cardinality of the table and recommended index. Table cardinality means the number of rows in the table. Index cardinality means the number of unique entries in the index.
- The recommended columns for the index.
- A count of the number of times that each of the recommended columns was referenced within the SQL statements included in the evaluation. This also describes how the column was used in the SQL statements, such as the number of times that the column was used to fulfill an equality, inequality or "order by" condition.
- The SQL statements evaluated for the index recommendation, including the number of times the statement was executed and the number of disk reads performed by the SQL statement. The calculated importance for each SQL statement is also provided. This importance rating is calculated by SQL Analyze and used behind the scenes to rank the SQL statements during the index evaluation.

Getting Index Recommendations

You can perform index tuning for a selected SQL statement by selecting **SQL=>Get Index Recommendations**.

> **Note:** The **Get Index Recommendations feature** is designed to tune indexes that will be used by the Oracle cost-based optimizer. You should not use this feature to tune indexes on tables that use the Oracle rule-based optimizer.

The **Get Index Recommendations** feature will identify the tables accessed in the SQL statement and will scan the databaseís SQL History for any other SQL statements that reference those tables. This subset of the SQL History is then used to evaluate the index requirements for the target tables. In this way, index tuning for a selected SQL statement is performed in the context of the larger SQL workload.

The index tuning evaluation can take several minutes depending on the number of tables being evaluated and the number of SQL statements in the SQL History. You can use SQL Analyze for other SQL tuning operations while this operation is being performed. If SQL Analyze determines that index changes are necessary it will

provide a tree list of index recommendations. The recommendations are organized by table.

Once you have index recommendations you can:

- Review the details of each recommendation by selecting **SQL=> Show Recommendation Details**.
- Create a SQL script that can be used to implement the recommendations by selecting **SQL=>Generate Recommendation Script**.

Generating Tuning Recommendation Scripts

You can create a SQL script to implement Tuning Index recommendations:

1. Highlight the SQL object in the navigator tree that contains the Index Recommendations, and select **SQL=>Generate Recommendation Script**.
2. A **Save As** dialog box appears.
3. Enter the file name and path for the recommendation script.
4. The file is saved in the specified location. You can edit and execute this script using the Oracle Enterprise Manager SQL Worksheet.

Alternatively, you can schedule the Recommendation Script for execution using the Oracle Enterprise Manager Console Job System. Refer to the *Oracle Enterprise Manager Administrator's Guide* for information on this feature.

Understanding Hints

As an application designer, you may know information about your data that the optimizer cannot. For example, you may know that a certain index is more selective for certain queries than the optimizer can determine. Based on this information, you may be able to choose a more efficient execution plan than the optimizer can. In such a case, you can use hints to force the optimizer to use your chosen execution plan.

You can use hints to specify:

- the optimization approach for a SQL statement.
- the goal of the cost-based approach for a SQL statement.
- the access path for a table accessed by the statement.
- the join order for a join statement.

- a join operation in a join statement.

Specifying Hints

Hints apply only to the optimization of the statement block in which they appear. A statement block is any one of the following statements or parts of statements:

- a simple SELECT, UPDATE, or DELETE statement
- a parent statement or subquery of a complex statement
- a part of a compound query

You can send hints for a SQL statement to the optimizer by enclosing them in a Comment within the statement.

A statement can have only one Comment containing hints. This Comment can only follow the SELECT, UPDATE, or DELETE keyword.

If you specify hints incorrectly, Oracle ignores them but does not return an error:

- Oracle ignores hints if the Comment containing them does not follow a DELETE, SELECT, or UPDATE keyword.
- Oracle ignores hints containing syntax errors, but considers other correctly specified hints within the same Comment.
- Oracle ignores combinations of conflicting hints, but considers other hints within the same Comment.

Oracle also ignores hints in all SQL statements in environments which use PL/SQL Version 1.

The optimizer only recognizes hints when using the cost-based approach. If you include any hint (except the RULE hint) in a statement block, the optimizer automatically uses the cost-based approach.

For more information about comments and hints, see the Oracle SQL Analyze online help, and *Oracle Server Tuning*.

The following hints, organized according to the optimization area they impact, can be added to your SQL statement manually or using the Hint Wizard.

Note that the availability of some hints is limited by the database version.

Optimization Approaches

ALL_ROWS

CHOOSE

FIRST RULES

RULE

Parallel Execution

APPEND*ORDERED

STAR**

STAR_TRANSFORMATION*

Join Operations

DRIVING_SITE*

USE_HASH**

USE_MERGE

USE_NL

Additional Hints

CACHE

NOCACHE

PUSH_SUBQ

MERGE***

NO_MERGE*

PUSH_JOIN_PRED***

NO_PUSH_JOIN_PRED***

ORDERED PREDICATES***

Access Methods

AND_EQUAL

CLUSTER

FULL

HASH

HASH_AJ

HASH_SJ ***

INDEX

INDEX_ASC

INDEX_COMBINE*

INDEX_DESC

INDEX_FFS*

MERGE_AJ**

MERGE_SJ***

ROW_ID

USE_CONCAT

NO_EXPAND***

REWRITE***

NOREWRITE***

Join Orders

NOAPPEND*

NOPARALLEL

PARALLEL

PARALLEL_INDEX*

NO_PARALLEL_INDEX***

* Available only for Oracle8 databases

**Available only for Oracle7.3 and Oracle8 databases

***Available only for Oracle8*i* databases

Understanding Rules-of-Thumb

The syntax of your SQL statements has a significant affect on performance. The use of certain command clauses can disable indexes or cause inefficient data sorting and filtering. In some cases, the order in which command clauses are used or data and tables are referenced can add an extra burden on resources.

Oracle SQL Analyze contains a set of rules, developed by database experts, that evaluates SQL statements and suggests alternative statements, when possible. These rules focus on principles of optimizing performance, such as:

- enabling indexes to eliminate the need for full table scans
- reducing the number of sorts, mergers, and filtering operations required
- reducing the number of rows that need to be sorted, filtered, or merged

Oracle SQL Analyze applies these “rules-of-thumb” when you tune your statement with the Tuning Wizard, and supplies alternative SQL statements when possible. Oracle SQL Analyze checks your statement against the following rules, which are explained in this section:

- Use NOT EXISTS instead of NOT IN
- Use NOT EXISTS or NOT IN with hints instead of MINUS
- Use TRUNC differently to enable indexes
- Use operators differently to enable indexes
- Do not use columns on both sides of operator
- Use WHERE in place of HAVING
- Use UNION ALL instead of UNION

Use NOT EXISTS instead of NOT IN

Using NOT EXISTS instead of NOT IN adds a limiting condition to your queries that can reduce the number of full table scans necessary.

The following example uses a NOT IN clause to find names and department IDs in the DEPARTMENT table where the department ID does not also exist in the EMPLOYEE table:

```
SELECT name, department_id
FROM department
WHERE department_id NOT IN
(SELECT department_id FROM employee)
```

Because NOT IN does not use a limiting condition, Oracle will perform a full table scan of DEPARTMENT. For each record in DEPARTMENT, the subquery will be executed. Since the subquery has no limiting WHERE clause, it will perform a full table scan for every record in the full table scan of DEPARTMENT.

Instead, NOT EXISTS can be used so that nested index scans will be used in the subquery for each row in the DEPARTMENT table. The logic of the NOT EXIST clause tells Oracle not to return the row if it finds a match in both tables. The only records that will be returned from DEPARTMENT are those that return no rows from the subquery, and no full table scans are performed by the subquery. The following statement, therefore, is more efficient than the previous example.

```
SELECT name, department_id
FROM department,
WHERE NOT EXISTS
(SELECT department_id
FROM employee
WHERE department.department_id=employee.department_id)
```

Use NOT EXISTS or NOT IN with hints instead of MINUS

MINUS returns the set of rows from one query that is not present in the set of rows returned by a second query. Rewriting queries using NOT EXISTS or NOT IN can enable them to take advantage of indexes, reducing the number of full table scans a clause may require.

In some cases, Oracle SQL Analyze might determine that because a hash anti-join (HASH_AJ) usually does not require a sort, it will produce better results than MINUS.

The following query, for example, matches names and birthdates in the EMPLOYEE table with those in the STOCKHOLDER table, then returns the names and birthdates of employees who are not stockholders. Because MINUS does not use indexes, Oracle will use two full table scans and perform a sort on each table before the MINUS operation can be performed.

```
SELECT birth_date, last_name, first_name
FROM employee
MINUS
SELECT birth_date, last_name, first_name
FROM stock_holder
```

If the statement is re-written using NOT EXISTS, Oracle can use nested index scans in the subquery for rows in the primary statement.

```
SELECT birth_date, last_name, first_name
FROM employee
WHERE NOT EXISTS
(SELECT 1
FROM stock_holder
WHERE stock_holder.birth_date = employee.birth_date
AND stock_holder.first_name = employee.first_name)
```

If Oracle SQL Analyze determines that a hash anti-join will produce better results, the example query could be rewritten to use two full table scans and an anti-join algorithm to join the rows, instead of performing sort and minus operations.

```
SELECT birth_date, last_name, first_name
FROM employee
WHERE (birth_date, last_name, first_name)NOT IN
(SELECT /*+ hash_aj (stock_holder) */ birth_date, last_name, first_name
FROM stock_holder)
```

Use TRUNC differently to enable indexes

Using the truncate command (TRUNC) on an indexed column disables the index. Rewriting your query so that fewer columns are truncated allows it to take advantage of indexes to increase performance.

In the following example, trans_date is an indexed column, but the index is disabled by the TRUNC command.

```
SELECT account_name, trans_date
FROM transaction
WHERE TRUNC(trans_date) = TRUNC(sysdate)
```

The query can be rewritten as shown below to use the trans_date index and increase performance.

```
SELECT account_name, trans_date
FROM transaction
WHERE trans_date BETWEEN TRUNC(sysdate) AND TRUNC(sysdate) + .99999
```

Use operators differently to enable indexes

The optimizer does not use an index if the indexed column is part of a function (in the WHERE clause). If Oracle SQL Analyze determines that an equation can be rewritten to avoid the use of operators, it can rewrite the statement as shown below.

In this example, the equation in the query can be rewritten as a simple inequality clause. statement. Therefore the query

```
SELECT account_name, trans_date, amount
FROM transaction
WHERE amount + 3000 < 5000
```

can be rewritten as

```
SELECT account_name, trans_date, amount
FROM transaction
WHERE amount < 2000
```

Do not use columns on both sides of operator

When an indexed column appears on both sides of an operator, the index for that column is disabled. Oracle SQL Analyze detects this condition and, when possible, rewrites the statement to allow the index to be used.

In the following example, the column account_name is indexed, but the index is disabled.

```
SELECT account_name, trans_date, amount
FROM transaction
WHERE account_name = NVL(:acc_name, account_name)
```

The query can be rewritten using LIKE so that the indexed column is only on one side of the operator.

```
SELECT account_name, trans_date, amount
FROM transaction
WHERE account_name LIKE NVL(:acc_name, '%')
```

Use WHERE in place of HAVING

The HAVING clause limits rows collected by the GROUP BY clause only after they have been aggregated. Whenever possible, it is better to limit the number of rows retrieved before they are merged and sorted into an aggregation. Using WHERE in place of HAVING eliminates rows before they are added to the aggregation.

The statement below sorts an entire list of items by quantity, then removes from the aggregation all items with a quantity less than 40.

```
SELECT quantity, AVG(actual_price)
FROM item
GROUP BY quantity
HAVING quantity > 40
```

The statement can be rewritten so that all rows where QUANTITY is less than 40 are removed before the aggregation is sorted.

```
SELECT quantity, AVG(actual_price)
FROM item
WHERE quantity >40
GROUP BY quantity
```

Note that if the HAVING clause is applied to aggregate functions, it cannot be replaced by WHERE. In the query below, for example, HAVING is applied to a SUM function.

```
SELECT program_name
   ,count
   ,min(end_date-start_date) "Min Runtime"
   ,avg(end_date-start_date)"Avg Runtime"
   ,max((end_date-start_date)"Max Runtime"
   ,sum(end_date-start_date)"tot Runtime"
FROM jobs
WHERE start_date>sys_date - 7
GROUP BY program_name
HAVING sum((end_date-start_date)>0.25 or max(end_date-start_date) > 0.04
```

Use UNION ALL instead of UNION

The difference between the UNION and UNION ALL is that UNION requires a sort operation to eliminate any rows that are duplicated across the two row sets, while UNION ALL returns all rows, even if they are duplicated. If duplicated rows are not important, using UNION ALL can avoid potentially expensive sorts, mergers, and filtering operations.

For example, the statement

```
SELECT acct_num, balance_amt
FROM debit_transactions
WHERE tran_date = '31-DEC-96'
UNION
SELECT acct_num, balance_amt
FROM credit_transactions
WHERE tran_date = '31-DEC-96'
```

Can be rewritten as

```
SELECT acct_num, balance_amt
FROM debit_transactions
WHERE tran_date = '31-DEC-96'
UNION ALL
SELECT acct_num, balance_amt
FROM credit_transactions
WHERE tran_date = '31-DEC-96'
```

Understanding Join Methodology

Most SQL queries involve selecting data from multiple tables. In these operations, data from several tables is merged, or "joined" from multiple tables in order to produce the desired results set. The type of join method used, and the order of table joining, is driven by several factors, such as the presence of indexes and the cardinality of columns involved in the selection process. If indexes are present on one or more of the tables involved in the query, then specific rows can be selected from each table and these sets of rows can then be compared and joined.

Given that the data obtained from one table is used as criteria for selecting data from another table, the order of table access operations is the major determinant of performance for a multi-table query. Determining the proper join order becomes more complex as the number of joined tables increases. The potential join orders increases by n! for n tables; a join involving six tables has 720 possible join orders.

The cost- and rule-based optimizers use specific rules, data, and strategies to determine the join order. As an informed developer, you can influence the behavior of both optimizers.

You can positively effect the performance of a multi-table join by determining the desired join order and listing the tables in reversed join order (last to first) in the select statement. In this case, where there are equal rule conditions for objects, the table order will determine the join order.

You can assist the cost-based optimizer and control the execution plan through the use of SQL hints. This can be valuable for specific queries where you are aware of details that may not be available to the optimizer, such as:

- the performance goal of a particular query (throughput or response time)
- outdated or non-existing statistics for certain objects
- bind variables that may affect a filter condition

Using hints to control the join method and order for specific queries can be complex and risky. To assist you with this, the Oracle SQL Analyze Tuning Wizard provides an automated methodology that can be used to evaluate alternative join strategies where appropriate. In the same manner as Oracle's internal cost-based optimizer, Oracle SQL Analyze estimates the cost for executing a statement in different ways:

- the cost of an access method is directly proportional to the number of blocks accessed by that method
- the cost of a join method depends on the number of blocks accessed by the join method and which table "drives" the join (the table with the lowest expected row count)

If an alternative NESTED LOOPS join order can be used, Oracle SQL Analyze will rewrite the statement with the necessary hints or reordering of objects. You may want to further analyze your statement to see if other join methods can be added to improve performance.

For example, the following decision-support query has a long execution time:

```
SELECT a.name, b.description, c.count, d.name, e.name
FROM a, b, c, d, e, f, g
WHERE a.id = :id
     AND a.id = b.id
     AND a.date BETWEEN :lo_date and :hi_date
     AND b.id = c.id
     AND c.val = DECODE(:frequency, 'COMMON', 0, 'UNKNOWN', g.unknown_val,
         'RARE', 2)
     AND c.id = d.id
     AND d.id = e.id
     AND d.in_stock < 1000
     AND e.id = f.id
     AND e.subid = f.subid(+)
     AND f.date BETWEEN :lo_date AND :hi_date
     AND f.id = g.id
```

The above query has 7!, or 5040 possible join orders. If the statistics in the data dictionary are out of date or missing for certain objects, the optimizer's efforts to compute the join order will be severely hampered. Even if the correct statistics are available, the BETWEEN filter conditions must be guessed by the optimizer.

Based on the selectivity of the filter conditions and estimated statistics, Oracle SQL Analyze can estimate the cost of different join strategies, and determine the join order g, c, d, a, b, e, f and join methods of nested loops joins for all tables driven to except f, which requires a hash join and access methods of unique index lookups for all tables except g and d, which need full table scans. If this statement was tuned with the Tuning Wizard, Oracle SQL Analyze would modify the object order and add NESTED LOOPS hints to implement those parts of this analysis.

For more information about join methodology, see *Oracle Server Concepts*.

Using the Tuning Wizard

The Tuning Wizard guides you through the entire SQL statement tuning process. It evaluates your SQL Statement using Rule-of-Thumb and Join Methodology rules and generates alternate, optimized versions of your SQL statement.

To use the Tuning Wizard:

Select **Tools=>Tuning Wizard**. The Tuning Wizard will guide you through the rest of this process.

1. Describe your SQL environment.
2. Gather statistics for tuning.
3. Apply SQL tuning rules-of-thumb to the SQL statement.
4. Add SQL hints to the SQL statement to optimize join strategies.

Figure 6–1 Tuning Wizard Opening Screen

The Tuning Wizard Process

The Tuning Wizard is an automated guide that leads you through a process for tuning a SQL statement. Throughout the process, you will be able to make choices that will help the wizard optimize your specific SQL statement. If you need more information to make your choices, select the **Help** button.

The Tuning Wizard will guide you through the following steps:

1. Describe your SQL environment.
   - Describe the tuning environment (OLTP or DSS).
   - Set the initialization parameters.
2. Gather statistics for tuning.
   - Set the method for object statistic collection.
3. Apply SQL tuning rules-of-thumb to the SQL statement.
4. Apply rules-of-thumb.
   - Add SQL hints to the SQL statement to optimize join strategies.
   - Choose to optimize your join strategies.
   - Assign values to bind variables.
5. Apply join methodology.
6. Compare original and modified statements.

One of the most powerful features of the Tuning Wizard is its ability to compare the original and modified statements. You can modify statements and know almost instantly whether or not your changes have improved statement performance.

Using the Hint Wizard

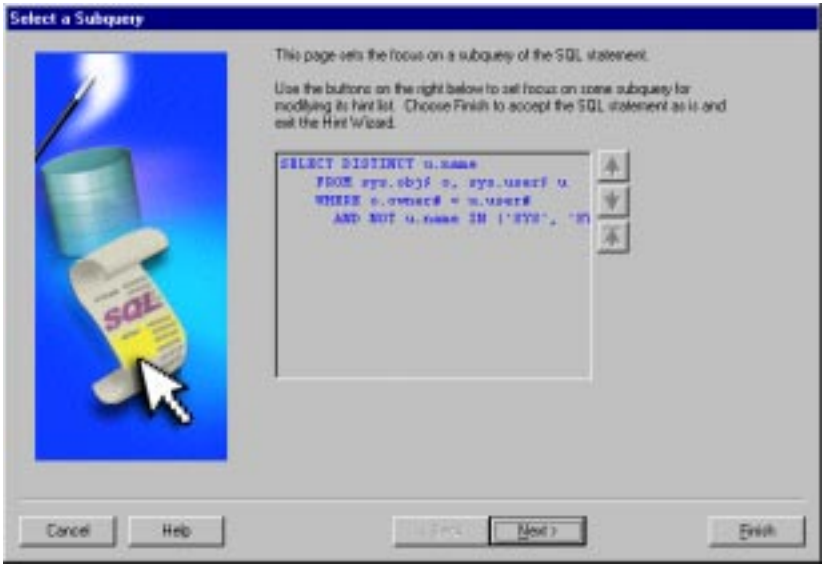
The Hint Wizard identifies hints in a statement and allows the user to present other hints that can be added to the statement. It provides a description for a selected hint and will automatically generate a new SQL statement if a hint is added or deleted.

To use the Hint Wizard:

Select **Tools=>Hint Wizard**. The Hint Wizard will guide you through the rest of this process.

1. Select a subquery to analyze from the Hint Wizard page.
2. View / delete the current hints.
3. Select a new hint to add, and:
   - supply table parameters, if necessary.
   - index parameters, if necessary.
4. Review the current hints.
5. Apply hints to the SQL statement.

Figure 6–2 Hint Wizard Opening Screen



7

Verifying Performance

The purpose of this brief chapter is to remind you of the importance of verifying that your tuning efforts have indeed improved the performance of your statements, and to point out the ways you can use Oracle SQL Analyze to do it.

How to Verify SQL Performance Improvement

You can use the same methods you used for gathering information to verify that the performance of your statement has been improved:

- Execute the new statements and compare the results, as shown in Chapter 3 on page 3-34.
- Generate new explain plans and compare them as described in Chapter 3 on page 3-34.
- Review the object details to ensure they're being used efficiently, as described in Chapter 3 on page 3-12.

Part IV

Getting Started with Oracle Expert

Oracle Expert is an Oracle Enterprise Manager integrated application used to automate performance tuning. It complements and automates many of the elements of a good structured tuning methodology such as the process of collecting and analyzing performance tuning data, and provides expert database tuning recommendations. Additionally, Oracle Expert generates scripts that assist with the implementation of tuning recommendations.

This part contains the following chapters:

8

Introduction to Oracle Expert

Oracle Expert is a software tool you use to optimize the performance of your database environment. Oracle Expert assists with the initial configuration of the database and with the collection and evaluation of the performance characteristics of existing databases.

Oracle Expert automates the process of collecting and analyzing performance tuning data, and provides expert database tuning recommendations. Additionally, Oracle Expert generates scripts that assist with the implementation of tuning recommendations.

The topics in this chapter include:

Advantages of Using Oracle Expert

Oracle Expert provides many advantages. Oracle Expert:

- uses trends to ensure continued performance as the database environment changes over time
- provides consistent and complete recommendations
- performs interdependency checking during analysis
- detects and reports symptoms of poor performance

- sifts through vast quantities of data quickly to identify performance problems
- identifies situations where advanced performance features of the database can be applied

Oracle Expert also serves as:

- an educational tool that explains all of its tuning recommendations through detailed reports
- a methodology tool that guides the DBA, analyst, and designer through the process of improving the performance of an Oracle database
- a maintenance tool that uses historical trends to inform the DBA of degrading performance and impending bottlenecks
- an information tool that collects tuning data from numerous sources. This data not only feeds the tuning process, but can be displayed, edited, and reported on.
- an automation tool that assists the DBA by automating routine database maintenance and tuning tasks

What Is Database Tuning?

The database tuning process consists of tasks such as:

- balancing the different types of competing resources in the database environment so that the most important applications have priority access to the resources they need to run
- identifying resource bottlenecks and eliminating them
- optimizing the use of existing resources within the database environment
- taking advantage of database features for the types of work performed on the database

Database Tuning Issues

Even though you may realize you have a poorly tuned database environment, you may not have the luxury to resolve the problem, because:

- Tuning the database environment may take more time than you can afford to spend on the task.
- Management may not want to disrupt business operations.

- You may not want to risk introducing further performance degradation.
- The complexity of the problem may be beyond the skill level of the person assigned the problem.
- The tools may not be available to identify the cause of the problem.

Resolving Tuning Issues

To resolve tuning issues, there are at least two important requirements:

- extensive tool expertise
- consistency in tuning

A significant amount of a database expert's time is spent collecting and sifting through vast quantities of information. Collecting information for a normal database tuning session requires knowledge of many tools.

Also, the effectiveness of the database tuning effort can vary a great deal depending on the expertise of the person doing the job. To further complicate the issue, with database tuning there often is no exact solution to a specific performance problem.

The tuning recommendations produced by Oracle Expert are both consistent and accurate. Oracle Expert can sift through volumes of tuning information without missing relevant symptoms. It automates many of the repetitive and time-consuming aspects of database tuning, thus reducing the time required to get meaningful performance improvements. Finally, Oracle Expert manages the history of the collected information over time.

What Are the Types of Performance Tuning?

Tuning an Oracle database can involve tuning the application, the instance, and the space usage in a database.

Whether you are writing new SQL statements or tuning problematic statements in an existing application, application tuning can improve CPU response time, reduce disk I/O resources, and reduce memory resources. The methodology for tuning SQL involves identifying the statements that consume the most resources and then tuning these statements to use fewer resources. In general, a small number of SQL statements are responsible for most of the activity that occurs in the database. Rather than trying to completely understand an application, focus your tuning efforts on those statements or tables where the benefit of tuning will exceed the cost of tuning.

Approaches to SQL statement tuning include: determining which optimal indexes should exist on a table, and identifying existing indexes that should be rebuilt to improve performance. You should also ensure SQL is shared effectively. Ineffective SQL sharing can result in unnecessary reparsing which requires more CPU usage.

Instance tuning can be used to solve a variety of problems such as inefficient memory allocation and I/O problems. Instance tuning involves tuning areas such as the redo log buffer, the shared pool, the buffer cache, and the sort areas. Instance tuning also tunes the log writer (LGWR) and database writer (DBWR) background processes.

Effective space usage and management will improve database availability and reduce performance problems resulting from poor space utilization. When an object such as a table is created, space in the database is allocated for the data. Therefore, correct placement and sizing of these objects is essential.

Through the available tuning scopes, Oracle Expert supports the above tuning areas by checking for the following:

Type of Tuning	Tuning Scope
Application Tuning	SQL Reuse Opportunities
	Optimal Data Access
Instance Tuning	Instance Optimizations
Space Management	Appropriate Space Management

Ways to Use Oracle Expert

Oracle Expert is as flexible as you are. Oracle Expert can help you achieve any and all of the following:

- comprehensive tuning

  Deals with the optimization of all aspects of the database environment and maintains the performance of the database over time.

- focused tuning

  Delves into the resolution of a known performance problem. This resolution occurs as you choose the appropriate tuning categories and focus on the particular problem.

- initial configuration

Oracle Expert can optimize a newly created database using additional information supplied about the workload, physical memory, and expected transaction volumes.

Sample Tuning Session

Oracle Expert provides you with a sample tuning session called “Personnel session.” Personnel session is a tuning session against a non-existent Personnel database. It contains example data used by Oracle Expert, such as database, instance, schema, environment, and SQL workload information.

Note: The sample tuning session does not use a real instance. Therefore performing a collection from the sample simply reloads the data from the sample file.

With this sample tuning session, you can experiment with the View / Edit page, analysis, reviewing recommendations, and generating script files.

To load this sample, choose Help=>Load Sample from the Oracle Expert menu bar. The Personnel tuning session displays in the tree list.

If you want to experiment with collecting data for this tuning session, you must use the XPPSO.XDL file in the $ORACLE_HOME\SYSMAN\EXPERT\SAMPLE directory.

When setting collect options for database, instance, schema, or workload, choose to collect from File, selecting the $ORACLE_HOME\SYSMAN\EXPERT\SAMPLE\XPPSO.XDL file as the source. This file contains the information needed for all the collection classes.

9

Oracle Expert Methodology

Oracle Expert performs tuning by following a designed methodology.

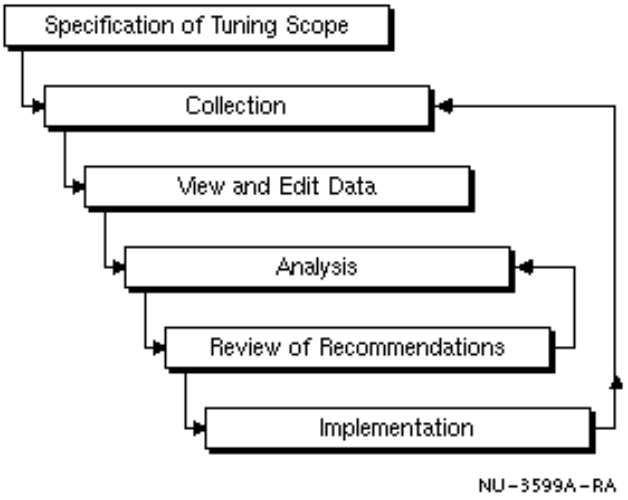
The topics in this chapter include:

- Steps in Oracle Expert Methodology
- Setting the Scope of the Tuning Session
- Collecting Data
- Managing SQL History Data
- Viewing and Editing Collected Data
- Generating Recommendations
- Reviewing Recommendations
- Implementing Recommendations
- Inputs and Outputs

Steps in Oracle Expert Methodology

The Oracle Expert methodology includes the following steps (see Figure 9–1).

Figure 9–1 Implementation of Oracle Expert Methodology



1. setting the scope of the tuning session
2. collecting the data
3. viewing and editing the collected data
4. analyzing the data and generating recommendations
5. reviewing the Oracle Expert recommendations
6. generating scripts for implementing the recommendations

Setting the Scope of the Tuning Session

When you set the scope of a new tuning session, you are telling Oracle Expert exactly what you want to tune. Your choices can be one or more of the following:

- Check for Instance Optimizations

  Use this option to confirm that your instance parameter settings are appropriate and contention is not a problem.

- Check for SQL Reuse Opportunities

  Use this option to confirm SQL is being shared as appropriate.

- Check for Appropriate Space Management

Use this option to evaluate database space management issues such as sizing and placement.

- Check for Optimal Data Access

  Use this option to optimize index usage on your database tables or to check for indexes that should be rebuilt.

Collecting Data

Oracle Expert collects the following classes of data as appropriate for the specified tuning scopes:

- database (database name and version, database users, tablespaces, and public synonyms)
- instances (parameters and statistics)
- schemas (tables, constraints, indexes, clusters, views, and synonyms)
- environment (system information)
- workload (can include data collected by Oracle Trace, a SQL History, an .XDL file or SQL cache)

Managing SQL History Data

Oracle Expert allows you to maintain a SQL History for your service. The SQL History can store SQL cache, Oracle Trace, or .XDL file data. The SQL History is shared among other Oracle Tuning Pack applications and tuning sessions. With the SQL History, the user can build up the complete set of SQL statements that are executed within the database environment and share the statements with multiple tuning session. By sharing the SQL history, you do not have to recollect statements for each tuning session. You can merge to the SQL history (building a complete set of SQL statements executing in the database environment) or you can replace an existing SQL History with a new one. Each service, however, can only have one SQL History at a time.

Viewing and Editing Collected Data

Once you have collected the various pieces of tuning data, you can view and edit that data. The data is organized as follows:

- database

  - instances (instance parameters)
  - schemas (tables, views, indexes, clusters, constraints, and synonyms)
  - tablespaces (datafiles and tablespace segments)
  - public synonyms
  - database users
- environment (system information)
- workload (application and request)

You have the option of editing both the attributes and the rules for this data. Attribute information is the actual data collected by the product. This data can be can be edited for "what if" tuning. You should change any dependent attributes for the evaluation to provide the best recommendations. The rules can also be changed. Rule adjustments allow you to influence Oracle Expert's evaluation process.

Generating Recommendations

Once you have collected and edited the data as needed, you can have Oracle Expert perform the analysis to generate tuning recommendations.

During the analysis, Oracle Expert evaluates the collected data in conjunction with all its rules and provides optimal performance recommendations.

Reviewing Recommendations

Once Oracle Expert has analyzed the data, you can review the recommendations and decide which to accept.

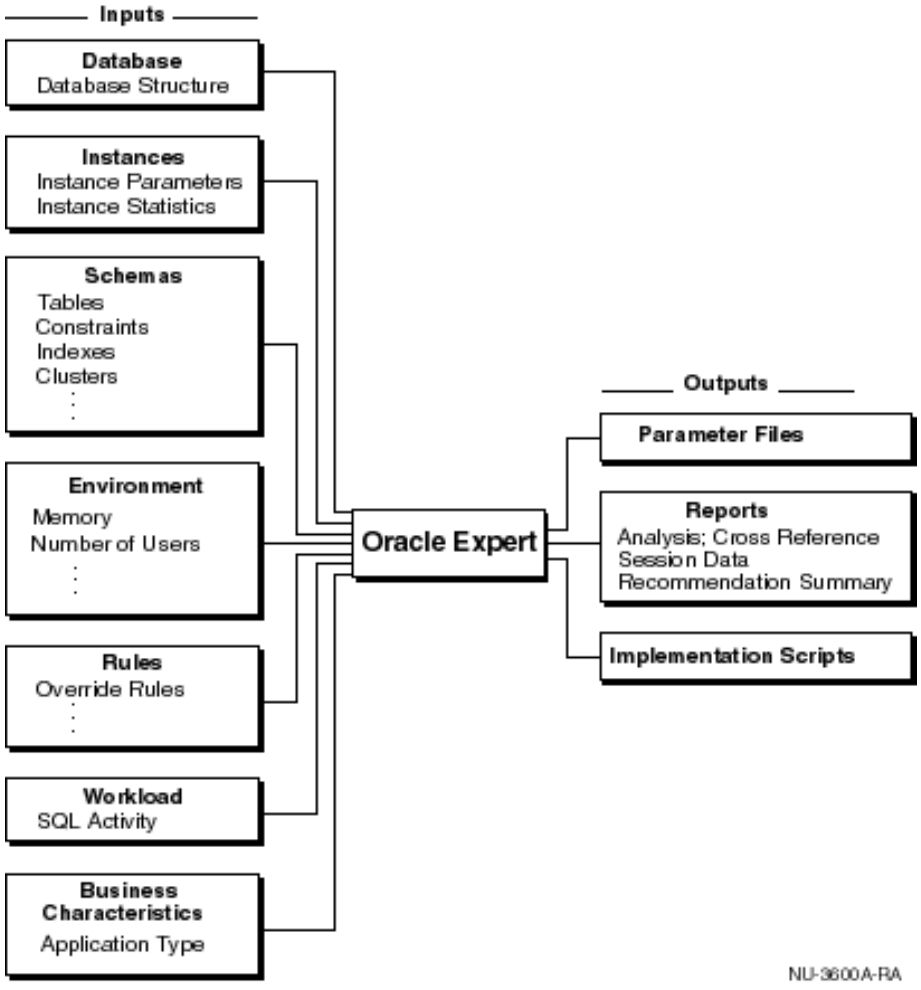
For example, assume that Oracle Expert recommends that you increase the shared_pool_size parameter from 300,000 bytes to 500,000 bytes. If you choose not to use this recommendation, you can decline the recommendation and analyze the data again. Oracle Expert keeps track of the recommendations you have accepted and takes into account interdependencies among the collected data before generating new recommendations.

Implementing Recommendations

When you are ready to implement the Oracle Expert recommendations, you can use Oracle Expert to create parameter files and implementation scripts. These files and scripts allow you to implement the Oracle Expert recommendations at your convenience. These files and scripts minimize the risk of introducing any new problems during implementation, and they reduce the level of expertise required to implement the recommendations.

Inputs and Outputs

Oracle Expert takes collected input, processes this input through various rules and algorithms, and creates recommendations, tuning scripts, and reports.



Tuning Inputs

Oracle Expert uses the following input data to generate effective tuning recommendations:

- Database class

  The Database class data contains database-wide attributes of the database, such as the database name and version, users, tablespaces, and public synonyms.

- Instance class

  The Instance class data refers to the instance parameters and instance statistics information collected by Oracle Expert not necessarily collected from V$ tables.

- Schema class

  The Schema class data refers to the tables, indexes, clusters, views, and constraints information collected by Oracle Expert.

- Environment class

  The Environment class data refers to the physical hardware resources available to the database; for example, and system data (which includes memory and CPU information).

- Workload class

  Workload class data refers to the nature, frequency, and importance of SQL requests that access the database.

- Rules

  Rules are pieces of knowledge used by Oracle Expert to analyze collected data for a tuning session. By changing rule values, you influence the tuning recommendations made by Oracle Expert.

- Business characteristics

  Business characteristics provide Oracle Expert with a higher level of guidance about how the database is used. This is data that cannot be collected directly from the database. For example, the Oracle Server has specific features that are designed to optimize performance in a data warehouse environment. If you set Application Type to Data Warehousing, Oracle Expert knows that these features are applicable. Downtime Tolerance allows the user to influence whether the system will bias its recommendations for recovery or performance.

Generated Output

With inputs such as database, instance, schema, environment, workload, and rules, Oracle Expert generates the output needed for solving database performance problems. Oracle Expert generates the following:

- reports

  Oracle Expert generates the Analysis, Session Data, and Recommendation Summary reports.

  - Analysis report

    Describes the tuning recommendations made by Oracle Expert. This report provides a detailed explanation of what Oracle Expert evaluated, how Oracle Expert interpreted the collected data and why, and any risks involved in implementing the recommendations.

  - Session Data report

    Provides summary database information. It also provides detailed information about instances, database users, tablespaces, schemas, environment, workload, and rules.

  - Recommendation Summary

    Describes, in a concise form, the recommendations made by Oracle Expert. This report is a summary of the Analysis report.

  - Workload Cross Reference Report

    Provides information on tables and related requests. The report is organized by table name and by request name so that you can quickly find the information you need. This report can show you if your current Oracle Expert workload is complete.

- implementation files

  In addition to the reports, Oracle Expert generates files to help you implement its recommendations. These files include:

  - implementation script (.TXT file)

    Contains SQL for implementing schema objects, tablespaces and database users. Some recommendations, such as table relocation to a new tablespace, are described textually and performed by the DBA.

  - instance parameter file subsets (.ORA file)

Contains instance parameter values that Oracle Expert recommends for improving the performance of a particular instance. Oracle Expert generates these subsets, which you can merge into the existing INIT.ORA file for the instance.

10

Getting Started with Oracle Expert

This chapter describes how to start Oracle Expert and how to identify the databases to tune.

The topics in this chapter include:

Starting Oracle Expert

Oracle Expert can be started in two ways:

- Through the Oracle Management Server

  When you start Oracle Expert through the Oracle Enterprise Manager console, you are connecting through the Oracle Management Server or *OMS*.

  > **Note:** For general information on the Oracle Management Server, see the *Oracle Enterprise Manager Administrator's Guide* and the *Oracle Enterprise Manager Concepts Guide*. For Management Server setup and configuration, see the *Oracle Enterprise Manager Configuration Guide*.

  You can start Oracle Expert using the Tuning Pack icon on the left side of the console or by choosing Tools=>Tuning Pack=>Oracle Expert from the console menu bar. There is no need to select a database from the console Navigator.

- Standalone (without the Oracle Management Server)

  To start Oracle Expert without going through the Oracle Management Server, you can navigate to the Oracle Tuning Pack program group. The Expert Login dialog box prompts you for the information necessary to connect directly to a repository. (Note that you may need to manually add the nodes in Oracle Expert if the database container is empty.)

  Oracle Expert maintains a repository where the information collected for your database environment and the results of the analysis are stored. This repository is a part of the Oracle Enterprise Manager repository. Use the Expert Login dialog box to provide the login information needed to access the repository.

  Note: If the Oracle Enterprise Manager repository has not been created, Oracle Expert automatically creates the repository (with user confirmation) at connect time. Similarly, if your repository version differs from the version of Oracle Expert and Oracle Enterprise Manager, the repository will be upgraded automatically.

When you start Oracle Expert for the first time by clicking the OK button in the Expert Login dialog box, there will be a pause. This occurs while Oracle Expert is loading its default rules into the repository. The status line informs you that Oracle Expert is loading default rules during this process.

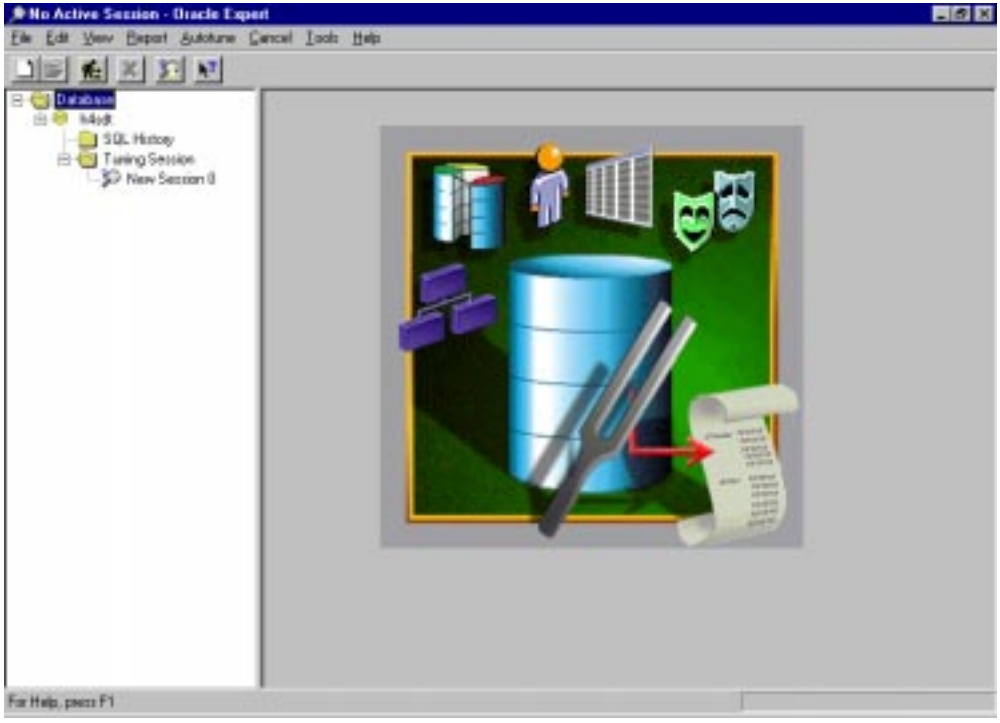
By default, the Tuning Session wizard displays each time you start Oracle Expert. You can disable this automatic display by clicking the appropriate box on the Tuning Session wizard Welcome screen.

You can now begin using Oracle Expert.

Oracle Expert Main Window

When you launch Oracle Expert, the Oracle Expert main window displays, as shown in Figure 10–1. Click F1 from this page for more information.

Figure 10–1 Oracle Expert Main Window



The window consists of menus, panes, a toolbar, and a status bar. Use the menus and toolbar to access Oracle Expert functionality. The status bar provides information relating to the task you are performing.

Identifying the Database to Tune

Before you can create a tuning session, you must identify the database you want Oracle Expert to tune. There are two ways to select a database:

- Click on the name of the database in the tree list

- Right-click on the word 'Databases' in the tree list. Select New from the pop-up menu. A dialog box will appear prompting you for a username, password, service name and node.

The database (service) name is the name Oracle Expert will use for the Database object and the Instance object on the View / Edit page of the tuning session window.

For services added through an Oracle Management Server login, Oracle Expert connects to the database by referring to the information in the Preferred Credentials defined in the Oracle Enterprise Manager console. If those credentials are not defined, Oracle Expert uses the credentials used to connect to the repository.

For services added manually, Oracle Expert uses the username, password, and service name supplied in the dialog box.If the repository credentials are not defined, you are prompted for the username and password.

Note that Oracle Expert requires the SELECT ANY TABLE privilege to be able to access and collect data from the database being tuned.

Creating a SQL History

Before creating a tuning session you should create a SQL History. The SQL History maintains a complete set of application SQL data and statistics that are executed within the database environment. Depending on the tuning scopes you select, you can use an existing SQL History as the source for your workload information.

To create a SQL History:

1. Right-click on the word 'SQL History' in the tree list.
2. Select New from the pop-up menu. The SQL History Options page appears.
3. Select the source of data for your SQL History. The source options are:
   - SQL Cache (all instances)
   - Oracle Trace
   - File (.XDL)

   Note: The typical means of collecting workload information is from the SQL Cache.
4. If a SQL History already exists on your database, you may choose to merge or replace the data in the existing SQL History with your new collection.

5. Click the Collect button.
6. Click the View / Edit tab to make changes to the workload request data.

The SQL History you create can be shared by multiple Oracle Tuning Pack applications. Sharing the same information across applications ensures that all SQL statements executing within the database environment are taken into consideration during an evaluation.

> **Note:** Once the SQL History is created, you can right-click on the SQL History name in the tree list for a list of options such as open, export, close, delete and rename. Before making changes, however, be sure that no other applications are accessing the SQL History.

11

Creating and Working with Tuning Sessions

This chapter describes how to create and work with tuning sessions after you have started Oracle Expert.

The topics in this chapter include:

Creating a Tuning Session

A tuning session is the framework within which Oracle Expert performs its tuning activities. Tuning sessions allow you to organize tuning activity and separate sessions within the same database. Oracle Expert provides the following ways to create a tuning session: manually or using the Tuning Session wizard.

Note: Click Help to get page specific information on any of the Tuning Session Wizard pages.

Creating a Tuning Session Using the Tuning Session Wizard

Oracle Expert offers an easy and quick way to create a new tuning session using the Tuning Session wizard. To activate the Tuning Session wizard, choose Tools=>Tuning Session Wizard.

The Tuning Session wizard automatically displays when you start Oracle Expert, unless you have disabled the automatic display.

Creating a Tuning Session Manually

To create a new tuning session, in the tree list click on the '+' to the left of the name of the database you want to tune, expand the tree list to display the Tuning Session folder, and perform one of the following:

- choose File=>Create
- click the Create Tuning Session button in the Oracle Expert toolbar
- click the right mouse button and choose Create in the pop-up menu

Oracle Expert assigns a unique name to the new tuning session. You can either accept this name or provide a new name. Edit the tuning session name by clicking on the name and typing the new name. The new name must be 40 characters or fewer in length. Oracle Expert preserves the case (lowercase and uppercase) of alphabetic characters in the name.

If you receive a "table not found" error message, you may not have adequate privileges. Oracle Expert requires SELECT ANY TABLE privilege to collect the required information from the database to be tuned.

When logging in to the target database, Oracle Expert uses the preferences provided by the console. If you manually created a service, the username and password supplied are used. The account specified must have the SELECT ANY TABLE privilege.

Note: Be sure to give your tuning sessions descriptive names so that you can identify the type of tuning activity contained in the session.

Setting the Scope of a Tuning Session

You must specify the scope of the tuning effort for each Oracle Expert tuning session. You do this by selecting the tuning scope or scopes you want Oracle Expert to address for the session. The tuning scopes you select determine the kinds of data that will be collected and the types of tuning recommendations Oracle Expert will make for the tuning session.

You can select one or more of the following tuning scopes:

- Check for Instance Optimizations
- Check for SQL Reuse Opportunities
- Check for Appropriate Space Management
- Check for Optimal Data Access

You can select any combination of tuning scopes. They are not mutually exclusive. If you select all the categories for all the tuning scopes, this is called comprehensive tuning. During a comprehensive tuning session, Oracle Expert generates every tuning recommendation it is capable of making for a database. Comprehensive tuning is resource intensive and it generally takes a considerable amount of time to get results.

If you do not select all the tuning scopes, this is called focused tuning. During a focused tuning session, Oracle Expert generates tuning recommendations for the selected tuning scopes.

Select the Scope tab to display the Scope page of the Oracle Expert tuning session window, then select one or more tuning scopes for the tuning session.

Table 11–1 uses a set of tuning considerations to compare the three types of tuning. Use the table to determine if the type of tuning you want to do is practical (based on the time and resources you have to devote to the tuning session). Note that Oracle Expert allows you to perform as many or as few categories of tuning as you want at a given time. If you do not have the time or resources to tune all the categories you are interested in at once, you can tune those you have time for first, then tune the other categories when you have more time or resources.

Table 11–1 Comparison of Tuning Scopes

Tuning Considerations	Instance Optimization	Space Management	SQL Reuse	Data Access by Table	Data Access by Workload	Data Access Index Rebuild
Amount of data to collect	small	med to large	small to med	small to large	small to large	small
Time Required to collect data	short	med to long	short to med	short to long	short to long	short
Impact of collection effort on database	negligible	low	low	low	low	low
Manual effort required to collect of edit data	low	low	low	low to high	low	low

Table 11–1 Comparison of Tuning Scopes

Tuning Considerations	Instance Optimization	Space Management	SQL Reuse	Data Access by Table	Data Access by Workload	Data Access Index Rebuild
Potential complexity of implementing tuning recommendations	low	low	low	low	low	low
Potential gain from implementing tuning recommendations	med	med	med	med to high	med to high	med

You can change the scope of a tuning session. For more information about changing the scope, see "Modifying a Tuning Session" on page 11-7.

Instance Optimizations

When *Check for Instance Optimizations* is selected, you can tune instance parameters, which control the behavior of the database and certain configuration options, such as how the database will use memory resources on the system as well as various contention problems. Oracle Expert tunes the following categories, assuming that you collect the data Oracle Expert expects:

- SGA parameters

  These parameters affect the total size of the instance's System Global Area (SGA). The appropriate setting of these parameters results in efficient utilization of memory and prevents reparsing SQL statements except when necessary. Examples of these parameters include the db_block_buffers and shared_pool_reserved_size parameters.

- I/O parameters

  These parameters affect the throughput or distribution of I/O for the instance. Examples of these parameters include the checkpoint_process and db_file_multiblock_read_count parameters.

- Sort parameters

These parameters influence how the Oracle Server performs sort operations on behalf of the user. Examples of these parameters include the sort_direct_write and sort_area_retained_size parameters.

- Parallel Query parameters

  These parameters are specific to the parallel query behavior for the instance. Examples of these parameters include the parallel_min_servers and parallel_max_servers parameters.

- Oracle Parallel Server parameters

  These parameters are specific to the Oracle Parallel Server environment. These parameters are the gc_db_locks, gc_files_to_locks, and gc_releasable_locks parameters.

- Operating System-Specific (OS-Specific) parameters

  These parameters are specific to the instance parameters of the operating system and their availability varies from platform to platform. These parameters can have a significant impact on performance. Examples of these parameters include async_write and db_writers parameters.

- Contention Problems

SQL Reuse Opportunities

When *Check for SQL Reuse Opportunities* is selected, Oracle Expert performs SQL statement matching when it analyzes the collected data.

During SQL reuse, Oracle Expert identifies similar statements that prevent SQL statements from being reused in the shared pool because of differences in case and/or spacing.

With SQL statement matching, Oracle Expert compares statements in the workload to determine if similar statements can be rewritten to eliminate redundancy. The Oracle Server maintains only one copy of a distinct SQL statement within the cache to maximize memory and minimize redundant parsing and validating. The Oracle Server does not consider statements to be identical unless they use identical spacing, punctuation, and case, and they match character by character. If Oracle Expert finds one or more statements that can be rewritten to increase cache efficiency, it makes these recommendations.

Space Management

When *Check for Appropriate Space Management* is selected, Oracle Expert evaluates the types and structures of tablespaces, the sizing and placing of schema objects, and the tablespace assignments of database users. If Oracle Expert determines that various guidelines are not being followed, then appropriate recommendations will be made.

Optimal Data Access

When *Check for Optimal Data Access* is selected, Oracle Expert ensures there is efficient access to data.

An access method is a strategy used to retrieve data from a database in the most optimal way. **The Oracle Expert access method tuning rules address the Oracle cost-based optimizer, not the rules-based optimizer.** One method that databases use to decrease data retrieval time is indexes. Oracle Expert offers three types of access method tuning:

- When performing comprehensive index evaluation on tables referenced by worst performing SQL statements, Oracle Expert will automatically focus the Data Access tuning on the tables referenced within the worst performing SQL statements identified in you tuning session workload. The tuning session's SQL statements will be ranked based on each statement's ratio of physical reads per executions. Oracle Expert will also automatically check existing indexes on the target tables for index fragmentation.
- When performing *comprehensive index evaluation on tables you specify,* Oracle Expert uses base index and workload analysis to determine whether new indexes should be created or existing indexes should be modified to enhance performance. Oracle Expert will also automatically check existing indexes on the target tables for index fragmentation.
  - With base index tuning, Oracle Expert scans schemas selected for tuning for evidence of implicit search operations, such as constraints and views. If a constraint or view is found, Oracle Expert determines whether an index is necessary to improve performance when the constraint or view is executed.
  - With workload analysis, Oracle Expert scans SQL statements in the workload. After scanning the workload, Oracle Expert can identify poor existing index structures and recommend index structure changes to improve performance and use of those indexes. Oracle Expert can also recommend that some data be accessed in sorted order (avoiding costly data sorts). Finally, Oracle Expert can recommend new index structures. For

more information about collecting workload data, see "Collecting the Workload Class" on page 12-13. For more information about managing workload data, see "Database Workloads" on page 20-1.

- When performing *index fragmentation evaluation on tables you specify*, Oracle Expert will identify indexes that suffer from index stagnation and should be rebuilt to enhance performance.

Selecting Values for Business Characteristics

Oracle Expert business characteristics are displayed in the Business Characteristics section of the Scope page.

Business Characteristics provide useful information for tuning the database environment. Business characteristics provide information about your database that cannot be collected from your database. For each business characteristic, select the most appropriate value for your database environment. To change the value of a business characteristic, click the arrow next to the current value, then select the new value from the list. Oracle Expert uses the business characteristic values you select to optimize its tuning recommendations for your database's specific environment.

Opening an Existing Tuning Session

You can open an existing tuning session by:

- clicking on the existing tuning session name in the tree list and then choosing File=>Open
- clicking on the existing tuning session name in the tree list and clicking the Open icon on the Oracle Expert toolbar
- double-clicking the session name to make the session the active session
- clicking the Tuning Session wizard button in the toolbar and choosing Open

Modifying a Tuning Session

To change an active session's scope and business characteristic values, use the Scope page of the tuning session window. Enter data in the same manner as you did when you created the tuning session.

You can change the tuning scope to perform a different type of analysis. For example, you might initially have selected and performed focused instance tuning.

After Oracle Expert has analyzed the data collected for the instance tuning session, you might decide to select a different tuning category.

You can also view and edit the rules and attributes associated with the tuning session by using the Edit pull-down menu.

Deleting a Tuning Session

You can delete a tuning session by clicking on the tuning session name in the tree list and choosing File=>Delete. This displays a dialog box that asks you to confirm the tuning session deletion. If you confirm the deletion, Oracle Expert deletes the tuning session and all the data in the repository that is associated with the tuning session.

12

Collecting the Data

This chapter describes how to collect data for a tuning session. The topics discussed in this chapter include:

Overview of Data Collection

The selected tuning categories for a tuning session determine the data that must be collected and stored in the Oracle Expert repository. After the appropriate data is collected, Oracle Expert can analyze (apply its rules to) the collected data and generate tuning recommendations.

The Collect page of the Oracle Expert tuning session window allows you to specify the type of data to collect for a tuning session. The types of data you can select on the Collect page are referred to as collection classes.

Collecting the Collection Classes

The collection classes appear on the Collect page of the tuning session window. They are:

- Database
- Instance
- Schema
- Environment
- Workload

Table 12–1 provides summary information for each of the collection classes, including the size of the class data, its volatility (which governs how frequently the class should be collected), whether Oracle Expert collects the class data automatically, and the source or sources of the data.

Table 12–1 Summary Table for Collection Classes

Collection Class	Size of Class Data	Volatility/ Collection Frequency	Automatic Collection	Sources of Class Data
Database	Variable	Low	Yes	Instance or .XDL file
Instance	Small	High	Yes	Instance or .XDL file
Schema	Variable	Low to Medium[1]	Yes	Instance or .XDL file
Environment	Small	Low to Medium	No	User input or .XDL file
Workload	Variable	Variable	Yes[2]	Instance (SQL cache, Oracle Trace, .XDL file, or SQL History)

[1] Schema Statistics, medium; Schema Data, low

[2] See "Collecting the Workload Class" on page 12-13.

How Oracle Expert Collects Class Data

When you display the Collect page, one or more of the collection classes is enabled and selected. After considering the tuning categories selected for the tuning session, Oracle Expert enables a collection class if data from that class is required. If it can automatically collect the data for a required class from the database being tuned, Oracle Expert selects the class. For more information about the required collection classes for each tuning category, see Table 12–1.

When the Environment class is required for a selected tuning category, Oracle Expert enables the Environment class option on the Collect page.

The Last Collected column on the Collect page shows the date and time that data was last collected for a class or "Never" if it has never been collected.

The Options Set? column indicates whether a class is ready for collection. A green check mark means valid options have been set for that class. A red X means valid options have not been set for the class.

Also, the following scenarios are possible:

- If the entire line is enabled, this indicates that the information is required for the current tuning scope.
- When the entire line is disabled, this indicates that the information is not required for the current tuning scope.
- If the collection class option is enabled, and the collection class box is not checked, the Last Collected column and the Options Set? column will be disabled. This indicates that the class of information will not be collected, but it is available for collection by checking the collection class box.

Specifying the Class Data to Collect

Each collection class has an associated Options button that brings you to an appropriate dialog box. These dialog boxes allow you to choose class options and provide Oracle Expert with the information it needs to collect class data automatically.

The selected tuning categories for a tuning session determine the classes you should collect as well as which class options you should collect. When you have entered data correctly in a Collect Options dialog box, a check mark is displayed at the bottom of the page.

You can start a collection at any time by clicking the Collect button on the Collect page. However, if some of the selected collection classes have a red X in the Options Set? column, Oracle Expert will display a message box stating that invalid options exist and prompt you whether you want to continue with the collection of the classes that have valid option settings. If none of the collection classes have valid option settings, Oracle Expert will display a message stating so and no collection is performed.

Collecting Data Efficiently

You can reduce the time you spend collecting tuning session data by collecting the minimum amount of data Oracle Expert requires to generate tuning recommendations for your selected tuning categories.

For example, suppose you have selected the Check for Optimal Data Access tuning scope for a tuning session to determine whether one or more tables in a particular schema have appropriate indexes. Although Oracle Expert requires that you collect Schema class data for the Optimal Data Access category, you should select a small number of tables that you want to tune. The more tables you chose, the longer the evaluation will take and the longer the reports will be.

If a SQL History exists, and it is a complete representation of SQL execution in the database, then collecting from the SQL History will perform better.

When you limit the amount of data you collect, you shorten the collection, analysis, and recommendation review cycle.

> **Note:** If you do have to collect a large amount of data, you do not need to visually monitor Oracle Expert for completion. You can configure your machine to produce an audible .WAV alert at the end of the operation. The audible alert will sound whether Oracle Expert is minimized or not. Please refer to your operating system's sound property page for more information.

For each class, the following sections describe:

- the options for the class
- when the class should be collected
- the recommendations Oracle Expert can generate after analyzing the class data
- how to fill out the appropriate page of the Collect Options dialog box to collect the class data

Collecting the Database Class

When the Database class is selected on the Collect page of the tuning session window, Oracle Expert can collect the following categories of Database class data:

- the name, installed version, installed options, and database-wide statistics for the database to be tuned
- database-wide information about database users, tablespaces, and public synonyms

Public synonym data is used to validate workload data. *When Check for Optimal Data Access* is selected, workload is validated as part of the collection. Depending on

which optimal data access scope was selected, public synonym is either automatically collected or needs to be collected as part of the database class.

When performing *comprehensive index evaluation of tables you specify* or performing *index fragmentation evaluation on tables you specify,* you should collect public prior to collecting the workload. If you do not, the workload data you collect may be incomplete.

When *performing comprehensive index evaluation on tables referenced by worst performing SQL statements*, the collection of public synonyms is automatically driven by the SQL validation requirement on the tables.

Collecting Database Class Data from an Instance

You can automatically collect Database class data directly from an instance. Database class data is primarily obtained from the database's data dictionary and dynamic performance tables (V$ tables) You must use this method the first time you collect the database class for a database.

Collecting Database Class Data from a File

You can collect Database class data from an .XDL file (Expert Definition Language) file. Use the Browse button to browse the available .XDL files on your system. The .XDL file for a Database class contains name and version data (including installed option and database-wide statistics data) for a database.

The file can also include SQL information in the form of database users, tablespaces, and public synonyms data.

An .XDL file containing Database class data is created when a Database class is exported or when an entire tuning session is exported.

Re-Collecting Database Class Data

During a tuning session, you can collect the Database class more than once. If you do, you can collect one or more categories of Database class data that have been collected previously. In this situation Oracle Expert replaces the existing category data with the new data collected for the category.

After you have collected the Database class for a tuning session, you do not need to collect it again for that session unless there are changes to:

- the name, version, or installed options (such as enabling or disabling the parallel server or parallel query options) for the database

- database users
- tablespaces
- public synonyms

Collecting the Instance Class

When the Instance class is selected on the Collect page of the tuning session window, Oracle Expert can collect:

- instance statistics
- instance parameters

Oracle Expert collects the instance statistics and instance parameters categories of Instance class data from a database's dynamic performance tables (V$ tables).

Collecting Instance Class Data from One or More Instances

You can automatically collect Instance class data directly from one or more instances. You must use this method the first time you collect the Instance class data from an instance. If only one instance exists for the tuning session, the Instance class will be collected from that instance. If multiple instances exist, choose the instances from which to collect the instance class by moving them to the Collect box on the Instance Collect Options dialog box.

At any given moment, the instance statistics in a database's V$ tables provide a snapshot of how the instance is performing. When you collect the Instance class during a peak period, Oracle Expert uses the data from the instance statistics sample to generate tuning recommendations to increase database performance during that period. Oracle Expert maintains a history of all the instance statistics samples for a database, so when samples are collected during different peak periods, Oracle Expert gains insight into the instance's performance in different situations and generates better tuning recommendations over time.

Collecting Multiple Instance Statistics Samples During a Collection

By default Oracle Expert collects multiple instance statistics samples during an Instance class collection. This enables Oracle Expert to make better tuning recommendations when you collect multiple instance statistics samples during an Instance class collection because:

- a greater volume of statistics is collected, which gives Oracle Expert more insight into how your instance performs in different situations

- additional statistics that only have meaning when collected over time are collected (these statistics are not collected during a single instance statistics sample collection)

The Options section of the Instance Collect Options dialog box lets you collect multiple instance statistics samples during a single Instance class collection and specify the length of time for which the instance statistics samples will be collected. For example, if you select a Duration of 1 hour and a Sample Frequency of 12 per hour, Oracle Expert will collect 12 instance statistics samples in an hour as part of the collection process (each instance statistics sample would begin 5 minutes after the previous sample).

If you request that multiple instance statistics samples be collected, a separate process is created to collect multiple instance statistics samples, so that the collection of instance statistics samples can continue even if you exit Oracle Expert.

Note that while multiple instance statistics are being collected, you cannot:

- collect other Instance class data
- delete instance objects (on the View / Edit page of the tuning session window)
- perform an analysis
- delete service or tuning session

Oracle Expert displays your instance statistics samples choices during the collection process. This dialog box displays the name of the tuning session, the collection duration, the sample frequency, and the time stamp of the last sample taken. It also displays the current status of the instance statistics sample collection (either "Pausing Between Samples" or "Collecting Instance Statistics") and the number of scheduled instance statistics sample collections that have been completed (for example, "2 of 13"). Click the Cancel button to cancel an instance statistics collection. After the current statistics sample is collected, Oracle Expert terminates the collection process.

Collecting Instance Class Data from a File

You can collect instance data for the Instance class from an .XDL file. Use the Browse button to browse the available .XDL files on your system. The .XDL file for an Instance class contains instance statistics and instance parameters data.

If you collect an instance from an .XDL file created by exporting an instance during a previous tuning session, after the import operation you need to manually enter data for the instance's Username and Password attributes. To do this, select the

instance on the View/Edit page, choose Edit=>Modify, and add the correct data on the Attributes page.

An .XDL file containing Instance class data is created when a Database class is exported.

Getting Less Conservative Instance Tuning Recommendations

By design, Oracle Expert is conservative with instance tuning recommendations until it has compiled a statistically significant amount of historical data about an instance.

The value of the "Samples for statistical significance" rule determines the number of instance statistics samples Oracle Expert will collect before reporting recommendations that reduce current instance resource allocations. The default value for the "Samples for statistical significance" rule is 10. This means, for example, that if Oracle Expert determines from the first Instance class collection that the instance's SGA could be reduced by 10% with no performance loss, this recommendation is withheld until 10 instance statistics samples are collected.

To have Oracle Expert report its less conservative instance tuning recommendations more quickly, you can do one or both of the following:

- request multiple instance statistics samples with each Instance class collection

  By collecting more samples with each Instance class collection, you can collect a statistically significant number of samples in less time.

- reduce the default value of the "Samples for statistical significance" rule

  If you reduce the value of the "Samples for statistical significance" rule from 10 to 5, for example, Oracle Expert will report recommendations to reduce instance resource allocations after 5 instance statistics samples have been collected. To change the rule's value, select the database for which you want to change the rule on the View/Edit page of the tuning session window, then choose Edit=>Modify. Select the Rules tab of the Edit dialog box, then the Common tab. Then click in the rule's Value column and enter a new value.

These measures can be most safely employed in a non-volatile database, for example, a database in which the same applications run at the same time every day, and the number of users and amount of data are fairly stable. For this type of database, the instance statistics samples taken at peak periods over time would probably be very similar. This means there is less risk of Oracle Expert making instance tuning recommendations that are detrimental to the database's performance.

Re-Collecting Instance Class Data

You should re-collect instance statistics and instance parameters if you have made any of the following changes to your database:

- enabled or disabled the parallel server option
- enabled or disabled the parallel query option
- added new users
- upgraded the version of the Oracle Server
- changed your parameters

In general, instance statistics should be collected regularly over time to help give Oracle Expert a better sense of how the instance uses resources in different situations.

Collecting the Schema Class

When the Schema class is selected on the Collect page of the tuning session window, Oracle Expert can collect the following categories of Schema class data:

- Schema data, which includes data about tables, columns, indexes, and constraints
- Statistics data, which can be one of the following:
  - statistics about the physical characteristics of tables, indexes, and clusters, including cardinality data (a SQL ANALYZE command collects these statistics) or the data from a previously performed SQL ANALYZE command
  - cardinality data only for the tables that are collected as part of the Schema class collection (Oracle Expert scans the tables that are collected to obtain the cardinality values)

One use of schema data is to validate workload data. If you plan to collect workload data during a tuning session, collect Schema data for the schemas and tables referred to by the workload requests prior to collecting the workload. During a workload collection, Oracle Expert removes those requests that refer to schemas or tables which have not been collected.

Collecting Schema Class Data from One or More Instances

You can choose the instance that will be used to collect schema information.

To collect Schema class data directly from an instance, select an instance from the Instance list box on the Schema Collect Options dialog box. Then click the Get Schemas button, and Oracle Expert accesses the instance and displays a list of all the instance's schemas. To collect Schema class data for all the tables in a schema, move the schema to the Collect box. To reduce collection time, specify only the schemas you want to tune.

If you do not want to collect Schema class data for all the tables in a schema, click the plus sign next to the schema icon to display the list of tables for the schema. Then move the tables you want to collect to the Collect box. To reduce collection time, specify only the tables you want to tune. Schemas and tables marked with a question mark have not had data collected. If a schema does not have a question mark, yet one of its tables does have a question mark, then you need to collect data for that particular table.

When the Schema data category in the Options section of the Schema Collect Options dialog box is selected, the schema data is collected directly from the instance's data dictionary tables.

When the Statistics category of Schema class data is selected, the statistics data is collected directly from an instance. If you specify the Expert Scan option, Oracle Expert executes a SQL "SELECT COUNT DISTINCT" statement during the collection to determine the table cardinality (the number of rows) of each selected table and stores the values in the repository. Oracle Expert automatically obtains column cardinality values (the number of distinct values in a column) for each of the columns in the tables being collected.

If you specify the Run ANALYZE Command option, Oracle Expert executes the SQL ANALYZE command with the STATISTICS option on each selected table. Oracle Expert then retrieves the resulting statistics from the database's data dictionary. This option updates the data dictionary statistics on the target node. It required write access privileges to data dictionary tables.

If you specify Read Existing ANALYZE Results, Oracle Expert assumes that a previous SQL ANALYZE operation was performed and retrieves the existing statistics from the database's data dictionary.

You can obtain exact or estimated statistical information when you select the Expert Scan or the Run ANALYZE Command option. Choose the Exact option to obtain exact statistical information. Select and provide a value for the Estimated/Limit option to obtain estimated statistical information. When the Expert Scan option is selected, the value in the Estimated/Limit text box causes Oracle Expert to scan that number of rows in each selected table before estimating the table's cardinality and the column cardinality for each column in the selected table. When the Run

ANALYZE Command option is selected, the value in the Estimated / Limit text box causes the SQL ANALYZE command to sample that number of table rows before estimating statistics values. Estimated statistics are usually accurate and can be collected more quickly than exact statistics. Oracle Expert provides better recommendations when provided with current statistics.

Collecting Schema Class Data from a File

You can collect the Schema and Statistics data category (cardinality data only) of Schema class data from an .XDL file. Use the Browse button to browse the available .XDL files on your system.

If you collect the Schema data category, Oracle Expert reads only the SQL DDL (Data Definition Language) statements it uses for the Schema data category and ignores SQL DDL statements it does not need. Oracle Expert provides warnings on items in the file that it cannot parse.

Re-Collecting Schema Class Data

After you collect the Schema class the first time, you do not need to collect it again unless one or more of the following is true:

- One or more of the schema definitions has changed since the last collection of the Schema class for the tuning session.
- You want Oracle Expert to collect Schema class data for one or more schemas for which Schema class data has not been previously collected during the tuning session.

In general, however, schema statistics should be collected regularly over time to help give Oracle Expert a better sense of the physical structure of the schema, for example, cardinality and index stagnation.

In the Options section of the Schema Collect Options dialog box, you can specify whether or not to allow Oracle Expert to overwrite existing data for schemas and tables, as follows:

- When you choose the Schema option for Overwrite Existing, if a schema you are collecting has previously been collected, Oracle Expert deletes the previously collected data for the schema before collecting the new data for the schema.
- When you choose the Table option for Overwrite Existing, if a table you are collecting has previously been collected, Oracle Expert deletes the previously collected data for the table before collecting the new data for the table.

Collecting the Environment Class

When Environment class data is required due to the selected tuning categories for a tuning session, Oracle Expert expects to be provided with system data. This category includes memory, CPU, and operating system page size data.

The two ways to provide Oracle Expert with Environment class data are:

1. Entering it manually

   You must use this method the first time you provide system data for a database.

2. Collecting it from a file

   You can use this method if you have previously exported system data during a tuning session.

Entering Environment Class Data Manually

The first time you provide Oracle Expert with system data for a database, you must enter it manually. You cannot manually enter Environment class data using the Environment Collect Options dialog box. You must use the View/Edit page.

Entering System Data Manually

System data includes memory, CPU, and operating system data.

Before you can enter the appropriate values for these attributes, the system must be created on the View/Edit page under the System folder (located under the Environment folder). In some cases, Oracle Expert may have already added to the System folder a system object for the system on which the instance runs.

After you add a new system to the System folder, you should make sure the value of the "Percent physical memory available" rule for the system is appropriate for the database being tuned. This rule represents the percent of physical memory available to the instance on the system for the database being tuned. The default value of 80% assumes that:

- the database being tuned is the only database that runs on the system
- the system is dedicated to the database (no other applications run on the system)

If either or both of these assumptions are incorrect, you should reduce the percentage of the "Percent physical memory available" rule, taking into account the amount of physical memory required for other databases or applications.

If there are multiple instances running on a system, the “Percent physical memory available” rule gives you control over the proportion of available memory to allocate to each instance. For example, you could allocate 40% to instance A, 20% to instance B, and so on.

Collecting Environment Class Data from a File

On the Environment Collect Options dialog box, you can provide Environment class data to Oracle Expert by collecting it from an .XDL file. An .XDL file with Environment class data is created by exporting an Environment class. For example, to export system data, click on System on the View / Edit page, choose Edit=>Export, and supply the name of the .XDL file you want to create. This will copy disk and system information of the current tuning session to the .XDL file.

To collect Environment class data from an .XDL file, specify the name of the .XDL file in the File text box. Use the Browse button to browse the available .XDL files on your system.

Re-Collecting the Environment Class

If the logical device or system data has changed since the last Environment class collection, you should provide Oracle Expert with updated system data.

During a tuning session, you can collect the Environment class more than once. If you do, you can import one or more categories of Environment class data that have been collected previously. In this situation, if you selected the Overwrite Existing option, Oracle Expert replaces the existing category data with the new data imported for the category. If you did not select the Overwrite Existing option, Oracle Expert displays an error message telling you that it cannot overwrite the existing data.

Collecting the Workload Class

When the Workload class is selected on the Collect page of the tuning session window, Oracle Expert can collect Workload class data.

A workload contains data that describes to Oracle Expert the nature, frequency, and relative importance of applications and requests (SQL statements) that access a database.

The Workload class has the single biggest impact on the Oracle Expert data access tuning recommendations. Therefore, when you select tuning categories that require workload data, it is important to provide Oracle Expert with a representative

workload. The SQL History should contain the set of SQL statements used in the database environment.

If you consider some of your applications to be more important than others, you might want to optimize your database to provide its best performance for your more important applications. Oracle Expert can help with this optimization when you provide importance values for workload elements. For more information about providing importance values for workload elements, see "Specifying Importance Values" on page 20-5.

Workload Options

The Workload Collect Options dialog box allows you to specify how to collect the Workload class for your tuning session. In addition, you can tell Oracle Expert how to store the collected workload by selecting one of the following options:

- SQL History options

  As long as SQL History is not selected as the source for your workload collection, the SQL History options allow you to merge, replace or ignore an existing SQL History.

  When you select *Merge source workload with an existing SQL History*, Oracle Expert takes new collection data merges it with the existing SQL History. Oracle Expert then uses the SQL History as the source for filtering statements that are applicable to the tuning scopes specified on the Scope page of the tuning session.

  When you select *Replace existing SQL History with source workload*, Oracle Expert deletes the existing data in the SQL History and create a new SQL History with the newly collected data. Oracle Expert then uses the new SQL History as the source for filtering statements that are applicable to the tuning scopes specified on the Scope page of the tuning session.

  When you select *Do not consider SQL History*, Oracle Expert uses the collected data as the source for filtering statements that are applicable to the tuning scopes specified on the Scope page of the tuning session.

- Tuning Session Workload options

  Allow you to merge or replace the existing tuning session workload with a filtered workload. A filtered workload contains only those statements that are applicable to the tuning scopes specified on the Scope page of the tuning session. For example, if you are only interested in tuning indexes on table

'emp', only relevant SQL statements for table 'emp' will be included in the filtered workload.

If you selected *Perform comprehensive index evaluation on tables referenced by worst performing SQL statements* on the Scope page of the Tuning Wizard, you will see the following additional options:

- TopSQL Option

  Oracle Expert will automatically evaluate the 25 worst performing SQL statements. You can change this number in the Consider Statements box. If you want a complete evaluation of all SQL Statements, choose the Consider all statements option.

- Statistics Option for Objects Referenced by TopSQL

  Allows you to specify the collection and calculation methods for objects referenced by TopSQL. For more information about the collection methods, see "Collecting Schema Class Data from One or More Instances" on page 12-9.

Collecting Workload Class Data from a Database's SQL Cache

If you choose the SQL Cache option, Oracle Expert can collect workload data from the SQL cache of the database you are tuning. At any given time, a database's SQL cache contains the statements that are currently the most frequently executed against the database. Therefore, the statements in the SQL cache are likely to be different at different times, depending on which applications are running against the database.

When there are multiple instances of the database, Oracle Expert collects the statements in the SQL caches of all the instances that are part of the collected data for the tuning session.

A workload request (SQL statement) is invalid if it refers to schema elements for which data has not been collected. Workload validation may or may not occur. Validation depends on the tuning scopes you select. When data access validation takes place, you may or may not have to collect tables.

Collecting Workload Class Data from a SQL History

If a SQL History already exists, you can use the SQL History as a source for collecting workload.

If the source for a workload collection is set to SQL History, the SQL History options will not be available.

The SQL History is shared between SQL Analyze, Oracle Expert, and the Index Tuning wizard. The SQL History is intended to contain a complete set of SQL for the database environment. A SQL History will ensure that consistent index tuning recommendations are made across the three tools.

Collecting Workload Class Data from a File

To collect workload data from an .XDL file, choose the File option. You create an .XDL file by exporting workload data from an existing tuning session.

In some cases it can be useful to collect workload data from an .XDL file even if the workload data is not exactly what you want to use for the database being tuned during a tuning session. After you import a workload from an .XDL file, you can modify the data for individual workload elements until you have a representative workload for your tuning session.

Collecting Workload Class Data from an Oracle Trace Database

You can use Oracle Trace to collect workload data directly from a database in real time (while users and applications are accessing the database). After collecting data with Oracle Trace, you can format it and store it in an Oracle Trace formatted database. This manual refers to a database that contains Oracle Trace formatted data as an Oracle Trace database.

To collect Workload class data from an Oracle Trace database, select the Trace option on the Workload Collect Options dialog box and identify the database from which to collect the Oracle Trace data. You identify the database by supplying information in the username, password, service, and collection name fields.

A workload request (SQL statement) is invalid if it refers to schema elements for which data has not been collected.

Re-Collecting Workload Class Data

During a tuning session, you can collect the Workload class more than once. If you have previously collected workload data for a tuning session, use the Overwrite Existing option to specify how Oracle Expert should manage the previously collected workload data.

Collecting Workload Class Data Manually

You can also manually enter workload data on the View / Edit page of the tuning session window. This is a tedious process. Manually entering workload data is a

reasonable approach only when you are using Oracle Expert to configure a new database (which no applications or users are accessing yet). If you know the important statements that will run against the database, you can manually enter them to help Oracle Expert generate a better initial configuration.

Starting a Collection

To begin a collection, you must have at least one collection class selected. However, if there are no valid options for that collection class, Oracle Expert displays a message stating it cannot perform a collection.

When you have selected valid options for a collection class, a green check mark appears in the Options Set? column for that class on the Collect page of the tuning session window. If there are invalid options for a collection class, a red X appears. Oracle Expert collects data only for collection classes that have valid options.

If you want specific details of the collection's progress while the collection is running, click the Work in Progress button in the toolbar.

Restrictions During a Collection

During a collection, you cannot:

- add, modify, or delete any data on the View/Edit page of the tuning session window
- perform an analysis
- delete service or tuning session

Canceling a Collection

To cancel a collection that is in progress, choose Cancel=>Process.

If some of the selected classes were not collected, a dialog box asks whether you want Oracle Expert to collect the uncollected classes. Choose Yes to continue collecting the other classes, No to stop the collection process and return to the Collect page.

If the collection stops while the last class is being collected, you may be given only the choice to Continue, which ends the collection and returns you to the Collect page.

If a collection stops due to an unexpected error, the dialog box displayed may include the Abort option, which allows you to exit Oracle Expert.

After a collection is terminated, some of the data collected may be incomplete and unusable. If Oracle Expert did not exit when the collection was terminated, a dialog box displayed after the collection informs you that incomplete and unusable data exists for a particular schema. This dialog box gives you the option of having Oracle Expert delete all the data for this schema and its children or having Oracle Expert remove the incomplete flag and use the data as it is. If Oracle Expert exited when the collection was terminated, this dialog box is displayed when you open the tuning session again.

Collecting Invalid Data

Some objects that Oracle Expert collects have dependencies on or references to other objects. If Oracle Expert collects data about a particular object and cannot find collected data about other objects referred to by the object, it displays error messages at the end of the collection and considers the collected object to be invalid. For example:

- Applications in a workload can be marked invalid.

  Oracle Expert considers a collected application to be invalid if it contains invalid requests (SQL statements that refer to schema objects that have not been collected). Only requests that were successfully validated will be used during analysis.

- Tables, clusters, and indexes can be marked invalid.

  When Oracle Expert collects a table, cluster, or index, the segment information for that object is also collected. If the tablespace containing the segment has not been collected, Oracle Expert is unable to create the segment in the repository and considers the table, cluster, or index to be invalid.

- Schemas can be marked invalid.

  If any schema object is marked invalid during a collection, the schema is also marked invalid.

When Oracle Expert determines during collection that a collected object is invalid, the object is marked with the international No symbol on the View/Edit page of the tuning session window.

By default, Oracle Expert includes invalid data when it analyzes tuning session data, but it cannot generate any recommendations regarding invalid objects. Different methods of dealing with invalid objects include:

- validating the object

This method allows Oracle Expert to make recommendations about the object.

- disabling tuning for the object

  This method allows Oracle Expert to exclude the object during an analysis, which speeds up the analysis. Oracle Expert cannot make any recommendations about the object.

- deleting the object

  This method removes the object from the repository.

Oracle Expert discovers that some objects are invalid during an analysis.

13

Viewing and Editing the Collected Data

When you use the *View/Edit Page* in conjunction with the *Edit Pull-Down Menu*, Oracle Expert gives you the ability to view, edit, add to, and delete from the data that you collect. Also, you can view and edit the rules and attributes associated with the data on the View / Edit page of the tuning session window. The hierarchical data on the View / Edit page consists of:

- database
  - instance
  - schema
  - tablespace
  - public synonym
  - database user
- environment
  - system
- workload application

View/Edit Page

To add, modify, or delete an object, select the View / Edit page of the tuning session window. Use this View / Edit page in conjunction with the Edit pull-down menu to perform the tasks you need. Click Help from the View / Edit page for detailed information.

Databases

For the database object, you can view and edit associated attributes and rules.To view and edit these rules and attributes, place your cursor on the name of the database, choose the Edit=>Modify menu option, and choose either the Attributes tab or the Rules tab.

Instances

Under the instance object, you can change instance rules and attributes, and statistics attributes.

Each instance runs on a system. Oracle Expert associates default rules with each instance (use the Edit=>Default Rules menu option). An example of the system rules you can edit is “Default operating system block size.”

Keep in mind as you review and modify the instance rules that the following general principles apply:

- Rules regarding minimums establish the lower boundary of an instance parameter. Oracle Expert will not make recommendations below this value. For example, if the minimum value for the db_block_buffers parameter is 50, then Oracle Expert will always recommend at least a value of 50 for that parameter.
- Rules regarding maximums establish the upper boundary of an instance parameter. Oracle Expert will not make recommendations above this value. For example, if the maximum value for the db_block_buffers parameter is 100, then Oracle Expert will not recommend more than 100 for that parameter.
- Rules regarding multipliers establish the factor by which Oracle Expert rule values are multiplied. For example, if Oracle Expert recommends increasing the db_block_buffers parameter of average concurrent users by a factor of 2, then the resulting number will be twice as large as the original number. To disable this user rule, use the value 0. To decrease the factor, multiply by a decimal, for example, 0.5.
- Rules regarding hit ratio (low or high) establish the upper and lower thresholds that affect the Oracle Expert activation point for different categories of rules. For example, if the buffer_cache_hit_ratio parameter falls below a LOW threshold, then Oracle Expert might recommend increasing the db_block_buffers parameter.
- Rules that enable and disable the tuning of certain objects govern the capability of Oracle Expert to make recommendations for an object and its children.

- The “Last statistics collection importance” rule (a Common rule) lets you tell Oracle Expert how important historical data is to the recommendations. The possible values are 0 to N (where N is an integer). The higher the number the more importance there is to the most recent collection.

> **Caution:** If you have collected instance statistics, do not change the attribute value(s) for any instance parameters from the View / Edit page. The resulting recommendation(s) can be misleading.

Schemas

A schema is required for some tuning categories. Because Oracle Expert is not a schema editor, it does not keep 100% of a user’s schema, nor does it provide editing capabilities for all possible schema information. In general, Oracle Expert manages only schema objects that are necessary to the tuning process.

Within Oracle Expert, you can edit data for all the schemas you can access in your database. The objects within a schema that you can edit are: tables (including columns, indexes, and constraints), clusters, and synonyms.

If a schema object is missing or incomplete for a tuning session that expects schema data, Oracle Expert will not attempt to make recommendations on objects related to the missing schema object.

The accuracy of the necessary data is very important. When Oracle Expert attempts to form tuning strategies, it assumes that the input data is accurate. If that is not the case, the quality of the recommendations may be unreliable. For example, cardinality is a critical attribute in both tables and columns. Oracle Expert relies heavily on the cardinality values to predict sizing and index formation. If the cardinality values are incorrect or missing, Oracle Expert may recommend very poor index structures or no index structures at all.

Tables Oracle Expert needs the physical structure of a table to do sizing and access methods tuning. Oracle Expert uses the following table information:

- cardinality (number of rows in a table)
- record size (physical size of all columns)
- columns
  - datatypes and sizes
  - cardinality (relative uniqueness of column data)

  - column constraints
- indexes - Oracle Expert manages all index characteristics except the PARALLEL option. Existing indexes can be collected from a database and analyzed by the rules to determine usefulness. The following index attributes are managed:
  - index characteristics (uniqueness, type, and so on)
  - index columns
  - storage characteristics and statistics
- table constraints
- storage attributes and statistics

Views Through the View information, you can inspect the SQL text that makes up the view. You can also display the columns that make up the view.

Oracle Expert uses the View information to track SQL text backwards to the appropriate base table. If no workload exists, Oracle Expert can use views to form access methods strategies based on SQL text.

Clusters As it does with indexes, Oracle Expert supports all cluster attributes except the PARALLEL option. If appropriate, Oracle Expert can analyze table accesses to identify clusterable entities.

Synonyms Oracle Expert uses synonyms for validation of workload requests. Oracle Expert processes the synonyms to find out what objects the synonyms reference, such as a table.

Tablespaces

As with other physical schema information, Oracle Expert considers a tablespace to be a tunable entity. It may recommend changes to existing tablespaces or it may recommend new tablespaces.

Public Synonyms

Because SQL statement analysis may identify references to synonyms, Oracle Expert must have full knowledge of all public and private synonyms.

Database Users

When a database user is defined, two types of tablespaces are used to define the database user: default tablespace and temporary tablespace. Through space

management, Oracle Expert uses database user information to verify that these tablespaces have the attributes necessary for optimum performance.

Environment

The environment object displays environment data about the database you are tuning.

System

Oracle Expert uses the system information and the physical characteristics of the hardware where the instance is running to evaluate optimal parameter settings. For example, Oracle Expert uses the Total Memory attribute in conjunction with the Memory Utilization attributes to assess the SGA configuration for each instance.

Workload Applications

Workload data describes to Oracle Expert the nature, frequency, importance, emphasis and rank of transactions that access the database. You can edit the following workload elements:

- application
- request (SQL statement)

Importance is the value assigned to a workload element. You can supply any value (up to 9999), with lower values being of less significance.

Frequency is the number of times a request is repeated when a specific application is executed.

Emphasis or statistical weight determines which factors are used to calculate importance.

Rank is given to a request based on the calculated importance. Rank is a good indicator of how valuable the element is in relation to other workload elements.

Note: Frequency and importance affect the recommendations produced by Oracle Expert. (*See Chapter 20, "Managing Workloads" for more information.*)

Application

You can edit the importance of an application. An application is the workload category that is used to group requests.

Request

You can edit the importance, frequency, and SQL statements of a request. A request is a SQL statement. Requests are the elements below Applications.

14

Generating and Reviewing Recommendations

The topics in this chapter include:

Generating Tuning Recommendations

When you have finished collecting and editing your tuning data, Oracle Expert is ready to generate tuning recommendations. Select the Recommendations page of the tuning session window.

When you click the Generate button, Oracle Expert begins its analysis of the collected data. If you want specific details of the progress of the analysis, while the analysis is running, click the Work in Progress button in the toolbar.

After an analysis is complete, you can examine the tuning recommendations on the Recommendations page. Note that Oracle Expert considers the interdependencies between recommendations while generating the list of recommendations for an analysis. If you use all the recommendations from an Oracle Expert analysis, you will be implementing the best overall Oracle Expert tuning recommendations for your database for the tuning categories you selected, based on the collected data provided for the analysis.

Analyzing Data Efficiently

Before performing an analysis, Oracle Expert looks at the tuning categories you have selected for the tuning session. Then, during the analysis, Oracle Expert analyzes only the collected data that it requires to generate recommendations for the selected categories. This means, for example, if you select the instance tuning category for a tuning session, Oracle Expert excludes collected Schema or Workload class data from the analysis because Oracle Expert does not use schema or workload data when making SGA tuning recommendations.

Tunable Rule

The Tunable rule is another name for the "Enable or disable tuning for the current object and its children" rule, which is a Common rule available for some of the objects that you collect during a tuning session. An object's Tunable rule value (either Enabled or Disabled) determines whether or not the object is included in an Oracle Expert analysis. If the Tunable rule is not available for an object or you do not explicitly set a value for its Tunable rule, Oracle Expert uses the value of the rule from the next higher level of the View/Edit page hierarchy where the rule is instantiated (or the default value of the rule, if none of the higher objects in the hierarchy have instantiated the rule). If you explicitly set the Tunable rule value for an object, Oracle Expert uses that value for the Tunable rule. The default value of the Tunable rule for all objects is Enabled.

A tuning session analysis excludes objects with a Tunable rule value of Disabled. Oracle Expert does not generate any tuning recommendations for these objects or their children.

Suppose you select the Access Methods tuning category for a tuning session because you want Oracle Expert to determine if a particular table is properly indexed. To perform Access Methods tuning, Oracle Expert requires that Schema class data be collected for the schema that includes the table for which you want recommendations (see Table 12–1). However, because you are interested in recommendations on only a single table, you can exclude the other tables in the schema from the analysis (by setting the Tunable rule's value to Disabled for the tables for which you do not want recommendations).

As another example, suppose you select the SQL Tuning category for a tuning session because you want Oracle Expert to tune the SQL statements for two applications. To perform SQL tuning, Oracle Expert requires that Workload class data be collected for these applications. If you have previously collected workload data for other applications besides the two you are interested in tuning, the other

applications can be excluded from the analysis. You can exclude an application from an analysis by setting the value of the application's Tunable rule to Disabled.

You can verify and change the value of the Tunable rule for an object by following these steps:

1. Select the object on the View / Edit page of the tuning session window.
2. Choose Edit=>Modify.
3. Choose the Rules tab.
4. Choose the Common tab.
5. Optionally, change the Tunable rule's value.

Deleting an Object

Another way to exclude an object from an analysis is to delete the object. Select the object on the View / Edit page and choose Edit=>Delete.

Restrictions During an Analysis

While an analysis is in progress, you cannot:

- change the tuning scope or business characteristic values
- add, modify, or delete data on the View / Edit page of the tuning session window
- perform any collections
- generate implementation files
- delete service or tuning session

Canceling an Analysis

To cancel an analysis that is in progress, choose Cancel=>Process. After confirming that you want to cancel the analysis, Oracle Expert displays a dialog box with the Continue option, which returns you to the Recommendations page.

Invalid Objects Discovered During an Analysis

When Oracle Expert collects data for a tuning session, during the collection process it can determine that some objects are invalid. For more information about objects

that are marked invalid after a collection, see "Collecting Invalid Data" on page 12-18.

During an analysis, Oracle Expert performs additional validation and can discover other invalid objects. For example, Oracle Expert may find schema objects that refer to other schema objects that have not been collected. Oracle Expert considers a schema object to be invalid if it refers to another schema object for which data has not been collected.

The Oracle Expert Analysis report describes invalid objects discovered during the analysis. Oracle Expert includes invalid data when it analyzes tuning session data, but it does not generate any recommendations regarding invalid objects. For more information about generating an Analysis report, see "Generating an Analysis Report" on page 16-1.

Invalidating an Analysis

A valid analysis is one that completes without errors. If a valid analysis exists and you attempt to perform a collection or modify or delete tuning session data, Oracle Expert warns you that the operation will invalidate the analysis. You will be given a choice as to whether or not to continue with the operation. If you continue, the analysis will be marked as invalid, which means:

- when you attempt to generate an Analysis report, you receive a message that the analysis is invalid
- when you click the View Detail button on the Recommendations page, you receive a message that the analysis is invalid
- you will not be able to generate implementation files on the Implement page from the invalid analysis

Reviewing Tuning Recommendations

When Oracle Expert finishes analyzing the collected data, you can select the Recommendations tab on the tuning session window. This displays the Recommendations page, where you can review the Oracle Expert tuning recommendations.

On the Recommendations page, recommendations of a particular type are grouped together under descriptively named folders similar to the folders on the View / Edit page. Instance rules are shown in descending order of priority. To view the recommendations of a particular type, click the plus sign (+) for the folder.

For more information about why Oracle Expert made a particular recommendation, select the recommendation and either click the View Detail button or double-click the recommendation. Details of the rationale for the recommendation will then be displayed.

When reviewing the Instance Analysis recommendations, you may decline individual recommendations. To decline a recommendation, select the recommendation and click the right mouse button, then click Toggle State. The icon next to the selected recommendation will be marked with the international No symbol. The recommendation's state can be restored by clicking Toggle State again.

When you decline a recommendation, you must perform an analysis again because other recommendations may have a dependency on the declined recommendation.

When you decline a recommendation, Oracle Expert instantiates an override rule for the parameter that was declined, using the current parameter setting. For example, if the parallel_max_servers parameter is currently set to 5 and you decline the "Decrease parallel_max_servers to 0" instance recommendation, Oracle Expert instantiates the "Parallel_max_servers parameter override" rule with a value of 5 for the Parallel query parameters object.

To see the newly instantiated override rule, select the Parallel query parameters object on the View / Edit page, choose Edit=>Modify, select the Rules tab, then the parallel_max_servers tab. As long as the "Parallel_max_servers parameter override" rule is instantiated at the Parallel query parameters object level, Oracle Expert will use the current value during instance evaluation for the tuning session. If you remove the instantiation of the "Parallel_max_servers parameter override" rule, Oracle Expert will re-evaluate the parameter.

If you change your mind about declining a recommendation, toggle its state by highlighting the recommendation and clicking the right mouse button, and then selecting Toggle State. This action also removes the instantiation of the affected override rule at the instance parameter level, which means Oracle Expert can make this recommendation again during the tuning session. Once you decline any recommendations and perform an analysis, you must manually remove the instantiation of the affected override rule (as described in the previous paragraph) before Oracle Expert will re-evaluate the parameter for the tuning session.

After you decline one or more recommendations from the current analysis, another analysis must be performed to re-evaluate all recommendations because other recommendations may have a dependency on the declined recommendations. If you attempt any of the following operations, a message is displayed advising you that declined recommendations exist:

- generate an Analysis or Recommendation Summary report (by choosing the appropriate option from the Report menu)
- display more detail about a recommendation on the Recommendations page (by double-clicking on the recommendation)
- generate implementation files on the Scripts page (by clicking the Generate button)

You have the option of continuing with these operations after receiving the message. Note that if you generate implementation files after declining one or more recommendations, the implementation files will contain parameters and scripts to implement ***all*** of the Oracle Expert recommendations (not just the recommendations you chose to use).

If you decline one or more of the recommendations, you can return to the Analyze page of the tuning session window and begin another analysis of the collected data. Oracle Expert will take the recommendations you declined into account, then generate new recommendations (after considering the interdependencies between the recommendations). The new set of tuning recommendations does not include the recommendations you declined.

During an analysis, Oracle Expert may generate one or more recommendations that you cannot explicitly decline (these appear as grayed-out recommendations on the Recommendations page) and that you do not want to implement. If so, you can change rules at the appropriate object level to influence Oracle Expert so that these recommendations are not generated during future analysis sessions, or you can manually edit the recommendations in the implementation files.

For example, suppose Oracle Expert recommends that you create a new index with 16 segments on a particular table. You may agree with the recommendation to create the index, but you do not want to create any indexes with more than 6 segments. In this situation, you can go to the View / Edit page, select the table for which Oracle Expert recommended the 16-segment index, and choose Edit=>Modify, which displays the Edit dialog box. Go to the Rules page, select the Index Analysis Parameter tab, and click the Advanced button. Find the “Maximum index key segments” rule, and change its value to 6. This will ensure that Oracle Expert recommends indexes with 6 or fewer segments for the table during future analysis sessions.

When Oracle Expert has generated recommendations you want to implement, select the Implement tab, which displays the Implement page.

To review all of the recommendations for a tuning session and the rationale for those recommendations, generate an Analysis report by choosing Report=>Analysis.

15

Implementing the Recommendations

The topics in this chapter include:

Implementing Tuning Recommendations

After Oracle Expert has analyzed the collected tuning data and generated recommendations you want to implement, select the Scripts tab of the tuning session window. This page allows you to generate files to help you implement the recommendations that Oracle Expert has suggested.

The Scripts page allows you to see:

- a description of the files Oracle Expert will create to help you implement its current recommendations
- the location where Oracle Expert will create each file

  You can select and edit the location field to have Oracle Expert create a file in a different directory.

After you select the implementation files you want Oracle Expert to create, click the Generate button. Oracle Expert creates the specified files.

How to Use the Implementation Files

Table 15–1 shows the type of implementation file that Oracle Expert generates to help you implement each type of tuning recommendation it makes.

Table 15–1 Implementation Files for Different Types of Recommendations

Type of Recommendations	Type of Implementation File
Instance parameter changes (instance tuning)	.ORA file
SQL changes (access method tuning and structure tuning)	.TXT file

To implement instance recommendations, replace the instance parameter values in your instance's INIT.ORA file with the instance parameter values in the .ORA file generated by Oracle Expert. You can also import the changes into your current configuration from Instance Manager.

To implement recommendations made on a SQL object, for example, index, tablespace, or database user, examine the .TXT file generated by Oracle Expert. This file contains SQL statements that include the string "<TBS>" in places where you must provide the appropriate information. It also contains comments explaining steps whose commands must be supplied; for example, the steps necessary for relocating a table to a different tablespace. After you have entered the correct information, the statements in this file can be executed.

Note: You may want to use other Oracle Enterprise Manager applications to perform some of the more complicated steps; for example, relocating a table to a different tablespace using Tablespace Manager.

16

Generating Reports

This chapter describes how to generate Oracle Expert reports. The topics discussed in this chapter include:

- Generating an Analysis Report
- Generating a Recommendation Summary Report
- Generating a Session Data Report
- Generating a Cross Reference Report

By default, each report is displayed after it is generated. To change the default, remove the check mark from the View After Creation button.

Generating an Analysis Report

You can generate an Analysis report to review all of the recommendations for a tuning session, as well as the rationale for those recommendations. An Analysis report for a tuning session is available only after Oracle Expert has analyzed the collected data.

During an analysis, Oracle Expert sifts through the collected data, uses its rules to generate tuning recommendations, then stores the information for the Analysis report in the repository. The Analysis report information stays in the repository until another analysis is performed for the tuning session.

You generate an Analysis report by choosing Report=>Analysis. This displays the Analysis Report dialog box, which allows you to specify a file into which Oracle Expert will write the Analysis report.

Generating a Recommendation Summary Report

You can generate a Recommendation Summary report to review all of the recommendations for a tuning session. A Recommendation Summary report for a tuning session is available only after Oracle Expert has analyzed the collected data.

You generate a Recommendation Summary report by choosing Report=>Recommendation Summary. This displays the Recommendation Summary Report dialog box, which allows you to specify a file into which Oracle Expert will write the Recommendation Summary report.

Note: One subtle difference between the Analysis report and Recommendation Summary report is that the Analysis report includes *suggestions* as well as recommendation information. Suggestions include observations about the database environment for which no specific recommendation can be made.

Generating a Session Data Report

You can generate a Session Data report to get detailed information about the data collected in the repository for a tuning session. If you generate this report before beginning an analysis, you can check to make sure that all the data you planned to collect has been made available to Oracle Expert.

You generate a Session Data report by choosing Report=>Session Data. This displays the Session Data Report dialog box, which allows you to specify a file into which Oracle Expert will write the Session Data report. For readability, you should print the Session Data report in landscape format (132 columns). When viewing the report in an editor, choose the Fixedsys font type, the Regular font style, and 12 for the font size.

Generating a Cross Reference Report

You can generate a Workload Cross Reference report to review a table and its associated requests or a request and its associated table in your tuning session's workload. The report lists the information both ways so that you can quickly locate the information you need. This can be used to help you determine if your workload for a given table is complete.

You generate a Workload Cross Reference report by choosing Report=>Workload Cross Reference. This displays the Workload Cross Reference dialog box that allows

you to specify a file into which Oracle Expert will write the Workload Cross Reference Report.

For readability, you should print the Session Data report in landscape format (132 columns). When viewing the report in an editor, choose the Fixedsys font type, the Regular font style, and 12 for the font size.

17

Using Oracle Expert Effectively

This chapter presents four major areas for you to consider as you seek to maximize the benefits provided by Oracle Expert. These areas include:

- Defining the Appropriate Tuning Scope
- Providing Complete and Accurate Data
- Using Iterative Tuning for Improved Performance
- Taking Advantage of Rules

By following the guidelines presented in this chapter, you will minimize the hardware resources and the time required for Oracle Expert to complete an evaluation.

Defining the Appropriate Tuning Scope

Oracle Expert provides you with the ability to selectively refine the scope of a tuning session. The tuning process consists of selecting both the appropriate tuning scope and collecting the appropriate subset of information about the database environment.

The tuning scope helps you to direct Oracle Expert into specific problem areas. This reduces both the amount of information to be collected and the time required to complete the analysis. For large or complex database environments, this means you get results in minutes rather than hours or days.

For example, performing data access analysis for a particular schema rather than a table in that schema results in large performance differences. Oracle Expert eliminates from consideration SQL statements that are not relevant to the currently selected tuning focus. Many more statements can be eliminated early in the

evaluation process when the data access tuning focus has been specified at a table level, so less work is required than if a complete schema is tuned.

The tuning scope determines the options that are available to you from the Collection page. Focused tuning eliminates the need to collect certain types of information, which saves time.

Providing Complete and Accurate Data

The recommendations generated by Oracle Expert are only as good as the information you provide. The more complete and accurate the data you supply is, the better the recommendations will be, because Oracle Expert will not have to make as many assumptions.

For example, if you are performing instance parameter tuning, you should supply accurate system information. System information includes details such as the amount of physical memory and different resource uses for the system. The instance tuning rules use this information during the evaluation of many of the parameters.

Another area that affects the evaluation process is the business characteristics. Specific rules within the Oracle Expert knowledge base are dependent on business characteristic settings. For example, the type of database environment you specify, either Data Warehousing, OLTP, or Multipurpose, has a large influence on the evaluation process. If you allow this particular business characteristic to default to OLTP and your database environment is primarily Data Warehousing, certain features that optimize performance for Data Warehousing environments will not be recommended.

It is also important to ensure that the information available in the Oracle Expert repository is up-to-date. Changes to the database, such as server upgrades or adding an index to a table, influence the evaluation process. Providing up-to-date information will result in up-to-date recommendations.

Using Iterative Tuning for Improved Performance

Oracle Expert maintains historical information about your database environment within its repository. Successive iterations of a tuning session provide additional performance optimizations over time. These successive iterations are also useful for tuning the environment as resources and workloads fluctuate.

Iterative tuning is particularly useful for instance parameter tuning. Oracle Expert provides two mechanisms for collecting statistics about the instance. These are:

- single snapshot

- duration-based statistics collection

The single snapshot mechanism uses the information from a single snapshot of the Oracle Server's dynamic performance tables. This is only a subset of the information available from the duration-based mechanism. In addition, the single snapshot approach cannot handle situations where the dynamic performance tables overflow the space allocated for them.

The duration-based snapshot mechanism uses the differences between two snapshots from the Oracle Server's dynamic performance tables. Multiple instance statistics records can be collected using this mechanism for a user-selected time period and duration. This approach collects additional statistics that are not available from the single snapshot approach

To effectively use Oracle Expert's instance parameter rules, you should collect instance statistics using the duration-based snapshot mechanism for time periods when the database is heavily used. This is the period when potential problems and resource bottlenecks will be most evident. In addition, the Oracle Expert rules are implemented to be conservative when making recommendations that result in resource reductions. Resource reduction rules require a minimum of 10 instance statistics samples to be collected before the rules are considered to be valid. This can be changed by setting the "Samples for statistical significance" common rule.

Another advantage of performing iterative tuning with Oracle Expert over time is that as changes within the database environment, such as server upgrades or hardware changes, are made Oracle Expert will automatically identify any dependencies that occur.

Taking Advantage of Rules

The Oracle Expert rule mechanism allows advanced users to have ultimate control over the analysis process. Rules can be used to:

- override the Oracle Expert recommendations
- limit the evaluation of specified objects within the database environment
- restrict or disable other rules
- constrain the range of certain Oracle Expert recommendations

For example, the "Percent physical memory available" rule controls the amount of physical memory on a system that can be used for a database instance. By default, the "Percent physical memory available" rule is 80 percent of the total memory available. If the user knows that the system must support multiple instances, the

"Percent physical memory available" rule can be instantiated for the system and adjusted to a lower value of 40 percent. The subsequent analysis would ensure that no more than 40 percent of the total memory on the system was used for the instance.

Rules reside with the object that they affect. The "Percent physical memory available" rule resides with the system object, because it affects the amount of total memory available to the instance on the system. The "Maximum sorted indexes per table" rule resides with the table, because it affects the number of indexes allowed on the table.

Using Existing Analysis Statistics for Very Large Tables

Oracle Expert does a "select distinct count *" for each column in the table when an Expert Scan is performed. This can take a very long time for large tables. To save time, use existing analysis statistics if they are current.

Index Rebuild Detection Requires Analysis Statistics

To make recommendations, the *Check for Optimal Data Access* requires ANALYZE statistics. Expert Scan does not generate statistics required for detecting indexes that should not be rebuilt.

Use SQL History to Avoid Extra SQL Collections

To avoid extra SQL collections, collect SQL History and use it as the source for each tuning session collection. The SQL History collection prevents you from having to recollect SQL from cache or trace.

18

Initial Configuration

Initial configuration implies that database attributes and parameters must be formed prior to building the desired database. However, that is not necessarily the easiest method to use.

When you use Oracle Expert to help configure a new database, you should allow the Oracle Server to create a default instance of the database and then build upon that implementation. By allowing a default database to be created, you enable Oracle Expert to acquire automatically much of the needed information regarding your database.

This chapter describes the following:

Advantages of Using Oracle Expert for Initial Configuration

When you use Oracle Expert for initial configuration tuning, Oracle Expert can provide more appropriate instance parameters for the database you are creating than the Oracle Server default instance parameters. Oracle Expert takes the following data into account before recommending instance parameters for the database you are creating:

- system data

  Includes memory and CPU data, plus the operating system page size.

- database class data

Includes the version of the database, specified options, and database user, tablespace, and public synonym data.

- business characteristic values

  Includes data about the type of application (OLTP, Data Warehousing, or Multipurpose) that runs against the database, your tolerance for unscheduled downtime, and other business characteristic values.

Oracle Expert analyzes all this data before providing instance parameter recommendations for a database you are creating. Oracle Expert makes better recommendations when provided with complete and accurate system, database class, and business characteristics data.

Configuring a New Database

Initial configuration allows you to start with a newly created Oracle database and quickly customize the database for your environment. As additional information about the environment becomes available, use Oracle Expert to further tune the database.

Performing Initial Configuration for a Database

Perform the following steps:

1. Discover the database from the Oracle Enterprise Manager console.
2. Select the database you want to tune from the Oracle Expert tree list.
3. Create a new tuning session by choosing File=>Create.
   a. To change the default name of the tuning session, click on the default name and type in the new name.
   b. On the Scope page, select all the instance tuning categories. Do not select the application and structure tuning categories.
   c. In the Business Characteristics section of the Scope page, choose the type of application (for example, OLTP, Data Warehousing, Multipurpose) you expect to run against the database. Adjust the other business characteristics according to the type of database you want.
4. On the View / Edit page of the tuning session window, do the following:
   a. Click on the System folder. Choose Edit=>Add. For the Name attribute, type in the name of the system where the instance will run. Supply the most

accurate values you can for the memory, CPU, and operating page size parameters. Click the OK button to save the values you added.

b. Click on the new system icon. Choose Edit=>Modify. On the Rules page, supply the most accurate value you can for the "Default concurrent sessions" rule.

c. Click on the Database object, choose Edit=>Modify and enter the values for the attributes that do not contain values.

d. Click on the Instance object, choose Edit=>Modify and enter the name of the system object you just created for the instance's System attribute. Also, enter values for attributes that do not contain values. Click the OK button to save the values you added.

5. On the Recommendations page of the tuning session window, click the Generate button. Oracle Expert evaluates the system, instance, database, and business characteristics to determine appropriate settings for the INIT.ORA file.

6. On the Scripts page of the tuning session window, click the Generate button to create the INIT.ORA file.

7. Update your database's INIT.ORA file, using the entries contained in the INIT.ORA generated by Oracle Expert.

Improving the Initial Configuration of the Database

Once you have performed the initial configuration of your new database, improve on the initial configuration by performing the following steps:

1. After the database is up and running, provide Oracle Expert with schema and workload data.

2. Collect instance statistics while requests from a representative workload are accessing the database. The instance tuning recommendations Oracle Expert generates from the instance statistics gathered while the instance is in use will be better than the initial instance tuning recommendations.

Now that the instance has been tuned, Oracle Expert will be able to provide you with good results when you perform application tuning and structure tuning.

User-Provided Information

Providing accurate information about the following aspects of your database helps Oracle Expert generate a good initial configuration:

- default concurrent sessions (system rules)
- amount of CPU and memory available for database use (system attributes)
- hardware information, such as total memory (system attributes)
- business characteristics

If you cannot provide an exact value for some of these items, you can specify a range of values using control parameters, and Oracle Expert will use this data to make initial configuration recommendations. In some cases, Oracle Expert can make initial configuration recommendations for part of your database even when you are unable to provide all the previously mentioned information.

19

Autotune

The Autotune feature of Oracle Expert assists the DBA by automating routine database maintenance and tuning tasks. Autotune is designed to unobtrusively monitor the database over time and make parameter changes on behalf of the DBA. These parameter changes are a subset of the tuning capabilities of Oracle Expert.

Autotune automates the collection and analysis process for certain low risk parameter recommendations that are made within Oracle Expert. The DBA must review the list of recommendations and then choose to implement them.

> **Note:** At this time, Autotune does not recognize Oracle Parallel Server databases. You can manually define a service for each parallel server instance you wish to tune in Oracle Parallel Server.

Once you start Autotune, Oracle Expert starts a separate process on the local client to periodically collect data from the database instance, analyze the data, and log the results of the analysis to the repository. This process continues until you stop Autotune.

Autotune was designed to have minimal impact on performance. It samples performance statistics once every 15 minutes and analyzes the database environment once a day.

Autotune evaluates the following instance parameters:

- open_cursors
- db_file_multiblock_read_count
- db_file_simultaneous_writes
- log_buffer

- checkpoint_process

Autotune complements the existing Oracle Expert technology by simplifying the collection and analysis process.

Starting Autotune

To start Autotune, do one of the following:

- In the left pane of the Oracle Expert main window, click on the database you want to tune, then from the Oracle Expert menu bar select Autotune=>Start.
- In the left pane of the Oracle Expert main window, click the right mouse button over the database you want to tune and choose Start Autotune from the popup menu.

Once you start Autotune, the letter A is superimposed on the database icon to signify that Autotune has been started.

Note: Autotune continues to run until it is either explicitly stopped using the Stop command or the system is rebooted. Exiting Oracle Expert *does not* stop Autotune.

Stopping Autotune

To stop Autotune, do one of the following:

- In the left pane of the Oracle Expert main window, click on the database, then from the Oracle Expert menu bar select Autotune=>Stop.
- In the left pane of the Oracle Expert main window, click the right mouse button over the database and choose Stop Autotune from the popup menu.

Viewing Autotune Recommendations

The View Recommendations option will not become available until Autotune has collected enough sample data to perform an analysis and at least one analysis has been performed.

Note: After you start Autotune, it takes 24 hours before the first analysis is performed.

To view the recommendations generated by Autotune, do one of the following:

- In the left pane of the Oracle Expert main window, click on the database, then choose Autotune=>View Recommendations from the Oracle Expert menu bar.
- In the left pane of the Oracle Expert main window, click the right mouse button over the database and choose View Autotune Recommendations.

Implementing Autotune Recommendations

Once you have reviewed the tuning recommendations generated by Autotune, do one of the following:

- Use the Instance Manager to edit instance parameters identified in the View Recommendations window.
- Edit the INIT.ORA file on the target node to include the instance parameter modifications identified in the View Recommendations window.

Note: Oracle Expert does not automatically implement Autotune recommendations.

20

Managing Workloads

This chapter provides guidelines for collecting and managing Oracle Expert workloads. Topics in this chapter include:

Database Workloads

Workload data consists of SQL statements and SQL statistics. Workload data for a Tuning Session is collected from a source workload. One of the following sources can be used to supply the source workload for a Tuning Session:

- the current SQL cache for the database instance
- an existing Oracle Trace collection
- workload from another tuning session that has been exported to an .XDL file
- the SQL History

When the source workload is collected by the Oracle Expert Tuning Session, it goes through a filtering process. This filtering process removes all SQL statements that are not required by the Tuning Session, leaving only those SQL statements that are related to the database tables to be tuned by the Tuning Session. The tables to be tuned are selected in one of two ways:

1. The Oracle Expert user selects the tables to be tuned in the Schema Collect Options window.
2. The user lets Oracle Expert determine which tables should be tuned based upon the highest impact SQL statements in the source workload.

The set of SQL statements remaining after this filtering process is referred to as the Tuning Session Workload.

Obtaining a good set of SQL statements for the Tuning Session Workload is critical. If the source workload contains a representative set of the SQL statements that are typically executed against the database tables being tuned, then the Tuning Session Workload will provide Oracle Expert with a good basis for tuning. Therefore, it is very important that the Oracle Expert user ensure that the source workload is collected from the database when the relevant SQL is in use by the database application.

Any of the sources referred to earlier can supply a representative set of SQL statements for a Tuning Session depending on the tuning circumstances. For example:

- The Oracle Expert user may want to tune certain tables based upon database activity over the past hour or so. She decides to use the current SQL cache as the source workload for the Tuning Session.
- The user may want to tune certain tables based upon the SQL activity of selected database sessions. He decides to use Oracle Trace to collect SQL data for those sessions and uses the Oracle Trace collection as the source workload.
- The user may want to tune certain tables, but knows that the SQL workload for those tables varies over the application processing period. He decides to use the SQL History feature to collect a snapshot of the SQL cache at various times over the processing period. He merges those snapshots together in the SQL History and then uses the SQL History as the source workload for the Tuning Session.
- The user may know that the Tuning Session Workload from a previous Tuning Session contains a representative set of SQL statements for certain tables that he wants to re-tune. She decides to use an imported workload .XDL file from the previous tuning session as the source workload.

The most comprehensive source workload is the SQL History. The SQL History allows the user to collect SQL workload over time, when the user knows that relevant SQL is available. The SQL History can be created, refreshed or replaced from any of the other source workloads: SQL cache, Oracle Trace collections and XDL files.

Collecting Workload Information with Oracle Trace

You can use Oracle Trace to collect SQL workload data. Oracle Trace collects data about SQL statements executing against a database in real time (while the statements are executing). Oracle Trace allows you to:

- collect data about all of the SQL statements executing against a database

  If you are unsure why performance deteriorates at different times from day to day, Oracle Trace may be able to help. You can use Oracle Trace to collect data about all the SQL statements executing against a database during periods of poor performance.

- collect data for only the SQL statements executed by a particular application

  You can use Oracle Trace to collect workload data for a single application by running the application during a period when there is no other activity against the database. This may only be practical for batch applications that do not require any user input during execution.

  You can run an interactive application requiring user input at a different time than usual to allow Oracle Trace to collect workload data for only that application. If you do, you need to consider whether the data collected by Oracle Trace could be appropriately included as part of a representative workload. A representative workload includes SQL statements that execute against a database during a period for which you want to improve performance. If you collect workload data while an application is running under artificial conditions, the workload may contain different data than a workload collected while the application is run under normal conditions.

- choose the time period during which Oracle Trace will collect the SQL statement data

  You control the duration of an Oracle Trace collection. If you want to obtain workload data for a 15-minute period of poor performance, you can do this by stopping the collection immediately after the poor performance interval ends.

- collect data for particular database sessions

  You can restrict an Oracle Trace collection to one or more database sessions.

Oracle Trace collects data about the sequence in which SQL statements executed. When a given SQL statement executes against a database, it does so within a transaction. For each SQL statement it collects data about, Oracle Trace identifies the transaction within which the statement executed. The order in which statements execute can affect how quickly they execute. When provided with the sequence of

statements within a transaction, Oracle Expert can use this information to generate more effective tuning recommendations.

See the *Oracle Enterprise Manager Oracle Trace User's Guide* for more information about collecting data using Oracle Trace.

Providing Oracle Trace Workload Information to Oracle Expert

After an Oracle Trace collection is completed, use the Oracle Trace format function to format the raw Oracle Trace data and store it in an Oracle database (hereafter referred to as an Oracle Trace database).

You can provide Oracle Expert with workload data that has been collected by Oracle Trace by collecting the Oracle Trace data directly into Oracle Expert from the Oracle Trace database. Use the Workload Collect Options dialog box to import the workload data directly from the Oracle Trace database into Oracle Expert.

Collecting Workload Information from the SQL Cache

An instance's SQL cache contains the SQL statements that are currently the most frequently executed against the instance. Therefore, if you want to collect the most frequently executed SQL statements from a particular application or group of applications, you can collect this data from the SQL cache while these applications are executing. To collect SQL statements from the SQL cache of one or more instances, choose the SQL Cache option on the Workload Collect Options dialog box.

Collecting Workload Information from an .XDL File

You can provide Oracle Expert with workload data that has been collected for another tuning session by exporting that session's workload data to an .XDL file. You then import the .XDL file into your current tuning session. Use the Workload Collect Options dialog box to import the workload data directly from the SQL cache into Oracle Expert.

Collecting Workload Information from the SQL History

If a SQL History exists, you can use the SQL History as a source for the tuning session workload.

The SQL History is shared between SQL Analyze, Oracle Expert, and the Index Tuning wizard. The SQL History is intended to contain a complete set of SQL for

the database environment. A SQL History will ensure that consistent index tuning recommendations are made across the three tools.

Specifying Importance Values

One of the major benefits of Oracle Expert is that you can use it to optimize the performance of your business's most important applications by tuning the indexes on high impact or frequently accessed tables. Oracle Expert uses the importance value for each element of the tuning session workload to determine the most important applications, then generates recommendations to optimize their performance. An element can be either an application or a SQL request. Importance values can be specified at either level.

Oracle Expert computes an element's relative importance value using the element's importance value and frequency value. When application and request data is collected by the workload source, Oracle Expert automatically assigns these elements an importance value of 5000. Unless you believe that all the elements in your workload are equally important, you may want to consider changing their importance values.

Oracle Expert computes the relative importance of individual workload elements based on the Primary workload emphasis value. The emphasis value is based on either a user-supplied importance, calculated frequency of executions or the physical I/O count. By default, Oracle Expert gives more statistical weight to I/O. This can be changed by modifying the Primary workload emphasis and Secondary workload emphasis user rules.

To change the default method of computing relative frequency, change the value of the Workload emphasis rule. This rule can be modified at the Workload Application level on the View/Edit page. To change the value of the Workload emphasis rule for an Application and its children, select a workload Application, choose Edit=>Modify, select the Rules tab and the Workload tab of the Edit dialog box, and make the desired change.

For every element in every category of the workload hierarchy, you can provide an importance value between 1 and 9999, with 1 being the lowest importance value.

Oracle Expert ranks the elements in the highest category of the workload hierarchy (Applications) to be the most important, and elements in each of the lower categories to be proportionately less important. What this means in practice is that Oracle Expert largely determines the relative importance of a given application or request element by taking into account the relative importance of the element's parent in higher categories of the workload hierarchy.

The importance value for an element in one of the lower three categories does not entirely determine its relative importance. Instead, much of an element's relative importance is determined by the relative importance of its parent in the workload hierarchy. Any of the Requests that are part of the most important Application in a workload will have a higher relative importance than any of the Requests for less important Applications.

You need to know which SQL statements are part of which applications in your workload to get the best performance for your most important applications.

What if you change the default behavior so that Oracle Expert gives more weight to the importance value of an element when computing its relative importance? Oracle Expert follows the same basic principles in computing the relative importance of elements. That is, the elements in the highest workload category (Applications) are still considered to be the most important and elements in each of the lower categories are considered to be proportionately less important. Therefore, the relative importance of a given workload element is still largely determined by the relative importance of the element's parents in higher categories of the workload hierarchy.

The difference is that when frequency is the dominant factor in determining relative importance, Oracle Expert deems the Application with the highest frequency value to be the most important element in your workload. Oracle Expert is also likely to give higher relative importance values to Requests that are part of that Application than to Requests that are part of Applications with lower frequency values.

To modify the importance value or frequency value for a workload element, select the element on the View / Edit page, choose Edit=>Modify, and enter a new value on the Attributes page.

Replacing a Tuning Session's Workload Data

The tuning session Workload Update option on the Workload Collect Options dialog box provides you with the ability to merge or replace workload data. When you select Replace existing tuning session workload with filtered workload, Oracle Expert deletes all existing workload data and replaces it with the workload data being imported for that Application.

Merging Workload Data

The Merge filtered workload with existing tuning session workload option on the Workload Collect Options dialog box pertains to workload data at the Application level. When you select Merge filtered workload with existing tuning session

workload, Oracle Expert saves the existing workload data for a particular Application and also saves the workload data being imported for that Application. Oracle Expert assigns a new name to the newly imported workload data for the Application.

Consider how this works in practice. The first time you import workload data from Oracle Trace, Oracle Expert stores that data as an Application named Server Application. Then, if you later collect other workload data using Oracle Trace and import it into Oracle Expert with the Merge filtered workload with existing tuning session workload option enabled, Oracle Expert preserves the existing data for the application named Server Application and stores the data you are importing as an Application named Server Application 1. The next time you import workload data collected by Oracle Trace and enable the Merge filtered workload with existing tuning session workload option, that workload data will be named Server Application 2, and so on.

Part V

Getting Started with the Oracle Index Tuning Wizard

The Oracle Index Tuning wizard is an Oracle Enterprise Manager integrated application used to detect tables with inefficient indexes. Once detected, the wizard makes recommendations that will improve access to those tables. This section describes how and when to use the Oracle Index Tuning wizard.

This part contains the following chapter:

21

Introduction to Oracle Index Tuning Wizard

The Oracle Index Tuning wizard (hereafter referred to as Index Tuning wizard) is a software application that identifies tables with inefficient indexes and makes recommendations which will improve access to those tables.

The Index Tuning wizard:

- Identifies tables in need of index changes
- Presents its findings as written recommendations
- Implements the recommendations for you

The Index Tuning wizard is intended for use with the Oracle cost-based optimizer. The recommendations made by the Index Tuning wizard will optimize index usage for the Oracle cost-based optimizer. Therefore, you should not use the Index Tuning wizard for those schemas where rule-based optimization is used.

This chapter describes when to use the Index Tuning wizard, how to access the Index Tuning wizard, and the Index Tuning wizard interface.

When to Use the Index Tuning Wizard

You can use the Index Tuning wizard to proactively maintain optimal indexes for your database. You should run the Index Tuning wizard regularly to evaluate whether index changes should be made to improve SQL query performance. The Index Tuning wizard may recommend adding new indexes, changing existing indexes, removing unused indexes, or changing the type of an index.

You should also use the Index Tuning wizard when one of the following situations occurs:

- A user has reported unacceptable response times for a query

- New applications have been added to the database environment
- Existing application SQL has been modified
- The database server has been upgraded to a new version
- Table sizes within the database have increased substantially

Any of these factors may impact the indexing decisions for the database.

Accessing the Index Tuning Wizard

You can access the Index Tuning wizard in the following ways:

- From the Oracle Enterprise Manager console palette
- Through the Oracle Expert application

  If you have the Oracle Expert application installed, you can launch the Index Tuning wizard from the Oracle Expert Tools menu.

> **Note:** In either case, you must first select a database in the navigator tree before launching the Index Tuning wizard.

The Index Tuning wizard makes two database connections:

1. The Index Tuning wizard connects to its repository to access and update database workload information required for index tuning.
2. The Index Tuning wizard uses the database credentials defined in Oracle Enterprise Manager to connect to the target database for index tuning.

Index Tuning Wizard Interface

When you first access the Index Tuning wizard, you are greeted with a Welcome screen that provides some of the advantages of using the Index Tuning wizard.

You will be lead through the following screens:

- Application Type
- SQL Statements
- Evaluation Focus
- Index Recommendations

- Index Recommendations - Details
- Implementation Summary

Each of these screens is described in the following sections.

Application Type

On this screen, you choose the type of application that is primarily being used for the target database being tuned: Online Transaction Processing (OLTP), data warehousing, or multipurpose.

SQL Statements

The first time you run the Index Tuning wizard you will not see this screen unless you have already collected a SQL history from Oracle Expert or Oracle SQL Analyze. The Index Tuning wizard will automatically collect your SQL statements from the SQL cache and save these statements in a SQL history for the next time you run the Index Tuning wizard.

> **Note:** There are three applications that use the SQL history discussed in this section: the Index Tuning wizard, Oracle Expert, and Oracle SQL Analyze. It is possible that you may have already collected a SQL history from Oracle Expert or Oracle SQL Analyze.

The Index Tuning wizard evaluates the SQL workload of the target database. On this screen, you are given the option of using your existing SQL history or updating your SQL history with new SQL statements from the SQL cache.

A SQL workload is the set of SQL statements that are responsible for the majority of your database processing. This workload must include a good representative profile of SELECT, INSERT, UPDATE, and DELETE statements. This workload is automatically collected and saved during previous Index Tuning wizard sessions.

The workload can also be collected by using the SQL History feature of Oracle Expert and Oracle SQL Analyze. If you have collected a SQL workload from either of these sources and consider the workload to be sufficient, then select "no".

If you are unsure if the collected workload is sufficient, or if you know that your workload has changed, then you should update your workload with a snapshot of the current SQL cache by choosing "yes".

Evaluation Focus

On this screen, you have the opportunity to focus the index tuning evaluation. You can let the Index Tuning wizard focus the evaluation on specific tables that are being used most frequently by the highest impact SQL statements in your workload. SQL statements are judged to be high impact based on their relative disk I/O volume.

Optionally, you can select the schemas(s) you want tuned.

The purpose of the evaluation focus is to hide recommendations for schemas that you are not responsible for tuning or you do not wish to tune at the current time.

When you first use the Index Tuning wizard, it is best to let the wizard focus the evaluation so you will see an overall view of the problem areas. You can then focus on specific areas of the database for which you have control.

Index Recommendations

From this screen you can:

- Have the Index Tuning wizard generate the index recommendations, that is, collect and analyze the data to provide the index recommendations.
- Choose the index recommendations to be implemented.

Note that generating the index recommendations can take some time. The Index Tuning wizard provides work-in-progress messages that keep you apprised of the progress of the operation.

You can stop the generation at any time. The Index Tuning wizard deletes all the created files in preparation for the next generation.

Once the recommendations are generated, use this screen to choose the index recommendations you want to implement. The Details button provides additional information about each recommendation. To activate the Details button, select a recommendation.

The Next button displays the implementation screen once the generation has completed.

Index Recommendations - Details

When you click the Details button on the Index Recommendations screen, the Details screen displays. The Details screen provides information about why the Index Tuning wizard made a particular recommendation.

Implementation Summary

This screen allows you to implement and save the index tuning recommendations. The possibilities are:

- Implement recommendations immediately
- Save the recommendations as an implementation script, an analysis report, and an Oracle Expert tuning session. You can save the recommendations in any or all of the formats available.

When you click Finish, the recommendations will be saved and/or implemented according to the choices made.

Part VI

Getting Started with Oracle Tablespace Manager

Oracle Tablespace Manager is an Oracle Enterprise Manager integrated application used to automate tablespace management. Oracle Tablespace Manager is part of Oracle Tuning Pack. Oracle Tuning Pack is an optional set of applications that provides advanced tools for tuning the Oracle environment.

Oracle Enterprise Manager Getting Started with Oracle Tablespace Manager covers Oracle Tablespace Manager, the components of the Oracle Tablespace Manager main window, obtaining an overview of tablespaces and datafiles, monitoring segments in a tablespace, and managing storage in a tablespace.

This part contains the following chapters:

22

Using Oracle Tablespace Manager

This chapter covers how you use Oracle Tablespace Manager, including the following topics:

- Introduction to Oracle Tablespace Manager
- Starting Oracle Tablespace Manager
- Oracle Tablespace Manager Main Window
- Obtaining an Overview of Tablespace Storage
- Obtaining an Overview of Datafiles in a Tablespace
- Monitoring Extents and Segments of a Tablespace
- History Options

Introduction to Oracle Tablespace Manager

Effective management of database space usage is necessary to ensure high database performance. The amount of storage space required by various database structures such as tables and indexes will change as the database changes and grows. An Oracle database grows dynamically, meaning that additional units of data storage will be automatically added to ensure continuous database operation. As data is changed or deleted from the database, space may be reclaimed for use by other database objects. Careful planning and use of storage settings will make this dynamic process as transparent as possible to administrators, without requiring frequent monitoring and adjustment.

When Oracle space management is required, it generally involves reorganizing table segments or rebuilding indexes to increase performance or reclaim unused space. The Oracle Tablespace Manager can be used to detect space management problems and automatically reorganize objects to maximize effective database space

usage. Oracle Tablespace Manager can also be used to automatically update database object statistics for maximum Oracle Cost-based Optimizer performance.

Starting Oracle Tablespace Manager

You can start Oracle Tablespace Manager from the console or as a stand-alone application. If you start Oracle Tablespace Manager as a stand-alone application, the Oracle Management Server (OMS) must be running before you can submit any jobs to reorganize or analyze objects from Oracle Tablespace Manager.

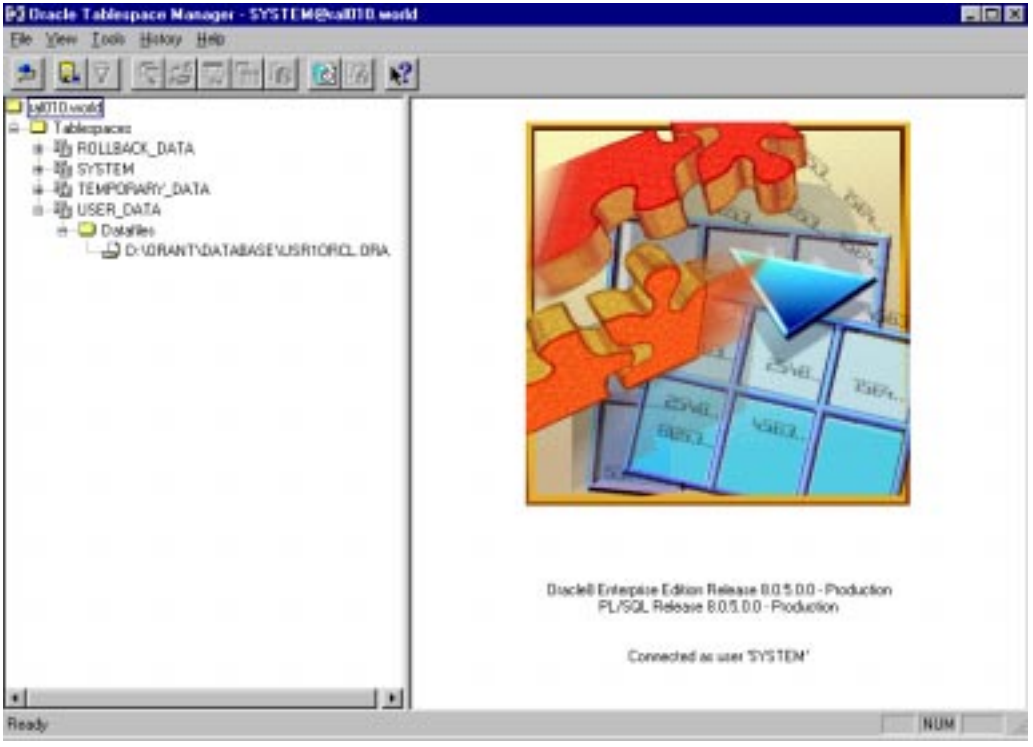
When you start Oracle Tablespace Manager, a login screen prompts you for your username, password, service (the destination database you are interested in monitoring) and connection type. You will be prompted for this information when you start Oracle Tablespace Manager from the console or as a stand-alone application.

Note: You will not be prompted to supply username, password, and service information if Preferred Credentials are set in the console. Refer to the *Oracle Enterprise Manager Administrator's Guide* for more information on setting Preferred Credentials.

Oracle Tablespace Manager Main Window

After you start Oracle Tablespace Manager, the main window is displayed. Figure 22–1 shows the Oracle Tablespace Manager main window.

Figure 22–1 Oracle Tablespace Manager Main Window



The Oracle Tablespace Manager main window includes the following components:

- Title bar
- Toolbar
- Status bar
- Menu bar
- Main window

Title Bar

The title bar of the Oracle Tablespace Manager main window displays the name of the application and the name of the database instance to which a connection has been made.

Toolbar

The Oracle Tablespace Manager toolbar includes icons that enable you to perform the following menu commands:

- Change Database
- Refresh
- Analyze Objects
- Deallocate Unused Space
- Reorganize Objects
- Coalesce Free Extents
- Proactive Problem Detection
- Tablespace Job History
- Proactive Problem Detection History
- Help

Status Bar

The Oracle Tablespace Manager status bar displays information about the current operation on the left.

Main Display

As Figure 22–1 shows, the Oracle Tablespace Manager main window includes a tree list in the left pane of the window. The Oracle Tablespace Manager tree list operates like the Oracle Enterprise Manager tree list, except that it only provides information about the tablespaces of the database instance.

When you start Oracle Tablespace Manager and the main window appears, the top container in the tree list shows the database instance being monitored. The tablespace instance containers are also displayed.

For more information on how a tree list is populated, see ”Navigator” in the *Oracle Enterprise Manager Administrator’s Guide*.

Menu Bar

The Oracle Tablespace Manager menu bar includes the following menus:

- File
- View
- Tools
- History
- Help

File Menu

The File menu items allow you to change the database connection and exit the Oracle Tablespace Manager application.

The File menu includes the following menu items:

Change Database Connection
Allows you to connect to another database instance.

Enable Roles
Displays the Enable Roles dialog box, from which you can select roles to enable. For more information, see "Overview of Database Tools" in the *Oracle Enterprise Manager Administrator's Guide*.

Exit
Exits the Oracle Tablespace Manager application.

View Menu

The View menu items allow you to change what is displayed in the window. For more information, see "Overview of Database Tools" in the *Oracle Enterprise Manager Administrator's Guide*.

The View menu includes the following menu items:

Refresh
Refreshes the data displayed by Oracle Tablespace Manager.

Expand One Level
Expands the selected container in the Oracle Tablespace Manager tree list by one level of detail.

Collapse Branch
Hides the level(s) of detail below the selected container in the Oracle Tablespace Manager tree list.

Collapse All
Hides all levels of detail below the database container in the Oracle Tablespace Manager tree list.

Toolbar
Shows / hides the toolbar.

Status Bar
Shows / hides the status bar.

Tools Menu

The Tools menu includes the following menu items:

Analyze Objects
Starts the Tablespace Analyzer tool. The Tablespace Analyzer tool submits a job that collects various statistics and stores them in the data dictionary. The tool also validates the structure of selected tables, clusters, indexes, and partitions. You can specify when the job will run.

Deallocate Unused Space
Starts the Tablespace Deallocation tool. This tool allows you to reclaim unused space in objects so that the space can be reused by other objects in the tablespace. You can specify when the job will run.

Reorganize Objects
Starts the Tablespace Reorganization tool. This tool submits a job which reorganizes schema objects and lets you set storage parameters and analysis options. You can specify when the job will run.

Coalesce Free Extents
Joins adjacent free extents in the database.

Proactive Problem Detection
Produces reports showing where chained rows and disorganized indexes may have affected database performance. Select objects and set thresholds to produce customized reports. You can specify when the job will run.

History Menu

The History menu includes the following menu items:

Tablespace Job History

Displays all job histories launched from Tablespace Manager. The Tablespace Job History screen shows the job name, destination database, start time, end time and job status.

> **Note:** The Tablespace job histories and the Oracle Enterprise Management jobs are maintained separately and must be individually managed.

Proactive Problem Detection History

The Proactive Problem Detection History screen shows all reports generated by the Proactive Problem Detection tool. These reports show where continued rows and disorganized indexes have affected your database performance.

Help Menu

The Help menu includes the following menu items:

Contents

Displays an overview of Oracle Tablespace Manager.

Search for Help on

Displays an alphabetical list of Help topics.

Using Help

Displays information about using the Help system.

About Oracle Tablespace Manager

Displays version information for this release of Oracle Tablespace Manager.

Obtaining an Overview of Tablespace Storage

The Oracle Tablespace Manager provides the administrator with a complete picture of the characteristics of all tablespaces associated with a particular Oracle instance. These characteristics include tablespace datafiles and segments, total data blocks, free data blocks and percentage of free blocks available in the tablespace's current storage allocation. The DBA has the option of displaying all segments for a tablespace or all segments for a datafile.

The Oracle Tablespace Manager also provides an allocation map which illustrates the organization of a tablespace's segments. This map displays an overview of the

sequential allocation of space for segment extents within a selected tablespace or datafile.

To obtain an overview of the tablespaces in the database instance, single-click on the tablespaces folder in the left pane. The right pane of the main window displays a multi-column list displaying storage information for each tablespace of the database instance. This list includes the following information:

Name
Tablespace name.

Datafiles
Number of datafiles in the tablespace.

Total Blocks
Total number of blocks in the tablespace.

Free Blocks
Number of free blocks in the tablespace.

% Free
Percentage of total number of blocks in the tablespace that are free blocks. A horizontal bar in the background of this field graphically represents the percentage of free blocks in the tablespace.

 # Adj. Free Extents
Number of adjacent free extents in the tablespace.

Obtaining an Overview of Datafiles in a Tablespace

To obtain an overview of the datafiles in a given tablespace, from the Oracle Tablespace Manager tree list, expand the tablespace you are interested in and single-click on the Datafiles folder of the tablespace of interest to you. The right pane of the main window displays a multi-column list including information for each datafile of this particular tablespace.

Name
Datafile name, including its directory path.

Total Blocks
Total number of blocks in the datafile.

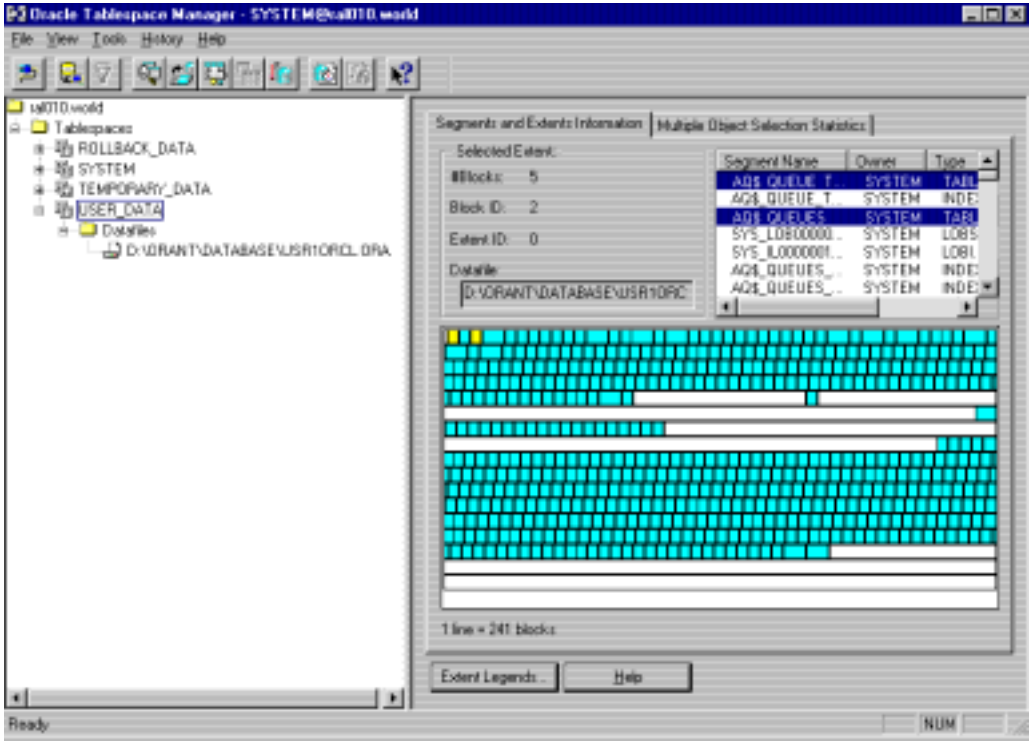
Free Blocks
Number of free blocks in the datafile.

% Free
Percentage of total number of blocks in the datafile that are free blocks. A horizontal bar in the background of this field graphically represents the percentage of free blocks in the datafile.

Monitoring Extents and Segments of a Tablespace

To monitor the segments of a given tablespace, in the tree list, single-click on the tablespace of interest. Figure 22–2 shows an example of the Segments and Extents Information page that displays.

Figure 22–2 Example of the Oracle Tablespace Manager Segments Page



Segments and Extents Information Property Page

The Segments and Extents Information property page includes the following elements:

Selected Extent
Lists the information associated with the extent of the segment selected in the Segment Multi-column list. Extent information includes the following elements:

 # Blocks
Number of blocks in the extent.

Block ID
ID of the first block in the extent.

Extent ID
ID of the extent.

Datafile
Full name and path of the datafile; the physical location of the extent.

Segment Multi-column List
Lists the segments of the tablespace (or datafile) selected in the tree list. When the Segments page first displays, the first segment in the list is automatically selected. If no segments have been allocated for the selected tablespace, the multi-column list is blank.

> **Note:** Click on a segment listed in this list to graphically highlight how the extents in the selected tablespace (or datafile) are allocated for this segment.
>
> Click on the desired column header to sort the list by column.

The multi-column list consists of the following columns:

Segment Name
Name of the segment. Segments are sorted alphabetically by name initially.

Owner
Name of the segment owner. Segments are first sorted alphabetically by segment owner, then alphabetically by segment name.

Type
Segment type. Options include: table, index, cluster, table partition, index partition, rollback, cache, LOB (large object) segment, and LOB index. Segments are sorted alphabetically by segment type, then alphabetically by segment name.

 # Extents
Number of extents within the segment. Segments are sorted first by the number of extents in the segment, then alphabetically by segment name (default).

 # Blocks
Number of blocks in the extent. Segments are sorted according to the number of blocks in the extent (lowest to highest).

Allocation Graphic

Graphically displays space allocation for the selected tablespace (or datafile), showing how space for its segments and extents have been allocated.

White areas denote free space. Colored areas denote used space, as follows:

- Yellow indicates extents allocated to the segment that is selected in the segments list.
- Yellow outlined with a dashed line indicates the extent with the highest block ID.
- Cyan (light blue) indicates extents that are allocated to other (nonselected) segments in the tablespace (or datafile).
- Grey indicates Header blocks.

Black vertical lines separate extents in the space allocation graphic.

> **NOTE:** If you see large **black** sections, it is because there are so many extents in the segment that the colors representing the extents themselves are not visible. To minimize this problem, maximize the size of the Oracle Tablespace Manager main window.

Color Legend

Clicking on the property page button Color Legend displays a quick reference pop-up defining significance of the different colors used in the Allocation Graphic. You can reposition this box to any area of the screen.

- Allocated Extent

  Allocated extents for all segments not selected in the tree list are displayed in cyan (light blue).

- Selected Segment

  Allocated extents of the selected segment in the tree list are displayed in yellow.

- Free Space

  Extents available in the tablespace that have not been assigned to any segment are displayed in white. These extents are available for allocation next time a segment needs to be extended.

- Header Block

  First blocks of a file are used as a header and are grey.

Multiple Object Statistics Page

If you select more than one segment from the Segments multi-column list, the Multiple Object Statistics page appears. This page is only available when you select more than one segment in the segment list. It shows the owner of the selected segments if they belong to the same owner or <Multiple Owners> if they belong to more than one owner. The Multiple Object Selection Statistics page also provides information on the number of objects selected, number of blocks, size (in kilobytes), and totals for the following segment types:

- Tables
- Indexes
- Clusters
- Table Partitions (for Oracle8 databases)
- Index Partitions (for Oracle8 databases)
- Rollbacks
- LOB Segments (Large Objects)
- LOB Indexes (Large Objects)

Analysis Results Property Page

Analysis results produced by the Analyzer tool will appear in the Analysis Results Property page. This screen is located behind the Segments and Extents Property page and appears only when analysis results are available.

Note: After analyzing an object, you will have to reselect the object to display the Analysis Results Property page.

History Options

Oracle Tablespace Manager provides two ways to view history options:

- Tablespace Job History
- Proactive Problem Detection History

Tablespace Job History

The Tablespace Job History screen shows all of the jobs you have launched from within Tablespace Manager. The job name, destination database, start time, end time, and status (completed, failed, started or scheduled) are displayed.

You also have the following options:

Delete Finished

Deletes all completed and failed jobs from the history view screen. *This does not stop pending jobs from running.*

Delete All

Deletes all completed, failed, started and scheduled jobs from the history view screen. *This does not stop pending jobs from running.*

Refresh

Updates list with newly launched jobs or jobs which have a change of status.

Close

Closes the Tablespace Job History screen.

Proactive Problem Detection History

Tablespace Proactive Problem Detection History screen shows all of the reports generated and saved by the Tablespace Proactive Detection tool for the database instance you are currently connected to. The reports are organized by history date (the date that the report was created.)

Next to each history date is a list of fields that further explains the report results:

Object

Name of the analyzed object

Proactive Detection Problem Summary

Brief description of the problem(s). See Recommendations for more information.

Recommendations

Possible remedy to existing problems.

Delete Selected Report

Deletes selected report(s) from the connected repository.

Reorganize Objects

Launches the Tablespace Reorganization tool with any selected objects.

Close

Closes the Tablespace Proactive Problem Detection History screen.

23

Oracle Tablespace Manager Tools

This chapter covers how you use Oracle Tablespace Manager tools. The Tablespace Manager tools provide step-by-step instructions for performing the following tasks:

- Reorganizing Objects in a Tablespace
- Deallocating Unused Space in an Object
- Analyzing Objects in a Tablespace
- Coalescing Adjacent Free Extents in the Database
- Detecting Tablespace Problems Proactively

IMPORTANT: Before using the Tablespace Manager tools, please note the following:

- Oracle Management Server (OMS) must be running prior to launching any jobs. If the console is not running, options will be disabled.
- Check the Oracle Enterprise Manager Console job subsystem to make sure your jobs have completed execution.
- Refresh your view screens to see newly launched jobs or status changes.
- Each Tablespace Manager screen has a set of buttons. Cancel closes the screen without taking any action, Help displays the help content for the screen, Back moves to the previous wizard screen, Next moves to the next wizard screen and Finish starts the tool processes.
- Tablespace Manager tools share some common screens and attributes. Please refer to the Reorganization tool section for details on the common screens and attributes for the other tools.

Reorganizing Objects in a Tablespace

When database space problems are detected, a database reorganization may be necessary. A database reorganization can solve many database space problems, such as free space fragmentation, inefficient block space usage, and stagnant indexes.

A database reorganization also provides the opportunity to change storage parameter settings that can lead to database space problems. For example, inefficient use of object storage settings such as PCTFREE and PCTINCREASE can lead to space problems such as continued (chained and migrated) rows and excessive unused block space. The reorganization process allows you to reset these storage parameters to more effectively handle the space requirements of your database application.

When space reorganization is necessary, the DBA can use the Oracle Tablespace Manager Reorganization tool to automatically correct the problem. The DBA can select a single segment, multiple segments, or the entire tablespace for reorganization, including tables, indexes, and cluster segments.

The Reorganization tool is built on the Oracle Enterprise Manager job system and uses the Oracle Intelligent Agent to control job scheduling, providing automated reorganization of database objects.

To reorganize objects in a tablespace, select the desired segment(s) from the Segments multi-column list on the Segments and Extents Information page.

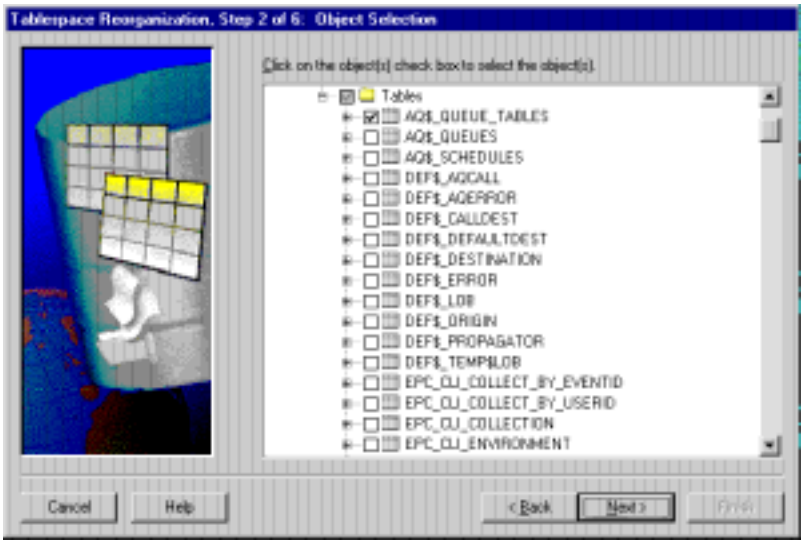
Select the Reorganize Objects option from the Tools menu. The Tablespace Reorganization tool appears.

The Tablespace Reorganization tool prompts you as shown in the following sections.

Note: You cannot reorganize objects owned by 'SYS.'

Selecting Objects

Figure 23–1 Object Selection Page



From the tree list on the Object Selection page, click on the checkbox for each object that you want to reorganize. The selection status of each object is denoted as follows:

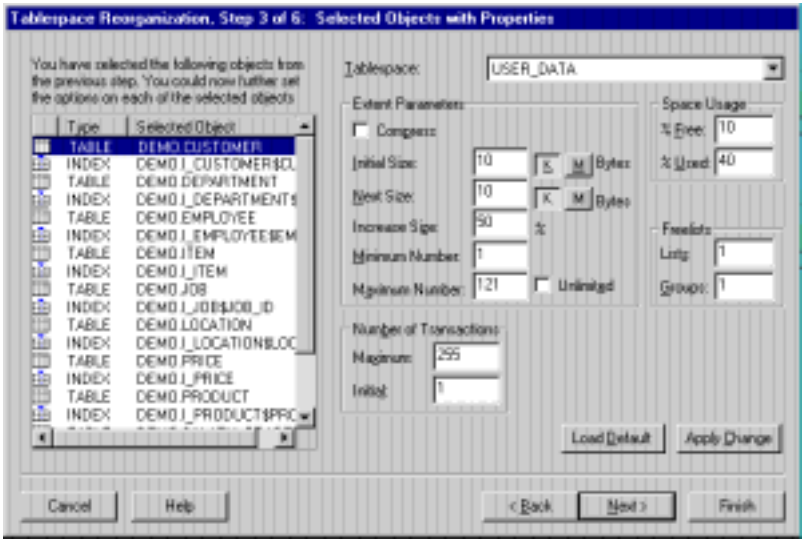
- A check mark in a white box indicates that the object and its children will be reorganized.
- A check mark in a grey box indicates that the object itself is not selected for reorganization but one or more of its children is.
- A check mark in a grey box next to either a folder or schema indicates that some, but not all of its object are selected for reorganization.

> **Note:** Be sure that the objects you select are not currently in use.

To specify parameters, select an object or objects and move on to the next page of the wizard. You can then customize the following storage parameters to suit your needs:

Reorganization Options

Figure 23–2 Reorganization Options Page



This page allows you to set specific options on the objects you already selected in the previous step.

Tablespace

Name of the tablespace containing the segment you want to reorganize.

Extent Parameters

- Compress

  Click this button if you want to reorganize the selected segment by compressing it into a single extent.

- Initial Size

  Size of the first extent of the table segment. Click either the K (Kilobytes) or M (Megabytes) button to specify the size of the measurement.

- Next Size

  Size of the next extent to be allocated to the segment. The default value is the size of ten data blocks. The smallest permissible value is the size of one data

block. Click either the K (kilobytes) or M (megabytes) button to specify the size of the measurement.

- Increase Size

  Percent by which each extent grows (after the second extent) compared to the previous extent.

- Minimum Number

  Total number of extents allocated when a segment is created. The default value is 1. You can enter a value of 1 or greater.

- Maximum

  Total number of extents, including the first, that Oracle can allocate for the segment. You can enter a value of 1 or greater. The default value varies, depending on the database block size.

- Unlimited

  Enabling this option disables the user-specified Maximum field and instead allows the maximum number of extents the system permits. For example, on many systems, the number is approximately 2.1 billion (2,147,483,645). The exact value varies depending on the platform. However, it is not recommended to have a segment with unlimited extents. Rogue transactions containing inserts, updates, or deletes that continue for a long time can continue to create new extents until a disk is full.

Space Usage

- % Free

  Percentage of space in each of the data blocks of the segment that is reserved for future updates to the segment. You can enter a value from 0 to 99.

- % Used

  Minimum percentage of used space that an Oracle database maintains for each data block of the table segment. A block becomes a candidate for row insertions when its used space falls below the % Used value. You can enter a value from 1 to 99. The default value is 40.

Free Lists

- Lists

  Number of free lists for each of the free list groups for the table, cluster, or index. You can enter a value of 1 or more. The default value is 0.

- Groups

  Number of groups of free lists for a table, cluster, or index. You can enter a value of 1 or more. The default value is 1.

Number of Transactions

- Initial

  Initial number of transaction entries allocated within each data block allocated to the table segment. You can enter a value from 1 to 255. The default is 1 for tables. For clusters and indexes, the minimum and default value is 2.

- Maximum

  Maximum number of concurrent transactions that can update a data block allocated to the segment. You can enter a value from 1 to 255.

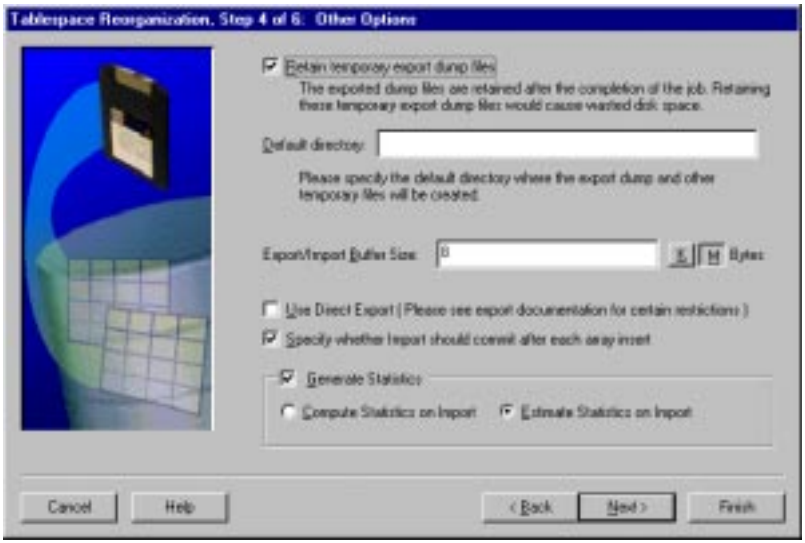
Load Default

Click on this button to assign the default storage parameters to the selected object(s).

Defining Tablespace Reorganization Options

The Other Options page allows you to select reorganization options that meet your tablespace reorganization requirements.

Figure 23–3 Other Options Page



Use the Other Options page to specify the following general options:

Retain export dump and other temporary files after job finishes
Enable this option to save export and temporary files after the reorganization job has executed. You can keep the constraint and object files as an audit trail for auditing purposes or reuse the export file to copy the data or save as a backup.

> **Note:** Oracle recommends that you always retain temporary dump files so that you may revert any changes made to your database.

Directory for export dump and other temporary files
Enter the directory path on the server in which the export file and temporary files (object and constraints files) are to be written. This directory path cannot contain any environment variables.

> **Attention:** If you do not specify a directory path, these files will be written to the directory in which the reorganization job script is run. Be sure the Oracle Intelligent Agent has permission to write to the directory for these files or the reorganization job will fail.

Export/Import Buffer Size
Size of the buffer. Click either the K (kilobytes) or M (megabytes) button to specify the size of the measurement.

Use Direct Export
Move the contents of the buffer cache directly to the output dump file. This method is quicker because no SQL processing is required on the data.

Commit After Array Insert
Based on the size of the buffer, commit after each array of rows has been inserted into the database.

Generate Statistics
Enabling this option executes the Oracle SQL ANALYZE command upon import. It is limited to objects that already had statistics before they were exported.

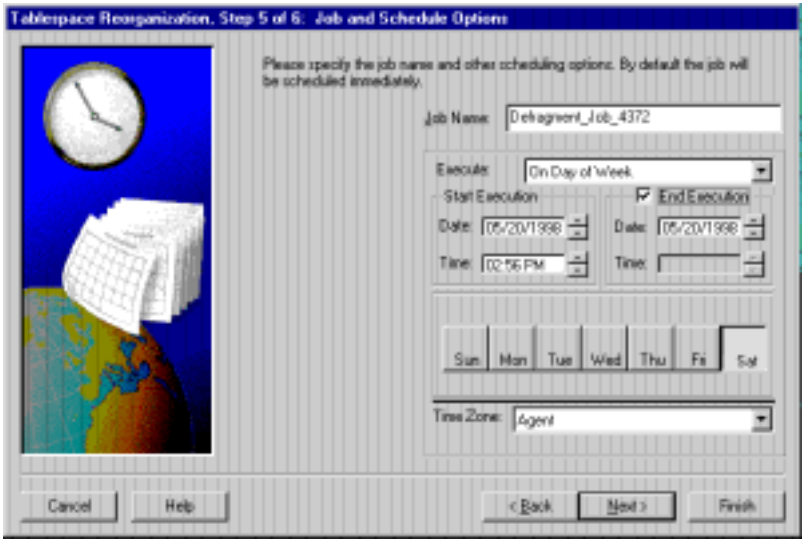
Compute Statistics on Import: Gives the exact statistics for all the segments. This can take a considerable amount of time.

Estimate Statistics on Import: Gives the estimated statistics for all segments. This can take a considerable amount of time.

Defining Job and Schedule Options

The Scheduling Options page allows you to schedule the execution of the reorganization.

Figure 23–4 Job and Schedule Options



Job Name

You can accept the default or use any character for the job name (except dollar sign, apostrophe and ampersand.) For ease of use, keep the job name short and simple.

Execute

Select the frequency with which you want the job executed. The choices are:

- **Immediately**

  Schedules the job as soon as you click the Submit button on the Summary page. The job executes only one time.

- **Once**

  Schedules the job to occur once at the date and time you choose.

- **Interval**

  Allows you to schedule a specific time interval between job executions. The interval can be a combination of hours and minutes, or number of days. Select the value you want to change in the Time field and click on the scroll buttons. You can also type in a new value.

- **On Day of Week**

Allows you to schedule the job on one or multiple days (Sunday, Monday, etc.) of the week. Click on the days of the week in the Date field to select the days on which you want the job to occur.

- **On Day of Month**

  Allows you to schedule the job on one or multiple days (1-31) of the month. Click on the dates of the month in the Date field to select the dates on which you want the job to occur.

Start Execution

Choose the first date and time that you want the job executed. This is the starting time for any job scheduled on an interval.

- **Date**

  Select the month, day, or year in the Date field and click on the scroll buttons to change the value. You can also type in new values.

- **Time**

  Select the hour, minute, or AM/PM in the Time field and click on the scroll buttons to change the value. You can also type in new values.

End Execution

Choose the last date and time that you want the job executed. This option does not apply if you chose the Immediately or Once execution options.

- **Date**

  Select the month, day, or year in the Date field and click on the scroll buttons to change the value. You can also type in new values.

- **Time**

  Select the hour, minute, or AM/PM in the Time field and click on the scroll buttons to change the value. You can also type in new values.

Time Zone

Select the time zone from the pull-down list. The choices are:

- **Agent**

  The agent schedules the job execution at each destination based on the actual system time of each agent. Jobs are not necessarily run simultaneously.

- **Console**

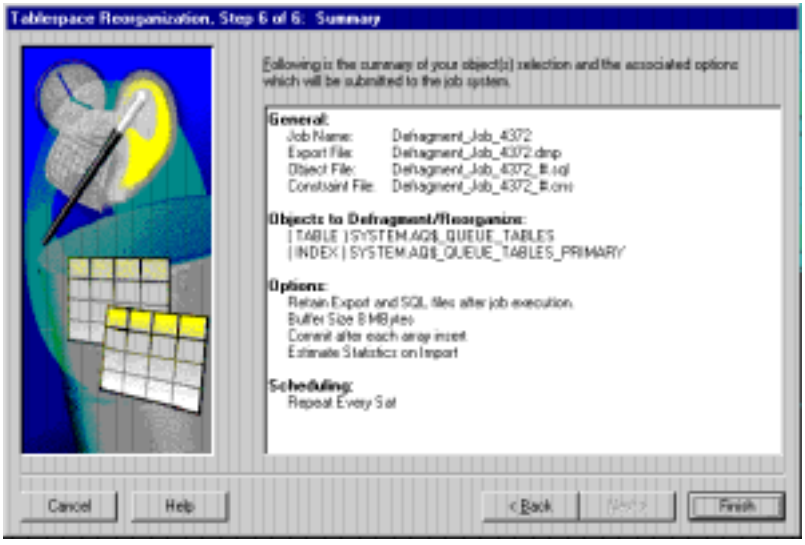
The agent schedules the job execution simultaneously on all destinations based on the system time of the console.

- **GMT**

  The agent schedules the job execution simultaneously on all destinations based on Greenwich mean time.

Viewing the Tablespace Reorganization Summary Page

Figure 23–5 Summary Page



The Tablespace Reorganization Summary page summarizes all the information you entered while using the Tablespace Reorganization tool.

The summarized properties are:

- General
- Objects to Reorganize
- Options
- Scheduling

Once you click Finish, Oracle Tablespace Manager submits the job to the Oracle Enterprise Manager job system. When the job is completed, Oracle Tablespace Manager receives notification and adds the job output and status to Tablespace Job History. Examine the Job History to verify that the job completed successfully.

Deallocating Unused Space in an Object

Use the Tablespace Deallocation wizard to deallocate unused space within an object. Select the desired object(s) in the Segments multi-column list located on the Segments and Extents Information page.

Start the Tablespace Deallocation wizard by selecting the Deallocate Unused Space option in the Tools menu. The wizard guides you through the space deallocation job definition process, job scheduling, and finally provides a job summary so that you can check the job settings before submission to the Oracle Enterprise Manager job system.

Default Options

The second page of the Deallocation wizard allows you to specify the default value for the amount of unused space to be reserved in each segment. The default options will apply to all objects you select. Space can be specified in units of megabytes or kilobytes.

Object Selection

From the tree list on the Object Selection page, click on the checkbox for each object that you want to deallocate space. See "Selecting Objects" on page 23-3 for more details.

Deallocation Options

After specifying default value for the reserved amount of unused space, you can also set objects to reclaim unused space. You can also specify the reserved amount of unused space per object. This setting overrides the default setting specified in the Default Options screen.

Job and Schedule Options

After setting the deallocation job parameters, you can now schedule the job for execution by the Oracle Enterprise Manager job system. See "Defining Job and Schedule Options" on page 23-8.

Deallocation Summary Page

The Deallocation Summary page summarizes all the information you entered. See "Viewing the Tablespace Reorganization Summary Page" on page 23-11 for more details.

Once you click Finish, Oracle Tablespace Manager submits the job to the Oracle Enterprise Manager job system. Oracle Tablespace Manager will attempt to deallocate storage for the object you specified. When the job completes, Oracle Tablespace Manager receives notification and adds the job output and status to the Tablespace Job History. Examine the Job History screen to verify that the job(s) completed successfully.

Analyzing Objects in a Tablespace

The Oracle Tablespace Manager Analyzer tool provides a simple solution for keeping database statistics up to date. If database statistics are out-dated or missing, the Cost-based Optimizer, which uses database statistics to determine the best execution plan for each SQL summary, will be unable to perform its task and database performance will be severely impacted.

The Analyzer tool also provides automated ANALYZE of one or more objects for a selected schema or for the entire database. ANALYZE can be run immediately or scheduled to run at a later time. ANALYZE can also be scheduled to run on a periodic basis. The Oracle Tablespace Analyzer is built on the Oracle Enterprise Manager job system and uses the Oracle Intelligent Agent to control job scheduling, providing automated ANALYZE of selected database objects.

The Tablespace Analyzer wizard provides step-by-step instructions for submitting a job that analyzes various statistics and validates structure for selected tables, clusters, indexes, and partitions. When the job completes, an Analysis Results tab appears beside the Segments and Extents Property tab and displays the analysis output.

Defining Default Options

On the Default Options wizard page, you define the default analysis options. The default options will apply to all objects you select.

Note: You may have to reselect the analyzed object to see the results page.

Default Options

The default options you can choose are as follows:

- Compute Statistics

  Computes exact statistics about the analyzed object and stores them in the data dictionary.

- Delete Statistics

  Deletes any statistics about the analyzed object that are currently stored in the data dictionary.

- Estimate Statistics

  Estimates statistics about the analyzed object and stores them in the data dictionary. You can specify a number of rows or a percentage of rows to use in the calculation.

- Validate Structure

  Verifies the integrity of the structure of an index, table, or cluster. Enable the Cascade option to validate the structure of all indexes on the cluster's tables, including the cluster index.

- List Migrated and Continued Rows

  Lists rows that have moved from one data block to another (migrated) or rows that are contained in more than one block (chained). Migrated and Continued Rows can cause excessive I/O, so you may want to identify them in order to eliminate them.

Object Selection

From the tree list on the Object Selection page, select the objects you want to analyze. See "Selecting Objects" on page 23-3 for more details.

Analysis Options

After selecting objects to analyze, you can also perform additional analyses on specific objects, beyond the default options set earlier. The same analysis options available for the default options definition are also available at the object level.

Job and Schedule Options

The Scheduling Options page allows you to schedule the execution of the analysis. Refer to "Defining Job and Schedule Options" on page 23-8 for details.

Analyzer Summary Page

The Tablespace Analyzer Summary page summarizes all the information you entered while using the Tablespace Analyzer tool. See"Viewing the Tablespace Reorganization Summary Page" on page 23-11 for more details.

Once you click Finish, the job is submitted and the results appear in the Analysis Results page which is located behind the Segments and Extents page.

Coalescing Adjacent Free Extents in the Database

If you administer an active database, you may want to join or coalesce adjacent free blocks in the database on a frequent basis.

To use Oracle Tablespace Manager to join adjacent free blocks in the database, take the following steps:

1. In the Oracle Tablespace Manager tree list, click on the tablespace or datafile of interest.
2. When the Segments and Extents Information page displays, select the segment for which you want to join adjacent free blocks.
3. Choose Coalesce Free Extents from the Tools menu. The free blocks are automatically joined. No job is created.

Each group of adjacent free blocks in the space allocation graphic of the Segments and Extents page should now appear as a single free block.

Detecting Tablespace Problems Proactively

The Tablespace Proactive Problem Detection tool allows you to perform space problem detection on selected database objects. Proactive Problem Detection wizards takes you through a step-by-step process and produces a report detailing space problems such as continued rows (which include both chained and migrated rows) and disorganized indexes. Each report is customized to the objects you specify.

Common Database Space Problems

The Oracle Tablespace Manager Proactive Problem Detection wizard allows you to detect the following database space problems:

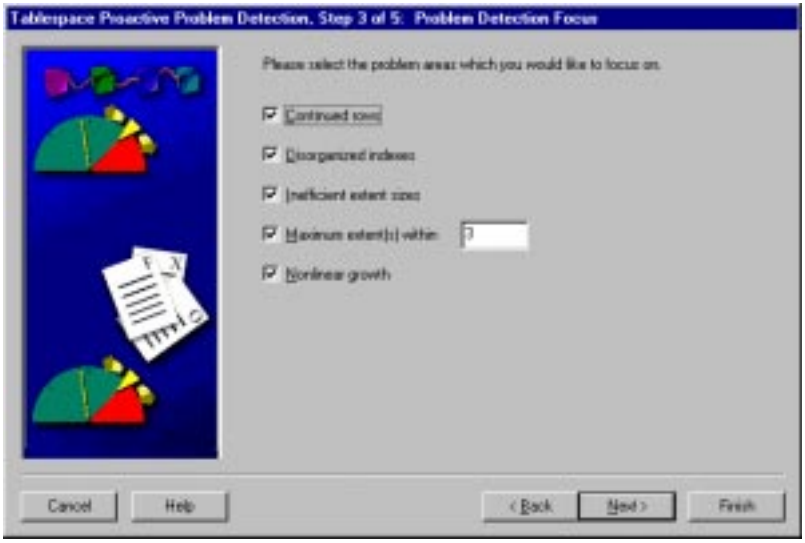
- **Continued rows**--When a table row becomes too large to fit into a single data block, the row continues to the next datablock causing row fragmentation. Row

fragmentation can cause unnecessary I/Os which leads to longer database response time. Space reorganization can eliminate this fragmentation or continuation. To avoid further fragmentation, the table's pctfree value can be increased.

- **Disorganized indexes--**As indexed values are inserted or deleted, the amount of unusable space within the index increases. Because the index contains a mixture of data and empty areas, scans of the index are less efficient. Rebuilding the index will eliminate unused index space.
- **Maximum extent sizes--**The MAXEXTENTS segment storage parameter specifies the maximum number of extents that can be allocated to the segment. Once a segment has filled the maximum number of extents, any row insertion will fail. Oracle Tablespace Manager will detect when a segment is nearing its MAXEXTENT allocation. Once detected, you should increase the value of the segment's MAXEXTENTS storage parameter or rebuild the segment with a larger extent size.
- **Non-linear Growth**--A segment whose extent space allocation is growing at a non-linear rate can produce free space fragmentation and other problems. Consider modifying the segment's PCTINCREASE storage parameter to specify PCTINCREASE 0.

Use the Proactive Problem Detection tool to analyze the tablespace and produce reports which will highlight the existence of these problems.

Figure 23–6 Problem Detection Focus



Selecting Objects for Proactive Problem Detection

The Tablespace Proactive Problem Detection Object Selection screen allows you to select specific objects to detect. See "Selecting Objects" on page 23-3 for more details.

Proactive Problem Detection Focus

An efficiently organized database enables users to do their jobs quickly, without interruption. Detecting and preventing database space problems ensures that the database will perform as efficiently as possible. Correcting existing space problems, by applying repair techniques to the smallest subsets of the database, will save time and expense.

Setting Job and Schedule Options

After selecting the problems you wish to look for, you can now schedule the job for execution by the Oracle Enterprise Manager job system. You have the option to launch the report immediately by selecting *Generate a report without submitting a job* or *Submit a job* to run the report with the Tablespace job subsystem. By default, the tool schedules to run the job with the job subsystem.

Viewing the Tablespace Proactive Problem Detection Summary Page

The Tablespace Proactive Problem Detection Summary page summarizes all the information you entered while using the Tablespace Proactive Problem Detection tool. See "Viewing the Tablespace Reorganization Summary Page" on page 23-11 for more details.

Glossary

access methods

One of the categories of application tuning. During access methods tuning, Oracle Expert determines what indexes are needed and generates the SQL statements to create, modify, and delete indexes as appropriate. Oracle Expert ensures that the indexing strategy is consistent with that of the Oracle cost-based optimizer. Oracle Expert addresses sorted (B-tree) and bit-mapped indexes. See also SQL reuse.

advanced rules

Rules that you would rarely want to view or edit, for example, low-level constants that are factored into the algorithms used by the Oracle Expert rules. See also default rules and rules.

analysis

The process by which Oracle Expert examines the tuning data collected for a database and generates tuning recommendations.

Analysis report

A report that describes the Oracle Expert tuning recommendations. It provides a detailed explanation of the data Oracle Expert evaluated, how Oracle Expert interpreted the data, and any risks associated with implementing the recommendations.

ANALYZE command

The ANALYZE command collects and stores table and index statistics which are essential for the efficient operation of the cost-based optimizer.

If an object has not been analyzed at least once and the CHOOSE optimizer method is the default, Oracle SQL Analyze will use the rule-based optimization method when generating an explain plan.

The frequency with which you analyze the objects depends on the rate of change within the objects. If you analyze a table, then its associated tables are automatically analyzed as well.

This command may lock portions of the database while it collects its statistics.

Application

The workload category at the top of the workload hierarchy. See also Request.

application tuning session

A focused tuning session in which one or more application tuning categories (access methods or SQL reuse) are selected for one or more applications. See also access methods andSQL reuse.

attribute

A characteristic of a data object. You can view and edit attributes for objects using the Edit dialog box.

base index tuning

The part of access method tuning where Oracle Expert scans schemas selected for tuning for evidence of implicit search operations, such as constraints or views. If a constraint or view is found, Oracle Expert determines whether an index is necessary to improve performance when the constraint or view is executed.

bitmap indexes

Bitmap indexing provides the same functionality as regular indexes, but uses a different internal representation, which makes it very fast and space efficient. They work best when each key references many records, such as employee gender.

bind variables

Bind variables allow variables within a SQL statement, such as data values or search keys, to be defined as parameters of the SQL statement. This approach allows SQL statements to be re-executed without re-parsing the statement.

While generating an explain plan for a SQL statement, Oracle SQL Analyze will request sample values for bind variables. Although it is not required to enter a value, the values will help Oracle SQL Analyze evaluate the statement and its environment to develop the best optimization plan.

business characteristics

A group of parameters whose values provide useful information to Oracle Expert for tuning your database environment. Examples of business characteristics for which you can supply values are the classes of workload (such as multipurpose, OLTP, or Data Warehousing) and the tolerance for database downtime. Oracle Expert takes business characteristic values into account when making tuning recommendations. (Previously referred to as control parameters.)

cardinality

Can refer to either table cardinality (the number of rows in a table) or column cardinality (the number of distinct values in a column of a table).

category

See tuning category.

chained rows

When a row no longer fits within the data block its row header is stored in, that row may store the remainder of its data in a different block or set of blocks. Such a row is said to be *chained*, and chained rows may be cause poor performance due to the increased number of blocks that must be read in order to read a single row.

choose

When you select Choose (optimizer's choice), the optimizer determines whether at least one object has been analyzed (using the ANALYZE command). If at least one object has been analyzed, the optimizer uses cost-based optimization for throughput. If not, the optimizer uses rule-based optimization.

For more information, see *Oracle Server Tuning.*

class

See collection class.

collection

The gathering of data that Oracle Expert analyzes to make tuning session recommendations.

collection class

A category of information that Oracle Expert collects and analyzes. The collection classes are Database, Instance, Schema, Environment, and Workload.

column cardinality

The number of distinct values in a column of a table.

compact view

Compact views are representations of the explain plan that illustrate the join methodologies selected by the Oracle optimizer.

comprehensive tuning session

A tuning session in which you have selected all the tuning categories at the same time. During analysis for a comprehensive tuning session, Oracle Expert generates all the tuning recommendations it is capable of (for all areas of your database environment).

cost-based optimization for response time

Cost-based optimization considers statistical information about the volume and distribution of data within tables and indexes before determining the execution path for a statement. It then tries to choose the execution path that has the least "cost."

In this context, "cost" is a measurement of many factors, including the amount of computer resources (I/O and CPU consumption, for example), and the time to complete the execution.

When optimizing for response time (also known as *cost all* optimizing), the optimizer chooses to execute the statement in a way that most efficiently retrieves the first row of data.

Note:	At least one referenced object (table or index) must be analyzed before cost-based optimization can be performed.

For more information, see “Understanding the Oracle Optimizer” on page 3-23, and *Oracle Server Tuning*.

cost-based optimization for throughput

Cost-based optimization considers statistical information about the volume and distribution of data within tables and indexes before determining the execution path for a statement. It then tries to chose the execution path that has the least "cost."

In this context, "cost" is a measurement of many factors, including the amount of computer resources (I/O and CPU consumption, for example), and the time to complete the execution.

When optimizing for throughput, the optimizer chooses to execute the statement in a way that most efficiently process all rows specified.

Note:	At least one referenced object (table or index) must be analyzed before cost-based optimization can be performed.

For more information, see “Understanding the Oracle Optimizer” on page 3-23, and *Oracle Server Tuning*.

cursor

A cursor is a handle (a name or pointer) for the memory associated with a specific statement. (The Oracle Call Interface, OCI, refers to these as statement handles.) For example, in precompiler application development, a cursor is a named resource available to a program and can be specifically used for the parsing of SQL statements embedded within the application.

cycle

The interval of time during which a sequence of a recurring succession of events is completed.

database object

A database object represents the database against which the active SQL statement is executed.

When the database object is selected, you can view some of the characteristics of the database, but you cannot edit them.

database tuning

A process that involves tasks such as balancing competing database resources so that important applications get the resources they need, identifying and eliminating resource bottlenecks, and optimizing the use of existing resources in the database environment.

Data Definition Statements (DDL)

DDL statements define, maintain, and drop objects when they are no longer needed. DDL statements also include statements that permit a user to grant other users the privileges, or rights, to access the database and specific objects within the database.

data dictionary

The data dictionary is a read-only set of tables that provides information about its associated database. For example, a data dictionary can provide the following information:

- the names of Oracle users
- privileges and roles each user has been granted
- names of schema objects (tables, views, snapshots, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers and so on)
- data distributions
- information about integrity constraints
- default values for columns
- how much space has been allocated for, and is currently used by, objects in a database
- auditing information
- other general database information

The data dictionary is structured in tables and views, just like other database data. Because the data dictionary is read-only, users can issue only queries (SELECT statements) against the tables and views of the data dictionary.

Data Manipulation Statements (DML)

DML statements manipulate the database's data. For example, querying, inserting, updating, and deleting rows of a table are all DML operations; locking a table or view and examining the execution plan of an SQL statement are also DML operations.

Data Warehousing

Data Warehousing applications typically use complex queries against large, mostly read-only database tables. See also multipurpose applications and OLTP applications.

Decision Support System (DSS)

Decision support or data warehousing applications distill large amounts of information into understandable reports. Typically, decision support applications perform queries on the large amount of data gathered from OLTP applications. The key goals of a decision support system are response time, accuracy, and availability.

An example of a decision support system is a marketing tool that determines the buying patterns of consumers based on information gathered from demographic studies. The demographic data is assembled and entered into the system, and the marketing staff queries this data to determine which items sell best in which locations. This report helps to decide which items to purchase and market in the various locations.

The key goals of a data warehousing system are response time, accuracy, and availability.

default rules

A portion of the rules that make up the extensive Oracle Expert knowledge base. Each Oracle Expert rule exists at the Oracle Expert system level. Rules at the Oracle Expert system level are called default rules, and they have default values. A copy of a default rule can exist at one or more object levels. Oracle Expert allows you to view some default rules. Oracle Expert allows you to change the default value of any default rule it allows you to view. See also advanced rules and rules.

degree of parallelism

The number of parallel server processes associated with a single operation is known as the *degree of parallelism*.

The degree of parallelism is specified at the statement level (with hints or the PARALLEL clause), at the table or index level (in the table's or index's definition), or by default based on the number of disks or CPUs.

Note that the degree of parallelism applies directly only to intra-operation parallel-ism. If inter-operation parallelism is possible, the total number of parallel server processes for a statement can be twice the specified degree of parallelism. No more than two operations can be performed simultaneously.

edit data

The act of viewing, modifying, adding, or deleting collected data that appears on the View / Edit page of the tuning session window.

emphasis

The statistical weight given to the frequency value of an element when computing its relative importance. The emphasis is based on either a user-supplied importance, calculated frequency, or the physical I / O count. By default, Oracle Expert gives more statistical weight to frequency. This can be changed by modifying the "Primary workload analysis style" and "Secondary workload analysis style" Workload user rules.

environment

The physical resources of the database, for example, the system the database runs on and the logical devices used by the database. Data about a database's environment is collected in the Environment collection class.

explain plan

An explain plan is an access path determined by the query optimizer.

Oracle SQL Analyze displays explain plans based on different optimizer modes, and helps you "step through" the execution path.

For more information, see "Understanding Explain Plans" on page 2-8.

explain plan object

An explain plan node represents explain plans generated for a specific SQL statement.

Selecting an explain plan object displays the explain plan in the SQL Text window.

filtered workload

A workload collection of statements that Expert has decided are applicable to tuning the scope specified for that tuning session.

focused tuning session

A tuning session in which you have not selected all of the Oracle Expert tuning categories at the same time. After an analysis is performed in a focused tuning session, Oracle Expert generates tuning recommendations for the selected tuning categories.

frequency

The number of times a workload element is executed by the workload element above it in the workload hierarchy. For example, for a workload Request, the frequency is the number of times the Request is executed during a workload Transaction.

hints

Hints are instructions in the SQL code that direct the Oracle optimizer to use specific methods when creating an explain plan.

implementation

The process during which Oracle Expert generates implementation files and implementation scripts that can help you put the Oracle Expert tuning recommendations into effect. See also implementation files and implementation scripts.

implementation files

Files (such as parameter files for instances) that Oracle Expert generates to help you put its tuning recommendations into effect.

implementation scripts

Scripts generated by Oracle Expert that you can run to put the Oracle Expert tuning recommendations into effect.

importance

The importance (or priority) value assigned to a workload element relative to the other workload elements at the same level of the workload hierarchy. For example, given two workload Applications, where one is a production, high-availability application and the other is a maintenance application, you would assign the first a higher importance value than the second. Oracle Expert uses this value to ensure

that higher priority Applications and Requests are favored over less important ones during the optimization process. The range of importance values is from 1 to 9999, with 1 being the lowest importance. See also relative importance.

index rebuild detection

One of the categories of access methods tuning in which Oracle Expert analyzes indexes to identify those that suffer from index stagnation and should be rebuilt to enhance performance. See also index stagnation, optimal index use, and access methods.

index stagnation

A condition in which the amount of unusable space within the index increases to a level that negatively impacts performance.

initial configuration tuning session

A tuning session in which Oracle Expert is used to help configure a new database.

initialization parameter object

Initialization parameter objects, located in the Navigator window, represent specific instances of the database they are connected to on the navigation tree.

By selecting an initialization parameter object, you can edit the parameters displayed for the purpose of simulating a SQL tuning environment other than that of the current database.

instance tuning session

A focused tuning session in which one or more categories of instance tuning (SGA parameters, I/O parameters, parallel query parameters, Oracle Parallel Server parameters, operating system-specific parameters, or sort parameters) are selected for one or more instances. See also I/O parameters, parallel query parameters, SGA parameters, Oracle Parallel Server parameters, operating system-specific parameters, and sort parameters.

instantiation

Making a copy of a default rule at an object level. This makes the object the owner of the copy of the rule. Oracle Expert uses an object's values for instantiated rules during an analysis. See also default rules and rules.

I/O parameters

These instance parameters affect the throughput or distribution of I/O for the instance. Examples of these parameters include the checkpoint_process and db_file_

multiblock_read_count parameters. These parameters can be selected as a category to be tuned as part of an instance tuning session. See also instance tuning session, parallel query parameters, SGA parameters, sort parameters, Oracle Parallel Server parameters, and operating system-specific parameters.

join strategy optimization

Joins allow rows of two or more tables to be merged, usually based on common key values. Improving the strategy the optimizer uses to join tables together while executing a SQL statement is a primary method of enhancing performance.

In Oracle SQL Analyze, you can affect join order optimization by adding hints or by analyzing the statement with the Tuning Wizard.

multipurpose applications

Mutlipurpose applications are characterized by one or few users doing long transactions where response times are not critical. Typically, multipurpose or batch applications are executed overnight with the restriction that they must complete before morning. See also Data Warehousing and OLTP applications.

node

The node is the node name where an instance is running.

object details

Object details provide information about data objects referenced by a SQL statement.

These details include information about tables, clusters, and indexes related to an object, and can be used to further understand the performance of associated explain plans.

OLTP applications

OLTP (online transaction processing) applications typically use simple queries that require quick response time against tables containing a mix of read and write requests. See also multipurpose applications and Data Warehousing.

Online Transaction Processing (OLTP)

Online transaction processing (OLTP) applications are high-throughput, insert/update intensive systems. These systems are characterized by constantly growing large volumes of data that several hundred users access concurrently. Typical OLTP applications are airline reservation systems, order-entry applications, and banking applications. The key goals of an OLTP application are availability.

operating system-specific parameters

These instance parameters affect performance and are specific to certain hardware platforms. Examples of these parameters include the async_write and the db_ writers parameters. These parameters can be selected as a category to be tuned as part of an instance tuning session. See also instance tuning session, I/O parameters, SGA parameters, parallel query parameters, sort parameters, and Oracle Parallel Server parameters.

optimal index use

One of the categories of access methods tuning in which Oracle Expert uses base index and workload analysis tuning to determine whether new indexes should be created or current indexes modified to enhance performance. See alsoindex rebuild detection, base index tuning, and access methods.

optimizer

The Oracle optimizer determines the execution path taken to perform the commands of a SQL statement.

There are four modes of optimization available:

- Rule
- Cost-based optimization for response time
- Cost-based optimization for throughput
- Choose

The mode used for optimization is set via the OPTIMIZER_MODE parameter in each instance's init.ora file. The mode can be overridden by using the ALTER SESSION SET OPTIMIZER_GOAL command or by adding hints to a statement.

Oracle Parallel Server parameters

These parameters influence the performance or configuration of the parallel server environment. Selecting the Oracle Parallel Server parameters category for tuning is relevant only if the Oracle Parallel Server option is installed. These parameters can be selected as a category to be tuned as part of an instance tuning session. See also instance tuning session, I/O parameters, SGA parameters, sort parameters, parallel query parameters, and operating system-specific parameters.

parallel query parameters

These instance parameters are specific to the parallel query behavior for the instance. Examples of these parameters include the parallel_min_servers and

parallel_max_servers parameters. Selecting the parallel query parameters category for tuning is relevant only if the parallel query option is installed. These parameters can be selected as a category to be tuned as part of an instance tuning session. See also instance tuning session, I/O parameters, SGA parameters, sort parameters, Oracle Parallel Server parameters, and operating system-specific parameters.

parallel server parameters

See Oracle Parallel Server parameters.

placement

A category of structure tuning. Oracle Expert makes recommendations about placement of structures, such as segment partitioning. Oracle Expert may recommend separating segments into different tablespaces to minimize tablespace free space fragmentation and maximize administrative flexibility. See also sizing.

Program Global Area (PGA)

The Program Global Area (PGA) is a memory region that contains data and control information for a single process (server or background). Consequently, the PGA is referred to as the *Program Global Area* or the *Process Global Area*.

rank

Oracle Expert considers the elements in the highest category of the workload hierarchy (Applications) to be most important, and elements in each of the lower categories to be proportionately less important. Rank is ordered sequence where 1 is most important.

Recommendation Summary report

A report that lists the Oracle Expert tuning recommendations. It provides the reasons why you received the recommendations.

relative importance

A value that ranks a workload element's importance compared to all the other workload elements. Oracle Expert computes the relative importance for each workload element. The factors that Oracle Expert takes into account when computing a relative importance value for a workload element are the element's workload category (Application or Request), importance value, and frequency value. See also importance.

reports

Information that Oracle Expert can generate about the collected data for a tuning session (Session Data report) or the recommendations it has generated (Analysis report or Recommendation Summary report). See also Analysis report, Session Data report, and Recommendation Summary report.

repository

A schema in an Oracle database that stores the data associated with each Oracle Expert tuning session.

An Oracle SQL Analyze repository stores the information from SQL tuning sessions, including:

- information necessary to construct the Navigator tree for that session
- database and session parameters
- SQL statements
- explain plans and their related statistics

Object details are not saved in the repository.

You can return to a saved repository at any time to continue a tuning session.

representative workload

A workload that includes a representative set of the SQL statements that execute during a period for which you want Oracle Expert to optimize database performance. You would typically provide valid importance values for the elements in a representative workload to allow Oracle Expert to optimize performance for the statements with the highest relative importance.

Request

A SQL statement. Requests are the elements at the lowest level of the workload hierarchy. See also Application.

rule-based optimization

When you choose rule-based optimization, the Oracle optimizer executes SQL statements based on a set of syntactical rules and the rankings of various access paths. It does not consider statistical information relating to the volume and distribution of data within tables and indexes.

In most cases, cost-based optimization is the preferred approach for new applications and for data warehousing applications because it supports new and enhanced features.

In general, the rule-based optimizer takes the following approach:

4. For each table in the WHERE clause, the optimizer considers and ranks every possible access path.
5. The optimizer selects the access path with the lowest rank.
6. The optimizer ranks every possible access path for each remaining table.
7. The optimizer selects the access path with the lowest rank.
8. This process is continued until every table is joined.

For more information, see "Understanding Explain Plans" on page 2-8 and *Oracle Server Tuning*.

rules

Pieces of knowledge used by Oracle Expert to analyze collected data for a tuning session. Oracle Expert allows you to view some of its rules, and you can modify the value of any rule you view. Modifying rule values may change the behavior of Oracle Expert during an analysis and also its recommendations. See also advanced rules and default rules.

rules-of-thumb

Rules-of-thumb are a set of guidelines that, when applied to a SQL statement, improve the syntax of the statement and make it more efficient. These guidelines have been developed over a number of years by the database experts at the Oracle Corporation.

You can use Oracle SQL Analyze to apply rules-of-thumb to your SQL statements by analyzing them through the Tuning Wizard.

To learn more about rules-of-thumb and how they're applied, see "Understanding Rules-of-Thumb" on page 2-16.

service

The database name Oracle Expert will use for the Database object and the Instance object on the View / Edit page of the tuning session window

scope

A setting that determines the categories of database tuning Oracle Expert will address for a given tuning session. You choose the scope when you create a new tuning session, and you can modify the scope for a tuning session. You can choose one or more tuning categories from one or more of the tuning types: instance tuning, application tuning, and structure tuning. See also instance tuning session, application tuning session, and structure tuning session.

session

See tuning session.

Session Data report

A report that describes all the collected data for an Oracle Expert tuning session. You can read this report before starting an analysis of the collected data to confirm that you provided Oracle Expert with all the data it expects for your tuning session.

SGA parameters

These instance parameters affect the total size of the instance's System Global Area (SGA). The appropriate setting of these parameters results in efficient utilization of memory and prevents reparsing SQL statements except when necessary. Examples of these parameters include the db_block_buffers and shared_pool_reserved_size parameters. These parameters can be tuned as part of an instance tuning session. See also instance tuning session, I/O parameters, parallel query parameters, sort parameters, Oracle Parallel Server parameters, and operating system-specific parameters.

sizing

A category of structure tuning. Oracle Expert makes recommendations for sizing various segments in order to improve space usage and performance. See also placement.

sort parameters

These parameters influence how the Oracle Server performs sort operations on behalf of the user. Examples of these parameters include the sort_direct_write and sort_area_retained_size parameters. These parameters can be selected as a category to be tuned as part of an instance tuning session. See also instance tuning session, I/O parameters, parallel query parameters, sort parameters, Oracle Parallel Server parameters, and operating system-specific parameters.

SQL history

The complete set of application SQL data and statistics that are executed within the database environment. The statements can be added to the SQL History from a variety of sources, including the SQL cache, Oracle Trace, and .XDL files. The SQL History provides a consistent source of SQL data for all of the Tuning Pack applications that make tuning recommendations. The SQL History also prevents the Tuning Pack applications from impacting the production database environment when SQL information is needed for an analysis.

SQL object

SQL objects, located in the Navigator window, represent specific SQL statements that you can tune against the database session they are connected to.

SQL objects may be created, copied or deleted. The statements within them may be edited in a number of ways.

Note, however, that the objects become read-only after an explain plan is generated for the statement. To proceed with further editing, you must create a copy of the node using **SQL=>Create Like** command.

SQL reuse

One of the categories of application tuning. The Oracle Server maintains only one copy of a distinct SQL statement within the library cache to maximize memory and minimize redundant parsing and validating. To effectively use this feature, you must write duplicate SQL statements using identical structure and form (two statements are considered the same only when they match character for character, including spaces and punctuation). Oracle Expert compares your workload statements to determine if any can be rewritten to take advantage of the cache behavior and reports its findings. See also access methods.

star queries

One type of data warehouse design is known as a “star” schema. This typically consists of one or more very large “fact” tables and a number of much smaller “dimension” or reference tables. A star query is one that joins several of the dimension tables, usually by predicates in the query, to one of the fact tables.

Oracle cost-based optimization recognizes star queries and generates efficient execution plans for them; indeed, you must use cost-based optimization to get efficient star query execution. To enable cost-based optimization, simply ANALYZE your tables and be sure that the Optimizer mode is set to its default value of CHOOSE.

star transformation

The star transformation is a cost-based query transformation aimed at executing star queries efficiently. Whereas the star optimization works well for schemas with a small number of dimensions and dense fact tables, the star transformation may be considered as an alternative if any of the following holds true:

- The number of dimensions is large.
- The fact table is sparse.
- There are queries where not all dimension tables have constraining predicates.

The star transformation does not rely on computing a Cartesian product of the dimension tables, which makes it better suited for cases where fact table sparsity and/or a large number of dimensions would lead to a large Cartesian product with few rows having actual matches in the fact table. In addition, rather than relying on concatenated indexes, the star transformation is based on combining bitmap indexes on individual fact table columns.

The transformation can thus choose to combine indexes corresponding precisely to the constrained dimensions. There is no need to create many concatenated indexes where the different column orders match different patterns of constrained dimensions in different queries.

The star transformation works by generating new subqueries that can be used to drive a bitmap index access path for the fact table.

structured query language (SQL)

Structured Query Language, or SQL, is an industry-standard programming language developed specifically to support access to relational databases.

The general characteristics of the SQL language are:

- It supports operations on sets of data, as opposed to row-by-row processing.
- It can access data independently of the data's physical location
- It is non-procedural. In other words, it only describes the data to be retrieved, not the method by which the data is to be retrieved.

Oracle7 and Oracle8 databases support SQL and PL/SQL, a supplementary implementation of SQL. Oracle SQL Analyze currently supports only standard SQL.

structure tuning session

A focused tuning session in which one or more categories of structure tuning are selected for one or more database structures. See also placement and sizing.

system data

Data about the machine on which an instance runs. You need to provide Oracle Expert with data such as the memory allocation and CPU utilization of the machine on which the instance runs.

System Global Area

The System Global Area (SGA) is a shared memory region that contains data and control information for one Oracle instance. An SGA and the Oracle background processes constitute an Oracle instance.

Oracle allocates the system global area when an instance starts and deallocates it when the instance shuts down. Each instance has its own system global area. Users currently connected to an Oracle Server share the data in the system global area. For optimal performance, the entire system global area should be as large as possible (while still fitting in real memory) to store as much data in memory as possible and minimize disk I/O.

The information stored within the system global area is divided into several types of memory structures, including the database buffers, redo log buffer, and the shared pool. These areas have fixed sizes and are created during instance startup.

table cardinality

The number of rows in a table.

TopSQL

TopSQL is an integrated function of Oracle SQL Analyze that lets you measure the resources a SQL statement consumes. Using these statistics, you can determine which statements consume the most resources and select them for tuning.

TopSQL takes all the SQL statements and statistics from V$SQLAREA. The V$SQLAREA view lists statistics on shared SQL areas and contains one row per SQL string. It provides statistics on SQL statements that are in memory, parsed, and ready for execution or that might have been executed already.

TopSQL object

The TopSQL node, located in the Navigator window, represents the TopSQL session available for the database object it is connected to.

A separate TopSQL object is created each time you link to another database.

tuning category

One of the specific tuning areas (such as SGA or sizing) that Oracle Expert addresses for a tuning type. Two or more tuning categories are associated with each Oracle Expert tuning type. See also tuning type.

tuning scope

See scope, comprehensive tuning session, focused tuning session, and initial configuration tuning session.

tuning session

The framework within which Oracle Expert performs its tuning activities. The steps in a tuning session include creating the session, specifying the session's tuning scope, collecting data, editing data, analyzing data, generating recommendations, validating recommendations, and implementing recommendations.

tuning type

One of the general areas of the database environment that Oracle Expert can tune. The three tuning types are instance tuning, application tuning, and structure tuning. Two or more tuning categories are associated with each tuning type. See also tuning category.

type

See tuning type.

V$SQLAREA

Oracle TopSQL uses information from V$SQLAREA to determine the resources used by a specific SQL statement.

V$SQLAREA is a view of an instance as a whole, and records statistics either since the startup of the instance or the current values, which remain constant until altered by some need to reallocate SGA space. Statistics are available for SQL statements that are in memory, parsed, and ready for execution.

For more information, see *Oracle Server Tuning* and the *Oracle Server Administrator's Guide.*

validation

The process of verifying, for a collected object, that other objects referenced by or dependent upon the collected object have also been collected. If other objects referenced by or dependent upon a collected object have not been collected, the collected object is marked as invalid. The international No symbol is used to mark invalid objects on the View / Edit page of the tuning session window.

what-if scenario

A tuning session in which you modify the values of collected data such as disks, memory, CPU, or cardinality to a value you expect will be correct in the future. This type of tuning allows you to determine the effect of changes on a database before the changes are actually made.

workload

Data that describes to Oracle Expert the nature, frequency, and importance of SQL statements that access a database. You should collect a representative workload for each tuning session that requires a workload. Workload data is displayed hierarchically on the View / Edit page of the tuning session window. See also representative workload, Application, and Request.

.XDL file

An Oracle Expert Definition Language file. Oracle Expert creates an .XDL file when it exports database, instance, schema, environment, or workload data.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X