

---

# **Oracle9i New Features Overview**

**Volume 1 • Studnet Guide**

---

D11741GC11  
Production 1.1  
July 2001  
D33524

**ORACLE®**

## Authors

Joel Goodman  
Lex de Haan  
Stefan Lindblad  
Ulrike Schwinn

## Technical Reviewers

David Austin  
Harald van Breederode  
Arturo Gutierrez  
Michael Moller  
Sander Rekvelde  
Jean-Francois Verrier

**Copyright © Oracle Corporation, 2001. All rights reserved.**

This documentation contains proprietary information of Oracle Corporation. It is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited. If this documentation is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

### Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to restrictions for commercial computer software and shall be deemed to be Restricted Rights software under Federal law, as set forth in subparagraph (c)(1)(ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

This material or any portion of it may not be copied in any form or by any means without the express prior written permission of Oracle Corporation. Any other copying is a violation of copyright law and may result in civil and/or criminal penalties.

If this documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with "Restricted Rights," as defined in FAR 52.227-14, Rights in Data-General, including Alternate III (June 1987).

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to Education Products, Oracle Corporation, 500 Oracle Parkway, Box SB-6, Redwood Shores, CA 94065. Oracle Corporation does not warrant that this document is error-free.

SQL\*Loader, SQL\*Net, SQL\*Plus, Net8, Oracle Call Interface, Oracle7, Oracle8, Oracle 8i, Developer/2000, Developer/2000 Forms, Designer/2000, Oracle Enterprise Manager, Oracle Parallel Server, PL/SQL, Pro\*C, Pro\*C/C++, and Trusted Oracle are trademarks or registered trademarks of Oracle Corporation.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

# Contents

## Introduction

- Oracle Server History 1-2
- Lesson Overview 1-8

## 1 Managing Initialization Parameters

- Objectives 1-2
- Server Parameter File (SPFILE) 1-3
- SPFILEs 1-4
- Creating an SPFILE 1-5
- Changing Parameter Values within an SPFILE 1-6
- STARTUP Command Behavior 1-7
- Viewing the Parameter Settings 1-8
- Exporting an SPFILE 1-9
- Migrating to a SPFILE 1-11
- Shared Initialization Parameter Files 1-12
- Shared SPFILE 1-13
- Summary 1-14

## 2 Managing Memory

- Objectives 2-2
- Automated SQL Execution Memory Management 2-3
- PGA Memory Management 2-4
- Enabling Automated SQL Execution Memory Management 2-5
- New Statistics and Columns 2-6
- Example 2-7
- Dynamic SGA 2-8
- Units of Allocation 2-9
- Cache Size and Number of Buffers 2-10
- Dynamic Shared and Large Pools 2-11
- Dynamic Buffer Cache 2-12
- Increasing the Size of a Component's SGA Memory Area 2-13
- New and Deprecated Buffer Cache Parameters 2-14
- Dynamic Buffer Cache Advisory Parameter 2-16
- New View to Support Buffer Cache Advisory Information 2-17
- Summary 2-18

## 3 Managing Data Files and Tablespaces

- Objectives 3-2
- Oracle Managed Files 3-3
- Benefits of Using Oracle Managed Files 3-4

Defining Oracle Managed Files	3-5
OMF Filename Structure	3-6
Managing Control Files with OMF	3-7
Managing Online Redo Log Files with OMF	3-8
Managing Tablespaces with OMF	3-10
The Delete Data Files Feature	3-12
Default Temporary Tablespaces	3-13
Creating Default Temporary Tablespaces	3-14
Altering Default Temporary Tablespaces	3-15
Restrictions on Default Temporary Tablespaces	3-16
Automatic Undo Management	3-17
The Automatic Undo Management Concept	3-18
Creating an Undo Tablespace	3-19
Altering an Undo Tablespace	3-21
Dropping an Undo Tablespace	3-22
Switching Undo Tablespaces	3-23
New Parameters to Support Automatic Undo Management	3-24
New View to Support Automatic Undo Management	3-26
Multiple Block Size Support	3-27
Creating Standard Block Size Tablespaces	3-28
Creating a Tablespace with a Nonstandard Block Size	3-29
Multiple Block Sizing Rules	3-30
Nonstandard Block Sizes	3-31
Multiple Block Size Support	3-32
Sizing the KEEP and RECYCLE Buffer Caches	3-33
Automatic Segment Free Space Management in Locally Managed Tablespaces	3-35
Automatic Segment Free Space Management	3-36
Automatic Segment Free Space Management at Work	3-37
Creating a Segment with Automatic Segment Free Space Management	3-38
Modifications to Views	3-39
Migration and Compatibility	3-41
Summary	3-42

#### **4 Managing Resumable Space Allocation**

Objectives	4-2
Resumable Space Allocation	4-3
Enabling and Disabling Resumable Space Allocation Mode	4-6
Altering Resumable Space Allocation	4-7

Life Cycle of Resumable Space Allocation 4-8  
Which Operations are Resumable? 4-9  
Detecting Suspended Statements: The `ALERT.log` File 4-10  
Detecting Suspended Statements: The After Suspend System Event 4-11  
Monitoring Resumable Space Allocation 4-12  
Summary 4-14

## **5 Database Resource Manager Enhancements**

Objectives 5-2  
Database Resource Manager Overview 5-3  
Active Session Pool 5-4  
Active Session Pool Mechanism 5-5  
Active Session Pool Parameters 5-6  
Setting the Active Session Pool 5-7  
Maximum Estimated Execution Time 5-8  
Automatic Consumer Group Switching 5-9  
Undo Quota 5-10  
Changing the Undo Quota 5-12  
Modified Views to Support Database Resource Manager Extensions 5-13  
Summary 5-14

## **6 Enterprise Manager Enhancements**

Objectives 6-2  
New Console Look-and-Feel 6-3  
Launching Enterprise Manager Console 6-4  
Console Launched in Standalone Mode 6-5  
Connections Using OMS 6-6  
Standalone Connection Benefits 6-7  
Standalone Connection Restrictions 6-8  
Standalone Repository 6-9  
Management Regions 6-10  
Creating a Management Region 6-12  
Enterprise Manager Support for Oracle9i Database Features 6-13  
Creating the SPFILE 6-14  
Changing Parameters 6-15  
Startup Using the SPFILE 6-16  
Undo Tablespace Support 6-17  
Undo Tab 6-18  
Buffer Cache Size Advice View 6-19  
Creating Default Temporary Tablespace 6-20  
Mean Time to Recovery 6-21  
Backup and Recovery Enhancements 6-22  
Advanced Queuing 6-23  
HTML Database Reports 6-24  
Database Configuration Report 6-25

- User-Defined Events 6-26
- User-Defined Event Tests 6-27
- Event Handler 6-29
- Summary 6-30

## **7 Performance Enhancements**

- Objectives 7-2
- Optimizer Cost Model Enhancements 7-3
- New Statistics-Gathering Estimates 7-5
- Gathering System Statistics 7-7
- Example of Gathering System Statistics 7-8
- New First Rows Optimization 7-9
- Outline Editing Overview 7-10
- Attributes Which Can Be Edited 7-12
- Outline Cloning 7-13
- Outline Administration and Security 7-14
- Configuration Parameters 7-15
- CREATE OUTLINE Syntax Changes 7-16
- Examples of Outline Cloning 7-17
- Example of Manual Outline Editing 7-19
- Cursor Sharing Enhancements 7-21
- The CURSOR\_SHARING Parameter 7-23
- Cached Execution Plans 7-24
- New View to Support Cached Execution Plans 7-25
- New plan\_hash\_value Column in V\$SQL 7-26
- Identifying Unused Indexes 7-27
- Enabling and Disabling Index Usage Monitoring 7-28
- The V\$OBJECT\_USAGE View 7-29
- Skip Scanning of Indexes 7-31
- Example of Skip Scanning 7-32
- Example of Skip Scanning: Search for SWITZERLAND 7-33
- Bitmap Join Indexes (BJIs) 7-41
- Advantages and Disadvantages of BJIs 7-43
- BJI: A More Complex Example 7-44
- Restriction on BJIs 7-45
- Summary 7-46

## **8 Partitioning Enhancements**

- Objectives 8-2
- List Partitioning Overview and Benefits 8-3
- Example of List Partitioning 8-4
- List Partitioning Pruning 8-5
- ALTER TABLE ADD PARTITION Example 8-6
- ALTER TABLE MERGE PARTITION Example 8-8
- ALTER TABLE MODIFY PARTITION ADD VALUES Example 8-10

- ALTER TABLE MODIFY PARTITION DROP VALUES Example 8-11
- ALTER TABLE SPLIT PARTITION Example 8-13
- List Partitioning Usage 8-15
- Maintaining Global Indexes 8-17
- Performance Considerations: Updating versus Rebuilding an Index 8-20
- Parallel Direct-Load Inserts Versus Partitioned Tables Enhancements 8-21
- DML Intra-Partition Parallelism 8-22
- Summary 8-24

## **9 Data Warehouse Enhancements**

- Objectives 9-2
- The Traditional Way: Fragmented Information Supply Chain 9-3
- The New Way: Oracle9i 9-4
- Full Extraction and Transportation Methods 9-7
- Overview of Oracle Change Data Capture 9-8
- Oracle Change Data Capture Overview 9-9
- The Publish and Subscribe Model 9-10
- Components and Terminology for Synchronous Change Data Capture 9-12
- Data Dictionary Views Supporting CDC 9-14
- Classical ETL Process 9-15
- Pipelined Data Transformation in Oracle9i 9-17
- External Tables 9-18
- Applications of External Tables 9-19
- Example of Defining External Tables 9-20
- Data Dictionary Information for External Tables 9-21
- Transformations with Pipelined Table Functions 9-22
- Creating Table Functions 9-23
- Transformations Using Table Functions 9-24
- Advantages of PL/SQL Table Functions 9-25
- Transformation Using Multitable INSERT Statements 9-26
- Transformation Using Multi-table Insert Statements 9-27
- Advantages of Multitable INSERT Statements 9-28
- Unconditional INSERT 9-29
- Pivoting INSERT 9-30
- Conditional ALL INSERT 9-31
- Conditional FIRST INSERT 9-32
- MERGE Statements 9-33
- Applications of MERGE Statements 9-34
- Example of Using the MERGE Statement in Data Warehousing 9-35
- Enhancements to Materialized Views 9-36
- Summary 9-38

## **10 Migrating and Upgrading the Database**

- Objectives 10-2
- Upgrading from Oracle8 to Oracle9i 10-3
- Migrating Specific Components 10-4
- Migrating from Server Manager to SQL\*Plus 10-5
- Migrating from LONG to LOB Data Type 10-7
- Example of LONG to LOB Migration 10-9
- Recommendations 10-10
- Obsolete Initialization Parameters (Since Version 8.1.7) 10-11
- Renamed Initialization Parameters 10-12
- New Initialization Parameters in Oracle9i 10-13
- New Data Types 10-16
- Summary 10-17

## **11 ISO SQL:1999 Enhancements**

- Objectives 11-2
- SQL:1999 Enhancements Overview 11-3
- SQL:1999 Joins 11-4
- Cross Joins 11-5
- Natural Joins 11-6
- Natural Join Example 11-7
- Equijoins and the USING Clause 11-8
- USING Clause Example 11-9
- Join Predicates and the ON Clause 11-10
- Three-Way Joins with the ON Clause 11-11
- Outer Joins 11-12
- Outer Join Example 11-13
- CASE Expression Enhancements 11-14
- Simple CASE Expressions 11-15
- Searched CASE Expression 11-16
- NULLIF and COALESCE 11-17
- Scalar Subqueries 11-18
- Scalar Subquery Example 11-19
- Grouping Sets 11-20
- Grouping Sets Example 11-21
- Composite Columns 11-23
- Concatenated Groupings 11-25
- Summary 11-26

## **12 SQL, PL/SQL, and Object Enhancements**

- Objectives 12-2
- Oracle9i Type System 12-3
- Datetime and Interval Data Types 12-4
- TIMESTAMP WITH TIME ZONE 12-5
- INTERVAL YEAR TO MONTH Data Type 12-6



- TIME\_ZONE Database Parameter 12-7
- Daylight Saving Time 12-8
- Unicode Support 12-9
- Migrating to Oracle9i NCHAR Data Types 12-10
- SQL Function Support for Implicit Type Conversion 12-11
- Overview of Native Compilation of PL/SQL 12-12
- PL/SQL Compilation 12-13
- New Parameters for Native Compilation 12-14
- Steps for Enabling Native Compilation 12-16
- Benefits of Native Compilation of PL/SQL 12-17
- Object Enhancements 12-19
- Multilevel Collection Types 12-20
- Tables with Multilevel Collections 12-21
- Examples of Multilevel Collections 12-22
- Creating Tables with Multilevel Collections 12-23
- Nested Tables in IOT Storage 12-25
- Model Completeness: Inheritance 12-26
- FINAL and NOT FINAL Types 12-27
- Substitutability 12-28
- Object View Hierarchies 12-29
- Privileges on Object Types 12-31
- Utilities 12-32
- Object Type Evolution 12-33
- Propagating Type Changes 12-34
- Examples Of Schema Evolution with No Dependent Tables 12-35
- Propagating Changes to Dependent Tables 12-36
- Summary 12-37

### **13 Database Availability Enhancements**

- Objectives 13-2
- Reducing Unplanned Downtime: Overview 13-3
- Minimal I/O Recovery 13-4
- Fast-Start Time-Based Recovery Limit: Overview 13-7
- Fast-Start Time-Based Recovery Limit 13-8
- Changes to Previous Parameters 13-12
- Changes to V\$INSTANCE\_RECOVERY 13-13
- Reducing Planned Downtime 13-14
- Online Index Rebuilding 13-15
- Index-Organized Table Availability Enhancements 13-18
- Online Operations on IOTs: Operations on Secondary Indexes 13-19
- Online Operations on IOTs: Online Moves 13-21
- Online Operations on IOTs: Updates of Block References 13-22
- Online Table Reorganization 13-23
- Online Table Reorganization: Applications 13-24
- Online Table Reorganization 13-25
- Online Table Reorganization: Syntax 13-26

- Online Table Reorganization: Synchronizing and Aborting 13-28
- Online Table Reorganization: Example 13-29
- Online Table Reorganization: Limitations 13-31
- Online ANALYZE VALIDATE STRUCTURE 13-33
- Overview of Oracle Flashback 13-34
- Oracle Flashback 13-35
- Oracle Flashback Versus Oracle LogMiner 13-44
- Summary 13-45

## **14 Real Application Clusters**

- Objectives 14-2
- Real Application Clusters Overview 14-3
- Benefits of Real Application Clusters 14-4
- Global Cache Service 14-5
- Dynamic Resource Remastering 14-6
- Global Cache Service Resource Modes 14-7
- Global Cache Service Resource Roles 14-8
- Block Transfers 14-9
- Block Transfer Examples 14-10
- Block Transfer Example Setup 14-11
- Consistent Read Block Transfer 14-12
- Cache Fusion Block Transfer 14-13
- Benefits of Cache Fusion 14-15
- High Availability Features 14-16
- Real Application Clusters Guard 14-17
- Real Application Clusters Guard Architecture 14-18
- Monitors and Failover 14-19
- Initialization Parameters 14-20
- Replaced Initialization Parameters 14-21
- Obsolete Global Cache Parameters 14-22
- Summary 14-23

## **15 Oracle9i LogMiner Enhancements**

- Objectives 15-2
- LogMiner New Features 15-3
- DDL Statement Support 15-4
- DDL Statement Support Example 15-5
- Data Dictionary Access 15-6
- Dictionary Information in the Redo Logs 15-7
- Using an Online Data Dictionary 15-8
- DDL Tracking in the Dictionary 15-9
- Dictionary Staleness Detection 15-10
- Skipping Past Log Corruptions 15-11
- Displaying Information Only for Committed Transactions 15-12
- Primary Key Information 15-13

- LogMiner Restrictions 15-15
- LogMiner Viewer 15-16
- LogMiner Views 15-22
- Summary 15-23

## **16 Oracle 9i Data Guard**

- Objectives 16-2
- Data Protection Components 16-3
- Oracle9i Data Guard Overview 16-4
- Oracle9i Data Guard Architecture 16-6
- Data Guard Broker and Data Guard Manager 16-7
- Oracle9i Data Guard Broker 16-8
- Oracle9i Data Guard Manager 16-10
- Physical Standby Database 16-12
- Physical Standby Database: Graceful Switchover 16-13
- Physical Standby Database: Additional New Features 16-16
- Log Transport Services 16-17
- Log Transport Services: Automated Online Log Transport 16-18
- Log Transport Services: Synchronous Data Copy Mode 16-19
- Log Transport Services: Immediate Data Copy Mode 16-20
- Log Transport Services: Writing Archive Logs to Disk 16-22
- Log Transport Services: Failure Resolution Policies 16-24
- Manageability Enhancements: Automatic Recovery of Log Gaps 16-25
- Manageability Enhancements: Updating the Standby at a Lag 16-26
- Manageability Enhancements: Single Log Stream 16-28
- Manageability Enhancements: Other New Capabilities 16-30
- Summary 16-32

## **17 Recovery Manager (RMAN) Enhancements**

- Objectives 17-2
- Persistent Configuration Parameters 17-3
- Retention Policies 17-4
- CONFIGURE RETENTION POLICY 17-5
- Automatic Channel Allocation 17-6
- CONFIGURE CHANNEL 17-7
- Other Configuration Commands 17-9
- CONFIGURE SNAPSHOT CONTROLFILE and CONFIGURE AUXNAME 17-10
- Long-Term Backups 17-11
- Mirrored Backups 17-12
- Backup File Optimization 17-13
- Restartable Backups 17-14
- Archive Log Backup 17-15
- Backup-Set Backups 17-16
- Restore File Optimization and Restartable Restore Operations 17-17
- Archive Log Failover and Automatic Log Switch 17-18

Backup Piece Failover	17-19
Block Media Recovery (BMR)	17-20
Reducing the Mean Time to Recover	17-21
Other BMR Benefits	17-22
The Recovery Manager Interface	17-23
Recovery Manager Interface	17-24
Test Recovery	17-26
Test Recovery: Allowing Corruptions of Data Blocks	17-28
Test Recovery	17-29
REPORT OBSOLETE	17-31
REPORT NEED BACKUP	17-33
Command Unification	17-34
LIST Command: New Syntax	17-35
New SHOW Command	17-36
CROSSCHECK Automatic Location	17-37
Other RMAN Enhancements	17-38
Summary	17-39

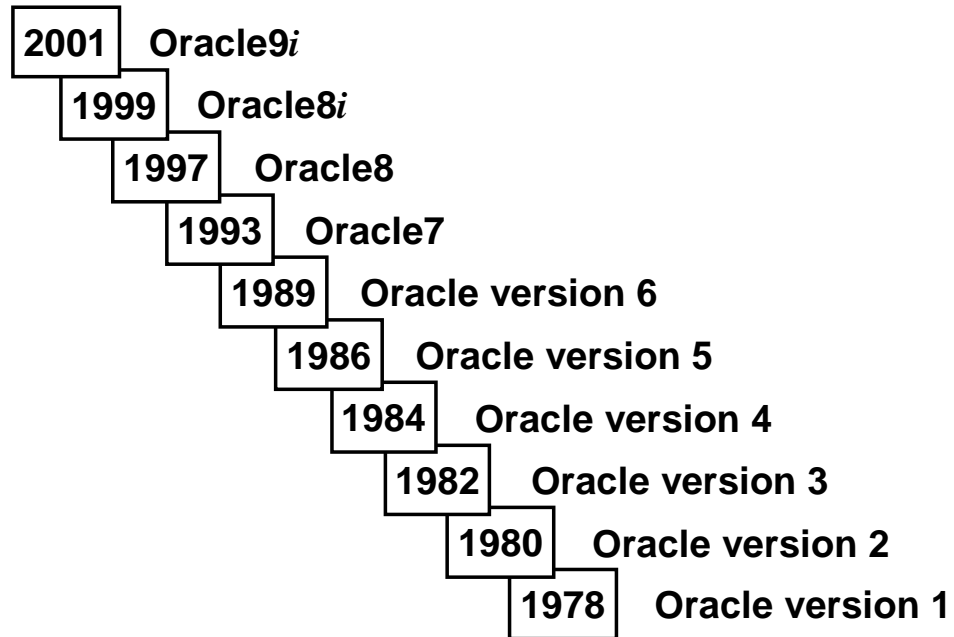
# **Oracle9i New Features Overview**

## **Introduction**

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# Oracle Server History



ORACLE

# Oracle Server History

- **1978: Oracle version 1**
  - Ran on PDP-11 under RSX; 128 KB memory
  - Written in assembly language
  - Separated Oracle code and user code
- **1980: Oracle version 2**
  - Written in PDP-11 assembly language
  - Ran on VAX/VMS in compatibility mode

ORACLE

# Oracle Server History

- **1982: Oracle version 3**
  - Written in C: Portable source code
  - Retained split architecture
  - Introduced transactions
- **1984: Oracle version 4**
  - Introduced read consistency
  - Ported to many platforms
  - First interoperability between PC and server

ORACLE



# Oracle Server History

- **1986: Oracle version 5**
  - True client-server
  - VAX-cluster support
  - Distributed queries
- **1989: Oracle version 6**
  - OLTP performance enhancements
  - Online backup and recovery
  - Row-level locking, PL/SQL procedures
  - Parallel Server

ORACLE

# Oracle Server History

- **1993: Oracle7**
  - Shared SQL
  - Data warehousing
  - Parallel everything
  - Advanced replication
- **1997: Oracle8**
  - Object-relational database
  - Partitioning

ORACLE

# Oracle Server History

- **1999: Oracle8i**
  - Java in the database
  - Partitioning enhancements
  - Data warehousing enhancements
  - Summary management
  - Oracle Internet Directory
- **2001: Oracle9i**

ORACLE

# Lesson Overview

1. Initialization parameters
2. Memory enhancements
3. Data files and tablespaces
4. Resumable statements
5. Resource management enhancements
6. Enterprise manager enhancements
7. Performance enhancements
8. Partitioning enhancements
9. Extraction, Transportation, and Loading (ETL)

ORACLE

## **Lesson Overview**

- 10. Migration and upgrading**
- 11. ISO SQL: 1999 enhancements**
- 12. SQL, PL/SQL, and objects enhancements**
- 13. High availability enhancements**
- 14. Real Application Clusters**
- 15. LogMiner enhancements**
- 16. Oracle9i Data Guard and standby databases**
- 17. Recovery Manager enhancements**

ORACLE



# 1

## Managing Initialization Parameters

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# Objectives

**After completing this lesson, you should be able to:**

- **Create a server parameter file (SPFILE)**
- **Change initialization parameter values**
- **View parameter settings**
- **Export an SPFILE**
- **Migrate to an SPFILE**

ORACLE



## Server Parameter File (SPFILE)

- **A new parameter file called an SPFILE has been introduced to store instance parameters persistently by the server.**
- **The benefits of SPFILEs include the ability to:**
  - **Make changes to parameter values persist across shutdown and startup**
  - **Have Recovery Manager support the backup of the initialization parameter file**
  - **Have all members of a Real Application Clusters database use a single parameter file**

ORACLE

1-3

Copyright © Oracle Corporation, 2001. All rights reserved.

### Server Parameter File

The SPFILE adds new functionality to the server for storing and managing its initialization parameters persistently in a server-side disk file. With the SPFILE, you can make persistent changes to the database: Changes which are unaffected by shutdowns and startups. This eliminates the need to update initialization parameters manually in order to preserve the changes effected by ALTER SYSTEM statements.

Recovery Manager can support the backup of the initialization parameter file, because the SPFILE resides on the server side. One SPFILE can be shared by all members of a Real Application Clusters database.

# SPFILEs

- An SPFILE is a small binary file.
- It is maintained by the Oracle server.
- It always resides on the server side.
- Front-end tools, such as Enterprise Manager and SQL\*Plus, do not need to specify a parameter file if using the default SPFILE.
- The default server-side location and name is platform specific; for example, for UNIX:
  - `$ORACLE_HOME/dbs` is the location
  - `spfile<sid>.ora` is the name

ORACLE

1-4

Copyright © Oracle Corporation, 2001. All rights reserved.

## SPFILE Characteristics

The SPFILE is a small binary file which cannot be browsed or edited using a text editor. The file is not meant to be modified manually, and it must always reside on the server side.

The default file location is specific to the operating system (for example, `$ORACLE_HOME/dbs` for UNIX), and the file has a default name such as `spfile<sid>.ora`. After the file is created, it is maintained by the Oracle server.

# Creating an SPFILE

An SPFILE is created from an `INIT.ORA` file using the `CREATE SPFILE` command:

```
CREATE SPFILE[ = 'SPFILE-NAME' ]  
FROM PFILE[ = 'PFILE-NAME' ]
```

Where:

- **SPFILE-NAME:** Name of the SPFILE to be created
- **PFILE-NAME:** Name of the `INIT.ORA` file to be used

Example:

```
SQL> CREATE SPFILE FROM PFILE;
```

ORACLE

1-5

Copyright © Oracle Corporation, 2001. All rights reserved.

## Creating an SPFILE

An SPFILE is created from an `INIT.ORA` file using the `CREATE SPFILE` command. This statement can be executed when the instance is in any state: `IDLE`, `NOMOUNT`, `MOUNT`, or `OPEN`.

Then, the next time you start up the instance, you can direct the database to read the initialization parameters from the SPFILE.

You must have the `SYSDBA` or `SYSOPER` system privilege to execute the `CREATE SPFILE` command.

## Changing Parameter Values within an SPFILE

- Use the **ALTER SYSTEM** command to specify whether the change is temporary or persistent.

```
ALTER SYSTEM SET parameter = value  
[SCOPE = MEMORY|SPFILE|BOTH]
```

- **Example:**

```
SQL> ALTER SYSTEM SET SORT_AREA_SIZE= 1048576  
      2 COMMENT='temporary change' SCOPE=spfile;  
SQL> show parameter sort_area_size  
NAME                                TYPE                                VALUE  
-----  
sort_area_size integer                524288
```

ORACLE

1-6

Copyright © Oracle Corporation, 2001. All rights reserved.

### Parameter Values within an SPFILE

Using a SPFILE overcomes the traditional limitations of the **ALTER SYSTEM** command. Use the **SET** clause of the **ALTER SYSTEM** statement to set or change initialization parameter values. Additionally, the **SCOPE** clause specifies the scope of a change as follows:

- **MEMORY:** Changes of the parameter value are only in the currently running instance
- **SPFILE:** Changes of the parameter value are in the SPFILE only
- **BOTH:** Changes of the parameter value are in the currently running instance and the SPFILE

For dynamic parameters, you could also specify the **DEFERRED** keyword. When specified, the change is effective only for future sessions.

A **COMMENT** clause allows a comment string to be associated with the parameter update with a maximum length of 255 characters.

**Note:** In the example on the slide, only the SPFILE value is changed, not the current parameter value. The new value will be used when the database is restarted with the SPFILE option (as shown in later slides).

## STARTUP Command Behavior

```
SQL> STARTUP
```

- The default SPFILE is used to start up the instance; if that file is not found, the default server-side INIT.ORA file is used.

```
SQL> STARTUP  
2 PFILE=$ORACLE_HOME/DBS/INITDBA1.ora
```

- The specified INIT.ORA file is used to start up the instance.
- The PFILE can contain a definition to indicate use of an SPFILE.

ORACLE

### STARTUP Command Behavior

The default SPFILE name is used to start up the instance when the STARTUP command is issued. If the default SPFILE is not found, the Oracle server attempts to start the instance using the default INIT.ORA file on the server side. When the STARTUP PFILE command is issued, the specified INIT.ORA file is used to start up the instance. However, the PFILE can contain an indication to use an SPFILE.

## Viewing the Parameter Settings

- The V\$SPPARAMETER view shows the SPFILE contents:

```
SQL> SELECT name, value, update_comment
2  FROM    v$spparameter
3  WHERE   name = 'sort_area_size';
```

NAME	VALUE	UPDATE_COMMENT
-----	-----	-----
sort_area_size	1048576	temporary change

- The UPDATE\_COMMENT column has been added to V\$PARAMETER, V\$PARAMETER2 and V\$SYSTEM\_PARAMETER.

ORACLE

1-8

Copyright © Oracle Corporation, 2001. All rights reserved.

### Viewing Parameter Settings

There are several options for viewing the parameter settings. The new view V\$SPPARAMETER provides the contents of the SPFILE. It has the following columns:

- SID: The SID for which the parameter is defined
- NAME: The parameter name
- VALUE: The parameter value
- ISSPECIFIED: Whether the parameter is specified in the SPFILE
- ORDINAL: Ordinal number of value, if in a list of strings
- UPDATE\_COMMENT: Last update comments

A new column, UPDATE\_COMMENT, which provides comments associated with ALTER SYSTEM . . . COMMENT command, has been added to the V\$PARAMETER and V\$PARAMETER2 views.

**Note:** SHOW SGA, V\$PARAMETER, and V\$PARAMETER2 display the parameter values currently in use.

# Exporting an SPFILE

The contents of an SPFILE can be exported into an old-style PFILE:

```
CREATE PFILE[='PFILE-NAME' ]  
FROM SPFILE[='SPFILE-NAME' ]
```

Example:

```
SQL> CREATE PFILE='newpfile.ora' FROM SPFILE;
```

The PFILE is created as a text file on the server side.

ORACLE

1-9

Copyright © Oracle Corporation, 2001. All rights reserved.

## Exporting an SPFILE

In order to make modifications to an SPFILE, you can first export the contents into an old-style PFILE, then edit the output file, and then re-create the SPFILE. When exporting the contents of the SPFILE, the default SPFILE name is used if you do not specify a name. The PFILE is created as a text file on the server side.

The command shown in the slide can be executed either before or after you start up an instance.

You can also export an SPFILE to a PFILE to create backups of the persistent parameter file.

**Note:** In Oracle9i, this is a secondary method, because RMAN can also back up persistent parameter files.

Finally, you can export an SPFILE to provide a list of parameter values, much like using the `SHOW PARAMETER` command.

You need the SYSDBA or SYSOPER system privilege to execute this command shown on the slide.

**Note:** If you specify no names for the files, then a platform-specific name is used for the initialization parameter file, and it is created from the platform-specific default SPFILE.

## Exporting an SPFILE (continued)

The following is part of an exported PFILE:

```
*.background_dump_dest='/bdu/o9i/9i.001214_Rel9/admin/O9iRel12/bdump'
*.compatible='9.0.0'
*.control_files='/bdu/o9i/9i.001214_Rel9/oradata/O9iRel12/control01.ctl', '/bdu/o9i/9i.001214_Rel9/oradata/O9iRel12/control02.ctl', '/bdu/o9i/9i.001214_Rel9/oradata/O9iRel12/control03.ctl'
*.core_dump_dest='/bdu/o9i/9i.001214_Rel9/admin/O9iRel12/cdump'
*.db_16k_cache_size=10m
*.db_block_size=4096
*.db_cache_size=40m
*.db_name='O9iRel12'
*.db_writer_processes=3
*.disk_asynch_io=false
*.dispatchers='(PROTOCOL=TCP)(PRE=oracle.aurora.server.SGiopServer)'
*.instance_name='O9iRel12'
*.java_pool_size='20971520'
*.large_pool_size='1048576'
*.open_cursors=300
*.pga_aggregate_target='10m'
*.processes=150
*.remote_login_passwordfile='EXCLUSIVE'
*.resource_manager_plan='SYSTEM_PLAN'
*.sga_max_size=80m
*.shared_pool_size=50m
*.sort_area_retained_size=524288
*.sort_area_size=1048576#temporary change
*.timed_statistics=TRUE
*.undo_management='AUTO'
...
```

This file can be used the next time you start the database instance.



## Migrating to a SPFILE

1. Transfer the `init.ora` file from the client side to the server side using FTP, if needed.
2. Create a server parameter file using the `CREATE SPFILE` statement:

```
CREATE SPFILE[='SPFILE-NAME' ]  
FROM PFILE[='PFILE-NAME' ]
```

3. Use the created SPFILE to start up the instance:

```
SQL> STARTUP
```

ORACLE

## Shared Initialization Parameter Files

- **Parameters for different instances can be mixed in a single initialization parameter file:**
  - There is only one file to maintain and propagate
  - Having all parameter values available in a single location helps reduce errors
- **Use a dot notation with the instance name for instance-specific parameter values**

ORACLE

1-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### Shared Initialization Parameter Files

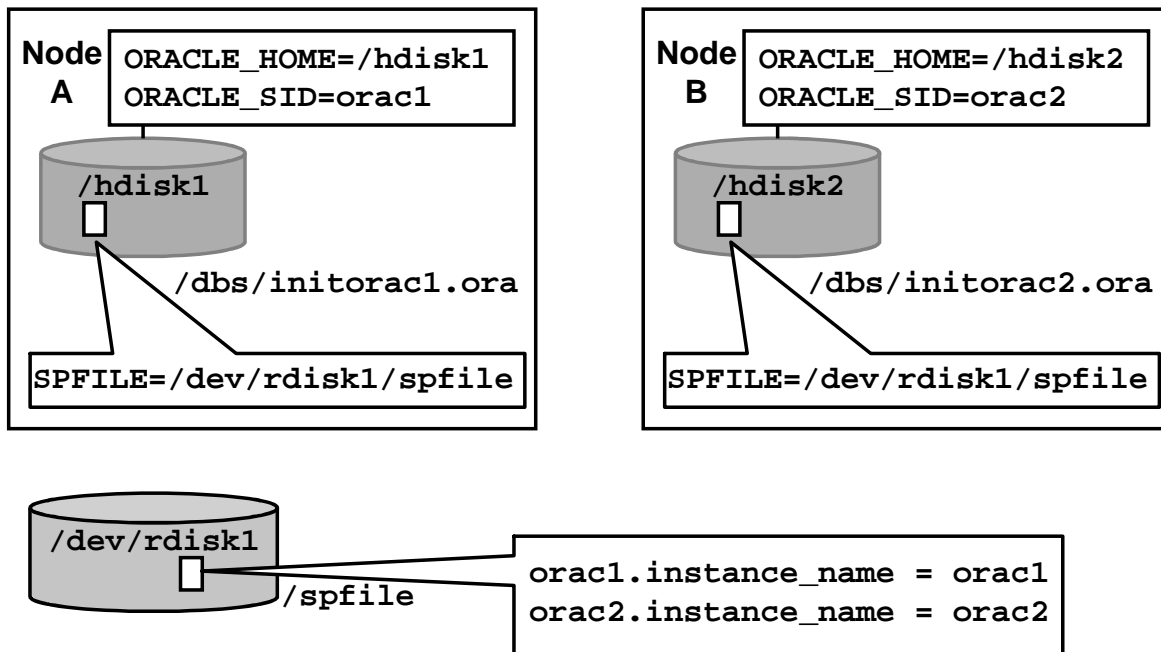
In previous releases, you needed separate initialization files to assign different parameter values to different instances of your database. At the same time, you had to provide identical values for some parameters across all instances. This required you to maintain multiple parameter files, typically using the `IFILE` parameter to link the common file to the instance-specific files.

In Oracle9i, you can store the parameters for all the instances of a Real Application Clusters database in a single file. This simplifies the management of the instances, because you can maintain only one file. It is also easier to avoid making mistakes, such as changing a value in one file but not another, if all the parameters are listed in one place.

To allow values for different instances to share the same file, parameter entries that are specific to a particular instance are prefixed with the instance name using a dot notation. For example, to assign different sort area sizes to two instances, PROD1 and PROD2, you can include the following entries in your common parameter file:

```
prod1.sort_area_size = 1048576
prod2.sort_area_size = 524288
```

## Shared SPFILE



ORACLE

1-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### Shared Server-Side Parameter Files

When you put the parameters for all your instances in a single initialization file, this file must be available to the processes that start up each instance. If your instances are started automatically as part of the system startup routines, you need a copy of the file on each node in the cluster. However, you can take advantage of server-side parameter files (SPFILES) in your Oracle9i Real Application Clusters environment. To do this, create your shared parameter file, containing parameter values for all of your instances, convert it to an SPFILE, and then store it on a shared disk (typically in a raw partition).

In the example in the slide, Node A uses `/hdisk1` for its `ORACLE_HOME` directory and supports an instance with an SID of `ORAC1`, while Node B uses `/hdisk2` for its `ORACLE_HOME` directory and runs an instance with an SID of `ORAC2`. A partition on the raw device, called `/dev/rdisk1/spfile`, holds the shared SPFILE.

Each node has an instance-specific initialization file configured, containing just one entry:

```
spfile = /dev/rdisk1/spfile
```

This entry points to the raw partition holding the shared SPFILE. Two entries from the SPFILE are shown in the example: The `INSTANCE_NAME` parameter for each of the two instances. The values of these parameters follow the recommended naming convention for instances (that is, naming the instance the same as the SID):

```
orac1.instance_name = orac1  
orac2.instance_name = orac2
```

# Summary

In this lesson, you should have learned how to:

- Create a server parameter file (SPFILE)
- Change initialization parameter values
- View parameter settings
- Export an SPFILE
- Migrate to an SPFILE

ORACLE

# 2

## Managing Memory

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# Objectives

**After completing this lesson, you should be able to:**

- **Set parameters to enable automatic and dynamic sizing SQL working areas**
- **Use new columns and views to gather information regarding SQL execution memory management**
- **Describe the allocation and tracking of memory behind a dynamic SGA**

ORACLE

## Automated SQL Execution Memory Management

- The Automated SQL Execution Memory Management features simplifies and improves memory allocation.
- The size of the SQL working areas can be adjusted automatically and dynamically.
- Benefits include:
  - Ease of memory tuning
  - Reduction of time it takes to tune memory
  - Better throughput with varying memory demands
  - Improved query response time

ORACLE

2-3

Copyright © Oracle Corporation, 2001. All rights reserved.

### Automated SQL Execution Memory Management

The Automated SQL Execution Memory Management feature simplifies and improves the way memory is allocated for SQL execution without shutting down the instance.

#### Manageability

In Oracle8i, a database administrator can control the maximum size of the working areas using various parameters, such as `SORT_AREA_SIZE`, `HASH_AREA_SIZE`, `BITMAP_MERGE_AREA_SIZE`, and `CREATE_BITMAP_AREA_SIZE`, but these parameter are difficult to set and tune. The Automated SQL Execution Memory Management feature releases the DBA from having to set these parameters. It provides automatic and dynamic memory tuning, thereby reducing the amount of time required to set and tune these parameters.

#### Performance

Because the parameters are adjusted automatically and dynamically, this feature can compensate for low or high memory usage, along with controlling the maximum amount of memory a query can use. In addition, memory-consuming operations such as hash joins, sorts, and so on, dynamically alter their memory profile to ensure the best possible use of system memory and optimal system performance.

#### Usability

The parameters mentioned above often waste PGA memory, because more memory is allocated than is needed. By reducing the amount of memory used, this feature allows the extra memory to be put to use by other queries. This is only available in dedicated server mode not shared server configuration.

# PGA Memory Management

- **Program Global Area (PGA) memory is classified into tunable and untunable memory:**
  - Tunable memory is memory consumed by SQL working areas.
  - Untunable memory is the remaining memory.
- **The PGA Memory Management feature sizes the tunable memory, when automatic mode is enabled.**
- **The size of the tunable portion allocated depends on an overall PGA memory target:**

$$\text{UNTUNABLE\_MEMORY\_SIZE} + \text{TUNABLE\_MEMORY\_SIZE} \leq \text{PGA\_AGGREGATE\_TARGET}$$

ORACLE

## PGA Memory Management

In order to define which part of the PGA memory is affected by the automatic tuning operation, the PGA is classified to differentiate between tunable and untunable memory. Tunable memory is the memory consumed by SQL working areas, and the rest is untunable memory.

The PGA Memory Management feature concentrates on sizing the tunable fraction of the PGA memory when the automatic mode is enabled. The size of the tunable portion allocated by an instance depends on an overall PGA memory target, which is explicitly set by the database administrator. Under this automatic mode, the system tries to have:

$$\text{UNTUNABLE\_MEMORY\_SIZE} + \text{TUNABLE\_MEMORY\_SIZE} \leq \text{PGA\_AGGREGATE\_TARGET}$$

When the automatic mode is enabled, the system can only control the tunable portion of the PGA memory. If this memory accounts for a very small percentage of the overall PGA memory, it can be impossible to enforce the above equation. This is typically the case in online transaction processing (OLTP) systems, where the tunable memory consumed is a relatively small percentage (<1%) of the total PGA memory.

Indeed, tunable memory probably constitutes a major portion of the PGA memory only under decision support system (DSS) workloads. For complex DSS workloads, the tunable memory accounts for most of the PGA memory (>90%) and most of the memory used by the instance. In that context, managing the tunable fraction is a very effective way to automatically manage the overall PGA memory consumed by the instance.



# Enabling Automated SQL Execution Memory Management

Enabling automatic tuning requires the setting of two parameters:

- **PGA\_AGGREGATE\_TARGET**: A dynamic initialization parameter
- **WORKAREA\_SIZE\_POLICY**: A dynamic session- and system-level parameter, with two values:
  - **MANUAL**: Tuning is performed using the \*\_AREA\_SIZE parameters.
  - **AUTO**: Tuning is performed automatically.

If **PGA\_AGGREGATE\_TARGET** is set, then the default value is **AUTO**.

ORACLE

2-5

Copyright © Oracle Corporation, 2001. All rights reserved.

## Enabling the Automated SQL Execution Memory Management

**PGA\_AGGREGATE\_TARGET** is a new system-level initialization parameter, which is set to specify the target aggregate amount of PGA memory available to the instance. This parameter is only a target, and can be dynamically modified at the instance level by the database administrator.

It accepts a number of bytes, kilobytes (K), megabytes (M), or gigabytes (G). The value must be between 10M and 400G.

**WORKAREA\_SIZE\_POLICY** is a new session- and system-level initialization parameter, which has two values, so that the user can select between automatic or manual tuning of work area sizes.

The **MANUAL** mode is the default if **PGA\_AGGREGATE\_TARGET** is not set. Under the manual mode, tuning is performed using the existing \*\_AREA\_SIZE parameters.

In **AUTO** mode, the work areas are sized to accomplish three goals:

- The overall size of the PGA memory should never exceed the value of **PGA\_AGGREGATE\_TARGET**.
- Tunable memory allocated by a process is regulated so that a process never runs out of memory.
- Memory should be allotted to work areas to optimize both throughput and response time.

# New Statistics and Columns

- **New statistics for V\$SYSSTAT and V\$SESSTAT:**
  - WORKAREA MEMORY ALLOCATED
  - WORKAREA EXECUTIONS - OPTIMAL
  - WORKAREA EXECUTIONS - ONEPASS
  - WORKAREA EXECUTIONS - MULTIPASS
- **New columns added to V\$PROCESS:**
  - PGA\_USED\_MEM
  - PGA\_ALLOC\_MEM
  - PGA\_MAX\_MEM

ORACLE

2-6

Copyright © Oracle Corporation, 2001. All rights reserved.

## New Statistics in V\$SYSSTAT and V\$SESSTAT

**WORK AREA MEMORY ALLOCATED:** Total amount of PGA memory dedicated to work areas allocated on behalf of a given session (V\$SESSTAT) or system (V\$SYSSTAT)

**WORK AREA EXECUTIONS - OPTIMAL:** The cumulative count of work areas which had an optimal size. For example, optimal size is defined if the sort does not need to spill to disk.

**WORK AREA EXECUTIONS - ONE PASS:** The cumulative count of work areas using the one-pass size. One-pass is generally used for big work areas where spilling to disk cannot be avoided.

**WORK AREA EXECUTIONS - MULTIPASS:** The cumulative count of work areas running in more than one pass. This should be avoided; it is a symptom of a poorly tuned system.

## New Columns in V\$PROCESS

**PGA\_USED\_MEM:** PGA memory currently used by the process

**PGA\_ALLOC\_MEM:** PGA memory currently allocated by the process, including free PGA memory not yet released to the operating system by the server process

**PGA\_MAX\_MEM:** Maximum PGA memory ever allocated by the process

## Example

```
SQL> select name, value from v$sysstat
```

```
2 where name like '%work area%';
```

NAME	VALUE
work area memory allocated (KB)	0
work area executions - optimal size	1544
work area executions - one pass size	11
work area executions - multipasses size	2

```
SQL> select sum(PGA_USED_MEM), sum(PGA_ALLOC_MEM)
```

```
2 , sum(PGA_MAX_MEM) from v$process;
```

SUM(PGA_USED_MEM)	SUM(PGA_ALLOC_MEM)	SUM(PGA_MAX_MEM)
9421418	10345058	22853922

ORACLE

### Example

In this example, PGA\_AGGREGATE\_SIZE is set to 10M, and WORKAREA\_POLICY\_SIZE is set to AUTO.

There is also a view called V\$PGASTAT that can give information regarding usage of memory.

# Dynamic SGA

- **The dynamic System Global Area (SGA) is an infrastructure which allows the SGA configuration to change without you shutting down the instance.**
- **Benefits include:**
  - **The Oracle server can modify its physical address space use to respond to the operating system's use of physical memory.**
  - **The SGA can grow and shrink dynamically, and in response to DBA commands.**

ORACLE

2-8

Copyright © Oracle Corporation, 2001. All rights reserved.

## Dynamic SGA

Since its inception, the SGA has always been a static allocation of memory, which was shared across all threads of execution. The size of memory is calculated based on values in the `init.ora` parameter file. After being allocated, the amount of usable shared memory cannot grow or shrink. If you want to increase the number of database block buffers, you needed to first shut down the instance, then modify the initialization parameter file, and then restart the instance.

A dynamic SGA implements an infrastructure which allows SGA configuration to change while the instance is running. This then allows the sizes of the buffer cache and shared pool to be changed without shutting down the instance.

In addition, the dynamic SGA infrastructure allows limits to be set at run time on how much physical memory is used for the SGA. The buffer cache and shared pool can be initially underconfigured, and they would grow and shrink depending upon their respective workloads.

# Units of Allocation

- **A new unit of allocation defined, called a granule:**
  - **Contiguous virtual memory allocation**
  - **Size based on `SGA_MAX_SIZE`**
- **SGA memory is tracked in granules by SGA components.**
- **The `V$BUFFER_POOL` view displays granule allocation and deallocation.**

ORACLE

2-9

Copyright © Oracle Corporation, 2001. All rights reserved.

## Granules

A granule is a unit of contiguous virtual memory allocation. The size of a granule depends on the estimated total SGA size, whose calculation is based on the value of the `SGA_MAX_SIZE` parameter: A granule is 4 MB if the estimated SGA size is less than 128 MB, and 16 MB otherwise.

The different components of the buffer cache can grow and shrink based on granule boundaries. For each component which owns granules, the `V$BUFFER_POOL` view tracks and displays the number of granules allocated to the component; any pending operations against the component (for example, the allocation or freeing of granules by means of the `ALTER SYSTEM` command, and the corresponding self-tuning); and the target size in granules. At instance startup, the Oracle server allocates granule entries, one for each granule to support `SGA_MAX_SIZE` bytes of address space. As startup continues, each component acquires as many granules as it requires. The minimum SGA configuration is three granules: One granule for the fixed SGA (including redo buffers), one granule for the buffer cache, and one granule for the shared pool.

## Cache Size and Number of Buffers

```
SQL> select block_size, current_size
2      , buffers, target_size, prev_size
3 from v$dbuffer_pool
4 where buffers!=0;
```

BLOCK SIZE	CURRENT SIZE	BUFFERS	TARGET SIZE	PREV SIZE
-----	-----	-----	-----	-----
4096	32	7868	32	48
2048	16	7570	16	0
16384	16	1013	16	0

ORACLE

2-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### Cache Size and Number of Buffers

Use the V\$BUFFER\_POOL view for statistics on all buffer pools available for the instance:

ID: Buffer pool ID number

NAME: Buffer pool name; possible values: DEFAULT, KEEP, RECYCLE (non-standard block size pools are always DEFAULT)

BLOCK\_SIZE: Block size in bytes for buffers in this pool

RESIZE\_STATE: Current status of the resize operation (STATIC, not resizing; ALLOCATING, allocating memory; ACTIVATING, creating new buffers; SHRINKING, deleting buffers)

CURRENT\_SIZE: Present size of the sub-cache in megabytes

BUFFERS: Current instantaneous number of buffers

TARGET\_SIZE: If a resize is in progress, records new target size in megabytes; otherwise the same as the current size of the pool

TARGET\_BUFFERS: If a resize is in progress, records new target size in terms of buffers; otherwise, the current number of buffers

PREV\_SIZE: Previous buffer pool size or zero if the pool has never been resized

PREV\_BUFFERS: Previous number of buffers in the pool or zero if the pool has never been resized

LO\_BNUM, HI\_BNUM, LO\_SETID, HI\_SETID, and SET\_COUNT are obsolete columns.

## Dynamic Shared and Large Pools

- **Shared pool can be dynamically resized to grow or shrink by using ALTER SYSTEM:**

```
SQL> ALTER SYSTEM  
2 SET SHARED_POOL_SIZE = 64M;
```

- **The allocation size has the following limits:**
  - **Size must be an integer multiple of the granule size**
  - **Total SGA size cannot exceed SGA\_MAX\_SIZE**

ORACLE

2-11

Copyright © Oracle Corporation, 2001. All rights reserved.

### Dynamic Shared Pool

In versions earlier than Oracle9i, the SGA is allocated once, when the instance is started. If the shared pool cannot find a large enough contiguous piece of memory, it signals an error.

With a dynamic SGA, on the other hand, the size of the shared pool can be dynamically resized to grow or shrink.

## Dynamic Buffer Cache

- The buffer cache can be dynamically resized to grow or shrink by using **ALTER SYSTEM**:

```
SQL> ALTER SYSTEM  
2 SET DB_CACHE_SIZE = 96M;
```

- The allocation size has the following limits:
  - Size must be an integer multiple of the granule size
  - Total SGA size cannot exceed **SGA\_MAX\_SIZE**
  - **DB\_CACHE\_SIZE** cannot be set to zero

ORACLE



## Increasing the Size of a Component's SGA Memory Area

```
Init.ora parameter values:
SGA_MAX_SIZE = 128M
DB_CACHE_SIZE = 96M
SHARED_POOL_SIZE = 32M

ALTER SYSTEM SET SHARED_POOL_SIZE = 64M;
(insufficient memory error message)

ALTER SYSTEM SET DB_CACHE_SIZE = 64M;
ALTER SYSTEM SET SHARED_POOL_SIZE = 64M;
(insufficient memory error message, check
V$BUFFER_POOL to see if shrink has completed)

ALTER SYSTEM SET SHARED_POOL_SIZE = 64M;
```

ORACLE

2-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### Increasing the Size of a Component's SGA Memory Area

A database administrator can increase the size of a component's SGA memory by issuing an `ALTER SYSTEM` command to modify the `init.ora` parameters value. The server takes the new size, rounds it up to the nearest multiple of the granule size (16 MB in this case) and adds or takes away granules to meet the target size.

You can add granules to a component (that is, increase the memory use of a component) with an `ALTER SYSTEM` command only if the system has enough free granules to satisfy the request. The database administrator must ensure this, because the system does not free another component's granules to satisfy the request. If the current amount of SGA memory is less than the value of `SGA_MAX_SIZE`, then the system can allocate more granules until the SGA size reaches `SGA_MAX_SIZE`.

The Oracle server, which invokes the `ALTER SYSTEM` command, reserves a set of granules for the corresponding SGA component (the `init.ora` parameter which defines the component's SGA use). After the reservation is complete, the foreground process hands the completion to the background process. The background process completes the operation by taking the reserved granules and adding them to the component's granule list. This is also referred to as growing a component's SGA memory area.

## New and Deprecated Buffer Cache Parameters

- **New parameters define cache sizes for primary block size buffers:**
  - DB\_CACHE\_SIZE
  - DB\_KEEP\_CACHE\_SIZE
  - DB\_RECYCLE\_CACHE\_SIZE
- **Deprecated parameters:**
  - DB\_BLOCK\_BUFFERS
  - BUFFER\_POOL\_KEEP
  - BUFFER\_POOL\_RECYCLE

**If you set these parameters, values are used but a warning message is issued.**

ORACLE

2-14

Copyright © Oracle Corporation, 2001. All rights reserved.

### New Parameters

The buffer cache consists of independent subcaches for buffer pools and for multiple block sizes. The DB\_BLOCK\_SIZE parameter determines the primary block size, which is used for the SYSTEM tablespace. The following three new parameters define the sizes of the caches for buffers for the primary block size:

- DB\_CACHE\_SIZE
- DB\_KEEP\_CACHE\_SIZE
- DB\_RECYCLE\_CACHE\_SIZE

The new value of DB\_CACHE\_SIZE refers only to the size of the DEFAULT buffer pool, as opposed to the total size of the DEFAULT, KEEP, and RECYCLE buffer pools.

### Deprecated Parameters

The DB\_BLOCK\_BUFFERS, BUFFER\_POOL\_KEEP, and BUFFER\_POOL\_RECYCLE buffer cache size parameters are deprecated, but have been maintained for backward compatibility. They will be made obsolete in the future. If the parameters are set, their values are used, but a warning message encourages the user to migrate to the new parameter scheme.

These parameters will continue to be static parameters. Furthermore, these parameters cannot be combined with the dynamic size parameters; combining them in the same parameter file produces an error.

## **Deprecated Parameters (continued)**

Prior to Oracle9i, the syntax for the `BUFFER_POOL_SIZE` parameter allowed the user to optionally specify the number of LRU latches for the buffer pool in addition to the number of buffers in the buffer pool. These latches were allocated out of the total number of latches specified by the `DB_BLOCK_LRU_LATCHES` parameter.

Because `DB_BLOCK_LRU_LATCHES` is now obsolete, the number of LRU latches for the buffer pool is calculated internally; if a specification is provided, it is ignored.

## Dynamic Buffer Cache Advisory Parameter

- **Enables and disables statistics gathering for predicting different cache size behavior**
- **Benefit:**
  - Information enables you to size the buffer cache optimally for a given workload
- **Enabled with DB\_CACHE\_ADVICE:**
  - Dynamic by means of `ALTER SYSTEM`
  - Three values available: `OFF`, `ON`, and `READY`

ORACLE

2-16

Copyright © Oracle Corporation, 2001. All rights reserved.

### Dynamic Buffer Cache Advisory Parameter

The buffer cache advisory feature enables and disables statistics gathering for predicting behavior with different cache sizes. The information provided by these statistics can help a database administrator size the buffer cache optimally for a given workload.

The buffer cache advisory is enabled by means of the `DB_CACHE_ADVICE` initialization parameter. It is a dynamic parameter with `ALTER SYSTEM`. Three values are available: `OFF`, `ON`, and `READY`:

- `OFF`: Advisory is turned off and the memory for the advisory is not allocated.
- `ON`: Advisory is turned on and both CPU and memory overhead is incurred.

Attempting to set the parameter to this state when it is in the `OFF` state may lead to error `ORA-4031: Inability to allocate from the shared pool when the parameter is switched to ON`. If the parameter is in a `READY` state, it can be set to `ON` without error because the memory is already allocated.

- `READY`: Advisory is turned off but the memory for the advisory remains allocated. Allocating the memory before the advisory is actually turned on avoids the risk of error `ORA-4031`. If the parameter is switched to this state from `OFF`, error `ORA-4031` may be generated.

## New View to Support Buffer Cache Advisory Information

- **V\$DB\_CACHE\_ADVICE displays the buffer cache statistics which are gathered.**
- **Rows predict the estimated number of physical reads for different cache sizes.**
- **The view also computes a physical read factor.**

ORACLE

2-17

Copyright © Oracle Corporation, 2001. All rights reserved.

### New View to Support Buffer Cache Advisory Information

The buffer cache advisory information is collected and displayed through a new view, V\$DB\_CACHE\_ADVICE. The view contains different rows that predict the estimated number of physical reads for different cache sizes. The rows also compute a physical read factor, which is the ratio of the number of estimated reads to the number of reads actually performed during the measurement interval by the real buffer cache.

The V\$DB\_CACHE\_ADVICE view has the following columns:

- **id:** Buffer pool ID (ranges from 1 to 8)
- **name:** Buffer pool name
- **block\_size:** Block size in bytes for buffers in this pool; possible values are the standard block size and powers of two: 2048, 4096, 8192, 16384, 32768
- **advice\_status:** ON indicates it is currently running; OFF indicates it is disabled (in this case the estimates are historical and calculated when last enabled).
- **size\_for\_estimate:** Cache size for prediction (in MB)
- **buffers\_for\_estimate:** Cache size for prediction (in buffers)
- **estd\_physical\_read\_factor:** Physical read factor for this cache size; ratio of number of estimated physical reads to the number of reads in the real cache. If there are no physical reads into the real cache, the value of this column is null.
- **estd\_physical\_reads:** Estimated number of physical reads for this cache size

## Summary

In this lesson, you should have learned how to:

- Set up automatic tuning of work areas using the `PGA_AGGREGATE_TARGET` and `WORKAREA_SIZE_POLICY` parameter
- Dynamically modify the size of the buffer cache and shared pool by using the `ALTER SYSTEM` command
- Enable and disable statistics gathering for predicting different cache size behavior
- Monitor work area allocation with the `V$DB_CACHE_ADVICE` view

ORACLE

# 3

## **Managing Data Files and Tablespaces**

ORACLE<sup>®</sup>

Copyright © Oracle Corporation, 2001. All rights reserved.

# Objectives

**After completing this lesson, you should be able to:**

- **Explain the concept and benefits of Oracle Managed Files (OMFs)**
- **Create and manage OMF**
- **Remove associated operating system files when removing a non-OMF tablespace**
- **Create and alter default temporary tablespaces**
- **Manage undo tablespaces**
- **Manage tablespaces with multiple block sizes**
- **Specify segment space management**

ORACLE



# Oracle Managed Files

- **Oracle Managed Files (OMFs) simplifies file administration by eliminating the need to manage files directly in an Oracle database.**
- **OMF has two major features:**
  - **You can create database objects without specifying the underlying operating system files.**
  - **It automatically removes obsolete data files and online redo logs.**

ORACLE

3-3

Copyright © Oracle Corporation, 2001. All rights reserved.

## Oracle Managed Files

An Oracle Managed File (OMF) is a file that is created automatically when needed and deleted automatically when no longer needed. The name of the Oracle-managed file is generated by the system when the file is created. An OMF is designed to simplify the administration of an Oracle database by eliminating the need to directly manage the files composing the database.

Database administrators specify operations in terms of database objects rather than file names. Oracle internally uses standard file system interfaces to create and delete files as needed for tablespaces, online logs, and control files. You need to specify only the file system directory to be used for a particular type of file, and the system ensures that a unique file (an Oracle-managed file) is created and deleted when necessary.

This feature has two major characteristics:

First, it uses file system directories rather than file names for creating control files, data files, and online logs. This simplifies administration, because there is no need to invent a file name for each file. The file system defines the characteristics of the storage and the pool where it is allocated. A consistent set of rules is used to name all files, and every new file has a single unique name so it cannot damage an existing file.

Second, it automatically removes old control files, data files, and online logs. Currently, removing files that are no longer needed on disk is a major administrative task, which frequently leads to deleting files that are still needed. A lot of disk space is wasted on large systems simply because no one is sure whether a particular file is needed any more. When all file deletions are handled by the system, there are fewer mistakes.

## Benefits of Using Oracle Managed Files

- **Eases the development of portable applications**
- **Simplifies the creation and administration of test and development databases**
- **Reduces corruption caused by specifying the wrong file names**
- **Reduces the wasted disk space consumed by obsolete files**

ORACLE

3-4

Copyright © Oracle Corporation, 2001. All rights reserved.

### Benefits of Oracle Managed Files

- Eases development of portable applications, by eliminating the need to put operating system-specific file names in SQL scripts
- Simplifies the creation and administration of test and development databases, because operation system files are deleted as soon as they are no longer required
- Reduces corruption caused by DBAs specifying the wrong file names
- Reduces the wasted disk space consumed by obsolete files

The target customer databases for the initial release of OMF are:

- Very low end databases
- Databases that are supported by:
  - A logical volume manager, such as Veritas, which provides striping/RAID and dynamically extensible logical volumes
  - A file system that provides large, extensible files

# Defining Oracle Managed Files

**Two basic configurations exist for establishing OMF:**

- **Setting two dynamic initialization parameters:**
  - **DB\_CREATE\_FILE\_DEST:** Defined for data files
  - **DB\_CREATE\_ONLINE\_LOG\_DEST\_*n*:** Defined for online redo logs and control files
- **Setting one dynamic initialization parameter:**  
**DB\_CREATE\_FILE\_DEST.**

**All files are located in one location.**

```
SQL> ALTER SYSTEM  
2 SET db_create_file_dest='/bdu/o9i/dbs';
```

ORACLE

3-5

Copyright © Oracle Corporation, 2001. All rights reserved.

## OMF Configurations

Two basic OMF configuration exist. The first, and preferred, configuration is to set two initialization parameters:

- **DB\_CREATE\_FILE\_DEST:** Set to give the default location for data files
- **DB\_CREATE\_ONLINE\_LOG\_DEST\_*n*:** Set to give the default locations for online logs and control files

This configuration provides good separation of data files, control files, and online redo logs, with optional Oracle multiplexing of online redo logs and control files where *n* is an integer between 1 and 5. For example:

```
DB_CREATE_FILE_DEST = '/u01/oradata/'  
DB_CREATE_ONLINE_LOG_DEST_1 = '/u02/oradata/'  
DB_CREATE_ONLINE_LOG_DEST_2 = '/u03/oradata/'
```

The second configuration provides a single initialization parameter, **DB\_CREATE\_FILE\_DEST**. In this configuration, all the data files, control files, and online logs are created in the same file system location.

## OMF Filename Structure

- Oracle-managed files comply with the Oracle Flexible Architecture (OFA)
- For example, on Solaris, Oracle-managed files are named as follows:
  - Control files: `ora_%u.ctl`
  - Redo log files: `ora_%g_%u.log`  
Example: `ora_1_xbwny5m9.log`
  - Data files: `ora_%t_%u.dbf`  
Example: `ora_omf_xbwn3h1m.dbf`
  - Temporary data files: `ora_%t_%u.tmp`
- File names are accepted in SQL commands whenever a file name is used to identify an existing file.

ORACLE

3-6

Copyright © Oracle Corporation, 2001. All rights reserved.

### OMF Filename Structure

Oracle-managed files comply with the Oracle Flexible Architecture (OFA) and are platform dependent. The special characters are defined as follows:

- `%u` is an eight-character string that guarantees uniqueness.
- `%t` is the tablespace name, truncated if necessary to fit into the maximum filename length. Placing the tablespace name before the uniqueness string means that all the data files for a tablespace appear next to each other in an alphabetical file listing.
- `%g` is the redo log file group number.
- `ora_` identifies the file as an OMF.

With the above naming structure, it is obvious from looking at the filename that a file is an Oracle-managed file.

# Managing Control Files with OMF

**To create Oracle-managed control files:**

- **Define the DB\_CREATE\_FILE\_DEST or one or more DB\_CREATE\_ONLINE\_LOG\_DEST\_n parameters.**
- **Omit the CONTROL\_FILES parameter at database creation.**
  - **One control file is created**
    - **In each of DB\_CREATE\_ONLINE\_LOG\_DEST\_n if set**
    - **In DB\_CREATE\_FILE\_DEST otherwise**
  - **Control file name is automatically generated.**
  - **Files are displayed in the alert log.**

ORACLE

3-7

Copyright © Oracle Corporation, 2001. All rights reserved.

## Managing Control Files

If the CONTROL\_FILES initialization parameter is not set when the database is created, a control file is created in each DB\_CREATE\_ONLINE\_LOG\_DEST\_n location or, if these parameters are not set, a single copy is created in the DB\_CREATE\_FILE\_DEST location. The file names are uniquely generated, for example ora\_cmr7t30p.ctl, and displayed in the alert log when the files are created. In this case, the CONTROL\_FILES parameter in the init.ora file must then be set to the generated file names. The control files thus created are Oracle Managed Files.

**Note:** If an SPFILE is used, the CONTROL\_FILES parameter is automatically set and saved when the database is created.

The DBA can also create a new control file for the database by using the CREATE CONTROLFILE command. The correct OMF filenames must be used in the DATAFILE and LOGFILE clauses. The ALTER DATABASE BACKUP CONTROLFILE TO TRACE command generates a script with the correct file names. Alternatively, the file names can be found by selecting from V\$DATAFILE, V\$TEMPFILE, and V\$LOGFILE.

# Managing Online Redo Log Files with OMF

**To create Oracle-managed online redo log files:**

- **Define the `DB_CREATE_ONLINE_LOG_DEST_n` parameter.**
  - **Two online redo log groups are created in `DB_CREATE_ONLINE_LOG_DEST_1` and `DB_CREATE_ONLINE_LOG_DEST_2`.**
  - **Each group has two members.**
  - **Names are automatically generated.**
  - **Filenames are displayed in the alert log.**
  - **The default size is 100 MB.**

ORACLE

3-8

Copyright © Oracle Corporation, 2001. All rights reserved.

## Managing Online Redo Log Files

The online redo logs can be created and added without specifying any filenames. Default file system locations are used. The log file member file names are generated automatically, for example `ora_1_wo94n2xi.log`. The default size is 100 MB.

If more than one file system location is provided, each group has one member in each directory. Automatically created log file members are automatically deleted when no longer needed. If the command fails, a partially created log file group is deleted automatically.

# Managing Online Redo Log Files with OMF

- A complete group can be added with no file specification:

```
SQL> ALTER DATABASE ADD LOGFILE;
```

- If a group is dropped, all the corresponding Oracle-managed files are deleted at the operating system level:

```
SQL> ALTER DATABASE DROP LOGFILE GROUP 3;
```

ORACLE

3-9

Copyright © Oracle Corporation, 2001. All rights reserved.

## Adding a Group

To create a new group of online redo log files, use the ALTER DATABASE ADD LOGFILE command. The command has been modified so that the file specification is not necessary. The above example adds a log file with a member in the DB\_CREATE\_ONLINE\_LOG\_DEST\_1 location and a member in the DB\_CREATE\_ONLINE\_LOG\_DEST\_2 location. Unique file names for the log file members are generated automatically.

## Dropping a Group

The GROUP clause can be used to drop a log file. In the above example the operating system file associated with each Oracle-managed log file member is automatically deleted.

## Archived Redo Logs and OMF

Archived redo log files cannot be Oracle-managed files. A file system location for the archived log files can be specified with the LOG\_ARCHIVE\_DEST\_n initialization parameters. The filenames are formed based on the LOG\_ARCHIVE\_FORMAT parameter or its default.

# Managing Tablespaces with OMF

**To create OMF tablespaces:**

- **Define the DB\_CREATE\_FILE\_DEST parameter.**
- **Use the CREATE TABLESPACE command, for example:**

```
SQL> CREATE TABLESPACE omf_2;
```

- The data file is created in DB\_CREATE\_FILE\_DEST.
  - The DATAFILE clause is no longer required.
  - The default size is 100 MB.
  - AUTOEXTEND UNLIMITED
- **When the tablespace is dropped, all Oracle-managed files are also deleted at the operating system level.**

ORACLE

3-10

Copyright © Oracle Corporation, 2001. All rights reserved.

## Managing Tablespaces with OMF

The above example creates the TBS1 tablespace. The data file filename is generated automatically, for example, ora\_tbs1\_2ixfh90q.dbf.

The example below sets the default location for data file creations to /u01/oradata/sample4 and then creates a tablespace called omf\_2 in that directory containing two autoextensible data files with an unlimited maximum size and an initial size of 500 MB:

```
SQL> ALTER SYSTEM SET  
2 DB_CREATE_FILE_DEST = '/u01/oradata/sample4';  
SQL> CREATE TABLESPACE omf_2 DATAFILE SIZE 500M, SIZE 500M;
```

All tablespaces created in Oracle9i have EXTENT MANAGEMENT LOCAL as default. If you want to have a dictionary managed tablespace you need to specify a EXTENT MANAGEMENT DICTIONARY clause when creating the tablespace.



## Managing Tablespaces with OMF

- An Oracle-managed data file can be added to an existing tablespace.

```
SQL> ALTER TABLESPACE TBS1 ADD DATAFILE;
```

- The default file system directory can be changed dynamically.

```
SQL> ALTER SYSTEM SET  
2 DB_CREATE_FILE_DEST='/u04/oradata/';
```

- This does not affect any existing data files, only future Oracle-managed file creation.

ORACLE

### Managing Tablespaces with OMF

An Oracle-managed data file can be added to an existing tablespace. The `ADD DATAFILE` command no longer requires file specification. The first statement above adds a data file to the `TBS1` tablespace. The data file is created in the location specified by the `DB_CREATE_FILE_DEST` parameter, and has the default size of 100 MB.

## The Delete Datafiles Feature

- This feature provides automatic removal of operating system files in non-OMF tablespaces.
- Benefits include:
  - Simplifying database administration
  - Reducing corruption caused by accidentally removing operating system files
  - Conserving disk space
- This feature also adds functionality to the **DROP TABLESPACE** command:

```
SQL> DROP TABLESPACE TBS1  
2 INCLUDING CONTENTS AND DATAFILES;
```

ORACLE

3-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### Delete Datafiles

The goal of this feature is to simplify the administration of Oracle databases by providing administrators with an option for automatically removing the operating system files in a non-OMF tablespace when dropping the tablespace from the database.

This feature can help reduce the corruption caused by administrators accidentally removing the wrong operating system files. It can also help reduce the wasted disk space consumed by obsolete files.

#### **DROP TABLESPACE**

A new option, **AND DATAFILES**, has been added to the **DROP TABLESPACE** command to delete the operating system files associated with a non-OMF tablespace.

**INCLUDING CONTENTS AND DATAFILES** deletes the contents and operating system files of the tablespace.

A message is written to the alert log for each file deleted.

The command succeeds even if an operating-system error prevents the deletion of a file, and a message describing the error is written to the alert log.

## Default Temporary Tablespaces

- You can define a databasewide default temporary tablespace.
- If not defined, the `SYSTEM` tablespace is the default temporary tablespace.
- You define the default temporary tablespace by using one of the following statements:
  - `CREATE DATABASE`
  - `ALTER DATABASE`

ORACLE

3-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### Default Temporary Tablespaces

Prior to Oracle9i, `SYSTEM` is the default tablespace for storing temporary data if a temporary tablespace was not explicitly assigned. With Oracle9i a default temporary tablespace can be defined databasewide. This eliminates the problem of users defaulting to the `SYSTEM` tablespace.

After a default temporary tablespace is defined, users not explicitly assigned to a temporary tablespace are assigned to the default temporary tablespace. Unfortunately, if the default temporary tablespace is not defined and a user is not explicitly assigned to a temporary tablespace, the `SYSTEM` tablespace becomes the default temporary tablespace, and a warning message is placed in the alert log.

# Creating Default Temporary Tablespaces

In this example, a default temporary tablespace is created during database creation:

```
CREATE DATABASE db1
  CONTROLFILE REUSE
  LOGFILE '/u02/oradata/log1.log' SIZE 50K
  LOGFILE '/u02/oradata/log2.log' SIZE 50K
  DATAFILE
    '/u01/oradata/df1.dbf' AUTOEXTEND ON
    '/u01/oradata/df2.dbf' AUTOEXTEND ON
    NEXT 10M MAXSIZE UNLIMITED
  DEFAULT TEMPORARY TABLESPACE dts1
  TEMPFILE '/u03/temp/dts_1.dbf' SIZE 20M
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M
```

ORACLE

3-14

Copyright © Oracle Corporation, 2001. All rights reserved.

## Creating Default Temporary Tablespaces

When created with the CREATE DATABASE command, the default temporary tablespace is of locally managed type.

You can query the DATABASE\_PROPERTIES view to see the current default temporary tablespace:

```
SQL> select property_value
      2   from database_properties
      3  where property_name= 'DEFAULT_TEMP_TABLESPACE';
```

```
PROPERTY_VALUE
-----
DTS1
```

## Altering Default Temporary Tablespaces

- The **ALTER DATABASE** command is extended, so you can change a default temporary tablespace:

```
SQL> ALTER DATABASE db1  
2   DEFAULT TEMPORARY TABLESPACE dts2;
```

- Perform the following steps to replace a default temporary tablespace:
  1. Create a new temporary tablespace.
  2. Identify this as the default temporary tablespace.
  3. Drop the old default temporary tablespace.

ORACLE

3-15

Copyright © Oracle Corporation, 2001. All rights reserved.

### Altering Default Temporary Tablespaces

Default temporary tablespaces cannot be dropped. Therefore, the default temporary tablespace must be altered to a new default temporary tablespace, then the old default temporary tablespace can be dropped. After altering the default temporary tablespace, users assigned to the default temporary tablespace are automatically reassigned to the new default temporary tablespace.

#### How to Create a New Default Temporary Tablespace

1. Create new default temporary tablespace using the **CREATE TEMPORARY TABLESPACE** command (or the **CREATE TABLESPACE** with the **TEMPORARY** option).
2. Use the **ALTER DATABASE** command to change default temporary tablespace from the old to the new.
3. Use the **DROP** command to remove the old default temporary tablespace.

## Restrictions on Default Temporary Tablespaces

- The default temporary tablespace cannot be dropped until after a new default is made available.
- You cannot alter the default temporary tablespace to a permanent tablespace.
- The default temporary tablespace cannot be taken offline.

ORACLE

3-16

Copyright © Oracle Corporation, 2001. All rights reserved.

### Dropping a Default Temporary Tablespace

Dropping the default temporary tablespace is not allowed until after a new default is made available. The `ALTER DATABASE` command must be used to change the default temporary tablespace to a new default. The old default temporary tablespace is then dropped only after a new default temporary tablespace is made available. Users assigned to the old default temporary tablespace are automatically be reassigned to the new default temporary tablespace.

### Changing to a Permanent Type

Because a default temporary tablespace must be either the `SYSTEM` tablespace or a temporary type tablespace, changing the default temporary tablespace to a permanent type is not allowed.

### Taking a Default Temporary Tablespace Offline

Tablespaces are taken offline to make that part of the database unavailable to other users; for example, for offline backup, maintenance, or making a change to an application that uses the tablespace. Since none of these situations apply to a temporary tablespace, taking a default temporary tablespace offline is not allowed.

# Automatic Undo Management

- **Simplifies and automates the management of undo data.**
- **Benefits of feature:**
  - DBA has no reason to create, drop, or alter rollback segments.
  - DBA has a choice of managing undo segments automatically or manually.
- **Mode set by the UNDO\_MANAGEMENT parameter:**
  - **AUTO:** Instance manages undo data automatically
  - **MANUAL:** DBA manages undo data manually
- **Undo data is managed by a single tablespace defined by UNDO\_TABLESPACE.**

ORACLE

3-17

Copyright © Oracle Corporation, 2001. All rights reserved.

## Automatic Undo Management

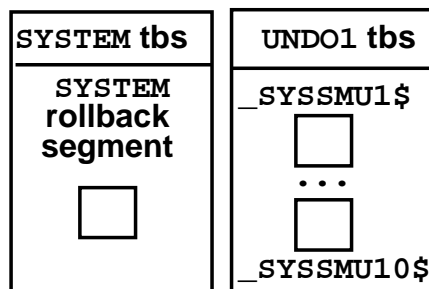
The Automatic Undo Management feature simplifies and automates the management of rollback segments, also referred to as undo segments. Using this feature, you no longer need to create, drop, or alter undo segments.

You have the choice of managing undo segments automatically or manually by choosing an undo mode. The undo mode is set by the UNDO\_MANAGEMENT initialization parameter, which can have a value of AUTO or MANUAL. If AUTO is selected, the instance automatically maintains undo data within the undo tablespace. If MANUAL is selected, you need to create and manage undo segments manually, as in versions prior to Oracle9i.

The undo data is managed by a single undo tablespace. The undo tablespace can be defined during database creation using either the UNDO TABLESPACE clause or the CREATE UNDO TABLESPACE command. These two options are discussed further within this lesson.

# The Automatic Undo Management Concept

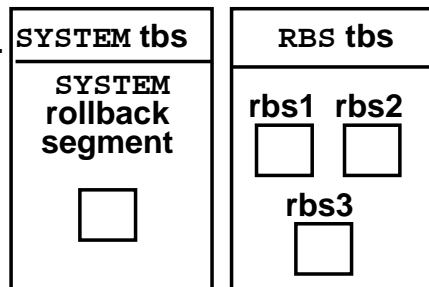
**AUTO**



**AUTO mode requires:**

- CREATE UNDO TABLESPACE UNDO1
- UNDO\_MANAGEMENT=AUTO
- UNDO\_TABLESPACE=UNDO1

**MANUAL**



**MANUAL mode requires:**

- UNDO\_MANAGEMENT=MANUAL
- CREATE TABLESPACE RBS
- CREATE ROLLBACK SEGMENT rbs1  
storage (initial ...);
- ROLLBACK\_SEGMENTS=  
(rbs1,rbs2,rbs3)

ORACLE

3-18

Copyright © Oracle Corporation, 2001. All rights reserved.

## The Automatic Undo Management Concept

With the Automatic Undo Management design, the notion of a single SYSTEM rollback segment is retained within the SYSTEM tablespace. It is created automatically during database creation, is automatically managed, and can not be taken offline.

You allocate undo space for the workload in that instance in a single undo tablespace, instead of distributing it into a set of statically allocated rollback segments. Rollback segments are still used for the execution of transactions, but they are created internally and maintained as undo segments.

A DBA no longer must decide the different number and sizes of rollback segments to create, or how to assign transactions of different sizes strategically to individual rollback segments. You also do not have to adjust the attributes of rollback segments in order to balance undo block contentions and space utilization issues.



## Creating an Undo Tablespace

```
CREATE DATABASE db01
...
UNDO TABLESPACE undotbs1
DATAFILE '/u01/oradata/undotbs1.dbf' SIZE 20M
AUTOEXTEND ON
```

If **UNDO\_MANAGEMENT** is set to **AUTO** and the **UNDO TABLESPACE** clause is not specified, a default is created, such as:

- **SYS\_UNDOTBS** tablespace
- **AUTOEXTEND** is set to **ON**
- Space is allocated from the default set of files used by the **CREATE DATABASE** statement

ORACLE

### The UNDO TABLESPACE Command

Automatic Undo Management requires an undo tablespace. More than one undo tablespace can exist in the database, but only one can be active. You can create the undo tablespace with the database by adding a clause to the **CREATE DATABASE** statement.

**Note:** In the creation of an undo tablespace, the **DATAFILE** clause is the only clause allowed. In addition, if the **UNDO\_MANAGEMENT** parameter is set to **AUTO** and you omit the **UNDO TABLESPACE** clause from the **CREATE DATABASE** statement, the system creates an undo tablespace with the name **SYS\_UNDOTBS**. This tablespace is built on one of the default data files used by the **CREATE DATABASE** statement and its attributes are predefined: the data file has a size of 10 MB and it is autoextensible.

## Creating an Undo Tablespace

- An undo tablespace can also be created after database creation:

```
SQL> CREATE UNDO TABLESPACE undotbs1  
2 DATAFILE '/u01/oradata/undotbs1.dbf'  
3 SIZE 20M;
```

- After an undo tablespace is created, it:
  - Can be specified by `UNDO_TABLESPACE`
  - Can be used only in **AUTO** mode
  - Can store only undo information
  - Is organized internally

ORACLE

3-20

Copyright © Oracle Corporation, 2001. All rights reserved.

### Creating an Undo Tablespace

After an undo tablespace is created, the `UNDO_TABLESPACE` parameter can be defined in the initialization parameter file.

An undo tablespace can only be used in the **AUTO** mode for storing undo data. It is organized internally.

#### Undo Tablespaces for Real Application Clusters

An undo tablespace must be created for each instance.

## Altering an Undo Tablespace

- To make changes to undo tablespaces, use the **ALTER TABLESPACE** command.
- For example, to add a second data file to the undo tablespace:

```
SQL> ALTER TABLESPACE undotbs1 ADD  
      2 DATAFILE '/u01/oradata/undotbs2.dbf'  
      3 AUTOEXTEND ON;
```

ORACLE

3-21

Copyright © Oracle Corporation, 2001. All rights reserved.

### Altering an Undo Tablespace

The following clauses are supported when altering an undo tablespace:

- ADD DATAFILE
- RENAME
- DATAFILE [ONLINE | OFFLINE]
- BEGIN BACKUP
- END BACKUP

## Dropping an Undo Tablespace

- To drop an undo tablespace, use the **DROP TABLESPACE** command:

```
SQL> DROP TABLESPACE undotbs1;
```

- The tablespace can be dropped only if it is not currently being used by an instance.
- Accessing transactions residing in dropped undo tablespaces generates the ORA1555 error.

ORACLE

3-22

Copyright © Oracle Corporation, 2001. All rights reserved.

### Dropping an Undo Tablespace

An undo tablespace can only be dropped if it is not in use by an instance. Therefore, its transaction tables must not contain any uncommitted transactions.

The syntax `DROP TABLESPACE undotbs1` behaves the same as:

```
DROP TABLESPACE undotbs1 INCLUDING CONTENTS;
```

Like dropped **MANUAL** rollback segments, transactions residing in dropped undo tablespaces cannot be accessed without generating a ORA1555 error, if the snapshot is older than the DROP-SCN of the undo tablespace.

## Switching Undo Tablespaces

- A DBA can switch from using one undo tablespace to another.
- Only one undo tablespace can be used by an instance at a time.
- Only one undo tablespace can be assigned active at a time.
- Switching is done by using the **ALTER SYSTEM** command:

```
SQL> ALTER SYSTEM  
2 SET UNDO_TABLESPACE=undotbs2;
```

ORACLE

3-23

Copyright © Oracle Corporation, 2001. All rights reserved.

### Switching Undo Tablespaces

For example, perform the following steps to switch from using undotbs1 to undotbs2:

1. First, create undo tablespace that is being switched to, if not already created:  

```
SQL> CREATE UNDO TABLESPACE undotbs2 DATAFILE  
2 'undodb02.dbf' SIZE 20M;
```
2. Change the current undo tablespace from undotbs1 to undotbs2 by using the **ALTER SYSTEM** command.  

```
SQL> ALTER SYSTEM SET UNDO_TABLESPACE=undotbs2;
```
3. Drop the undo tablespace undotbs1 after all transactions within the tablespace are complete.
4. If there are no active transactions, drop undotbs1:  

```
SQL> DROP TABLESPACE undotbs1;
```

## New Parameters to Support Automatic Undo Management

- **UNDO\_MANAGEMENT:** Specifies automatic or manual mode
- **UNDO\_TABLESPACE:** Specifies the undo tablespace to be used
- **UNDO\_SUPPRESS\_ERRORS:** Suppresses errors when attempting to execute manual operations while in automatic mode
- **UNDO\_RETENTION:** Controls the amount of undo information retained

ORACLE

3-24

Copyright © Oracle Corporation, 2001. All rights reserved.

### UNDO\_MANAGEMENT

This parameter determines the Automatic Undo Management mode of the database. The parameter may be set to one of two values, AUTO or MANUAL, and must be set in the initialization parameter file. AUTO mode sets the database to Automatic Undo Management and requires an undo tablespace. MANUAL mode is the default value, and allows the DBA to create and manage undo segments as need within the database. UNDO\_MANAGEMENT cannot be changed dynamically after the database starts.

### UNDO\_TABLESPACE

This parameter specifies the undo tablespace to be used when the AUTO mode of the UNDO\_MANGEMENT has been selected. This parameter must be defined in the initialization parameter file. It can be altered dynamically using the ALTER SYSTEM command:

```
SQL> ALTER SYSTEM SET undo_tablespace = UNDO1;
```

or

```
SQL> ALTER SYSTEM SET undo_tablespace = '';
```

**Note:** Do not set the UNDO\_TABLESPACE parameter during database creation. This causes an error to be generated. After the database is created, shut down the database, set UNDO\_TABLESPACE, and then start up the database.

## **UNDO\_SUPPRESS\_ERRORS**

When set to TRUE, this parameter suppresses the errors generated by attempting to execute MANUAL operations, such as ALTER ROLLBACK SEGMENT ONLINE or SET TRANSACTION USE ROLLBACK SEGMENT, while in AUTO mode. Instead of receiving an error in your application, the database will only generate the following error in the alert log:

```
alter rollback segment rbs1 online
Sun Jul 08 22:41:59 2001
Unsupported Rollback Segment operation issued
(and ignored) in SMU mode
```

The default value is FALSE.

## **UNDO\_RETENTION**

Committed undo information is typically lost when its undo space is overwritten by a newer transaction. But for consistent read purposes, long-running queries can require old undo information for undoing changes and producing older images of data blocks. The UNDO\_RETENTION initialization parameter provides a means of explicitly specifying the amount of undo information to retain.

If the UNDO\_RETENTION initialization parameter is not specified, the default value is 30 seconds. Given a specific UNDO\_RETENTION parameter setting and some system statistics, the amount of undo space required to satisfy the undo retention requirement can be estimated using the following formula:

$$\text{Undo space} = (\text{UR} * \text{UPS}) + \text{Overhead}$$

where:

Undo space = Number of undo blocks

UR = Value of the UNDO\_RETENTION parameter, in seconds

UPS = Undo blocks per second

Overhead = Overhead for metadata (transaction tables, bitmaps, and so on)

For example, if UNDO\_RETENTION is set to 2 hours, and the transaction rate is 200 undo blocks per second, with a 4 KB block size, then the required undo space is computed as follows:

$$(2 * 3600 * 200 * 4 \text{ KB}) = 5.8 \text{ GB}$$

## New View to Support Automatic Undo Management

- **V\$UNDOSTAT:** Displays information to monitor how undo segments are executed in the current instance
- **Used to:**
  - Estimate the amount of undo space required
  - Tune undo usage
- **This view is available for both MANUAL or AUTO mode.**

ORACLE

3-26

Copyright © Oracle Corporation, 2001. All rights reserved.

### New View to Support Automatic Undo Management

The V\$UNDOSTAT view displays statistical data showing how well the system is working. Each row keeps statistics collected in the instance for a ten-minute interval. It can be used to estimate the amount of undo space required for the current workload. The database uses the the view to tune undo usage in the system.

#### Example

```
SQL> select begin_time, end_time, undoblks, txncount,
2          maxconcurrency as maxcon
3   from    v$undostat
4  order   by begin_time;
```

BEGIN_TIME	END_TIME	UNDOBLKS	TXNCOUNT	MAXCON
17.03.01 23:21	17.03.01 23:31	8	25	1
17.03.01 23:31	17.03.01 23:41	37	67	2
17.03.01 23:41	17.03.01 23:51	5	20	2
...				



## Multiple Block Size Support

- **Oracle9i supports the creation of databases with multiple block sizes.**
- **Benefits include:**
  - **Ability to locate objects in tablespaces of appropriate block size in order to maximize I/O performance**
  - **Ability to transport tablespaces between databases with different block sizes (for example, between an OLTP database and an Enterprise Data Warehouse)**

ORACLE

3-27

Copyright © Oracle Corporation, 2001. All rights reserved.

### Multiple Block Size Support

Previous to Oracle9i, tablespaces having different block sizes within a database were not supported. Such functionality is required, for example, to transport a tablespace from an OLTP database to an Enterprise Data Warehouse. Support for multiple block sizes within the database facilitates such transportability.

## Creating Standard Block Size Tablespaces

- The standard block size is set during database creation using the `DB_BLOCK_SIZE` parameter, and it cannot be changed subsequently without re-creating the database.
- The standard block size is used for the `SYSTEM` tablespace, and by default for the other tablespaces.

ORACLE

## Creating a Tablespace with a Nonstandard Block Size

- An Oracle tablespace can be created with a standard block size, or with a nonstandard block size.
- The nonstandard size can be any power of two between 2 KB and 32 KB.
- You create tablespaces of nonstandard block sizes by using the `CREATE TABLESPACE` command with the `BLOCKSIZE` keyword:

```
SQL> CREATE TABLESPACE tbs_1
2  DATAFILE '\u01\oradata\tbs_1.dbf'
3  SIZE 10M BLOCKSIZE 4096;
```

ORACLE

3-29

Copyright © Oracle Corporation, 2001. All rights reserved.

### Creating a Tablespace of Nonstandard Block Size

The statement above creates a new tablespace called `tbs_1` with the file `file_1.dbf` having a block size of 4 KB. In order for this statement to succeed, the buffers of size 4 KB must currently be configured in the buffer cache.

**Note:** To use nonstandard block sizes, you must configure subcaches within the buffer cache area of SGA memory for all of the nonstandard block sizes (see later in this lesson).

## Multiple Block Sizing Rules

- All partitions of a partitioned object must reside in tablespaces of the same block size.
- All temporary tablespaces, including permanent tablespaces being used as default temporary tablespaces, must be of standard block size.
- Index Organized Table (IOT) overflow and out-of-line large object (LOB) segments can be stored in a tablespace with a block size different than that of the base table.

ORACLE

## Nonstandard Block Sizes

- **Corresponding to the tablespaces additional caches must be configured with the following dynamic initialization parameters:**
  - **DB\_ *n* K\_CACHE\_SIZE**, where *n* = 2, 4, 8, 16, or 32
  - **DB\_CACHE\_SIZE**
- **The default size of the DB\_ *n* K\_CACHE\_SIZE parameter, for all values of *n*, is 0 MB.**
- **DB\_ *n* K\_CACHE\_SIZE can be set to zero if there is no online tablespace with the *n* KB block size.**
- **The DB\_CACHE\_SIZE parameter replaces the DB\_BLOCK\_BUFFERS parameter.**
- **The DB\_CACHE\_SIZE parameter specifies the size of the cache of standard block size buffers.**

ORACLE

3-31

Copyright © Oracle Corporation, 2001. All rights reserved.

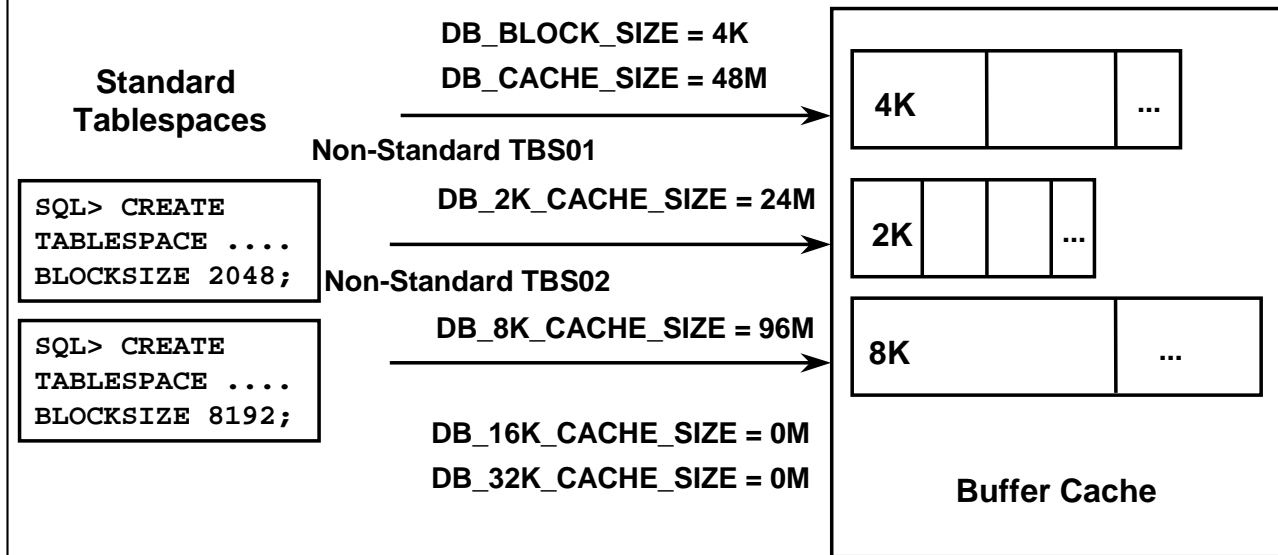
### Nonstandard Block Sizes

Each subcache is sized independently of the others, and can be specified in KB, MB, or GB. The default size of DB\_ *n* K\_CACHE\_SIZE is 0M. The value of the DB\_ *n* K\_CACHE\_SIZE parameter can be set to zero, if there is no online tablespace with the *n* KB block size. In addition, DB\_ *n* K\_CACHE\_SIZE can be used only if *n* is not the standard block size, otherwise an ORA-380 error is generated:

```
ORA-380: "cannot specify db_%sk_cache_size since %sK is the
standard block size"
```

These parameters are dynamic, therefore their values can be changed without shutting down and re-starting the database.

# Multiple Block Size Support



ORACLE

3-32

Copyright © Oracle Corporation, 2001. All rights reserved.

## Multiple Block Size Support

Each Oracle9i database supports a standard block size and up to four additional non-standard block sizes. These nonstandard block sizes can have any power-of-two value between 2 KB and 32 KB (that is, 2 KB, 4 KB, 8 KB, 16 KB, or 32 KB). Subcaches can be configured within the buffer cache for each of these block sizes.

### Standard Block Size

The standard block size is set during database creation using the `DB_BLOCK_SIZE` parameter. It cannot be changed subsequently without re-creating the database. The standard block size is used for the `SYSTEM` tablespace. The `DEFAULT` buffer pool for the standard block size is defined with the `DB_CACHE_SIZE` parameter. The minimum size equals the size of one granule, 4 MB or 16 MB. The default value is 48 MB.

### Nonstandard Block Size

To enable the usage of nonstandard block sizes in the database, the corresponding caches need to be configured.

- `DB_2K_CACHE_SIZE` is used for 2 KB blocks
- `DB_4K_CACHE_SIZE` is used for 4 KB blocks
- `DB_8K_CACHE_SIZE` is used for 8 KB blocks
- `DB_16K_CACHE_SIZE` is used for 16 KB blocks
- `DB_32K_CACHE_SIZE` is used for 32 KB blocks

## Sizing the KEEP and RECYCLE Buffer Caches

- The KEEP and RECYCLE buffer caches can be created only for the standard block size.
- These caches are sized using the following parameters:
  - DB\_KEEP\_CACHE\_SIZE
  - DB\_RECYCLE\_CACHE\_SIZE
- The BUFFER\_POOL\_KEEP and BUFFER\_POOL\_RECYCLE parameters are deprecated.
- DB\_CACHE\_SIZE does not include the size of KEEP and RECYCLE.

ORACLE

3-33

Copyright © Oracle Corporation, 2001. All rights reserved.

### Sizing the KEEP and RECYCLE Buffer Caches

The BUFFER\_POOL\_KEEP and BUFFER\_POOL\_RECYCLE parameters are deprecated, in favor of DB\_KEEP\_CACHE\_SIZE and DB\_RECYCLE\_CACHE\_SIZE. It is not possible to use old and new buffer parameters simultaneously; if you do use them together, the following error is generated:

```
ORA-381: Cannot use both new and old parameters
        for buffer cache size specifications.
```

## Sizing the KEEP and RECYCLE Buffer Caches

DB\_CACHE\_SIZE 48M

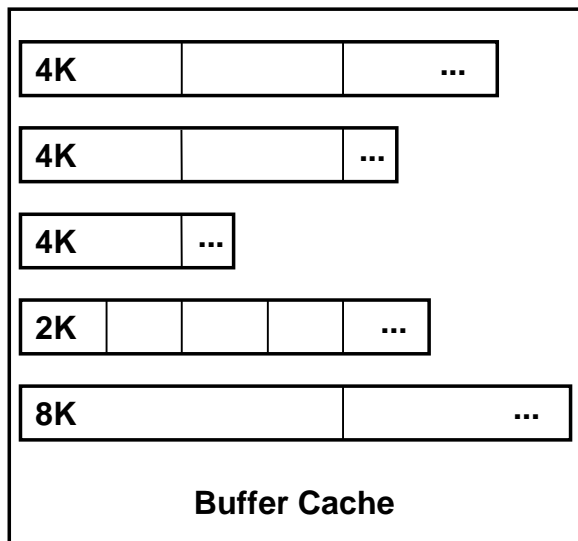
DB\_KEEP\_CACHE\_SIZE 24M

DB\_RECYCLE\_CACHE\_SIZE 12M

DB\_2K\_CACHE\_SIZE 24M

DB\_8K\_CACHE\_SIZE 96M

-----  
Total 204M



ORACLE



## **Automatic Segment Free Space Management in Locally Managed Tablespaces**

- **New approach to managing free space inside the database segments**
- **Free and used space is tracked using bitmaps**
- **A bitmap describes how full each block in the segment is**
- **The map is contained in a separate set of blocks referred to as bitmapped blocks (BMBs)**

ORACLE

3-35

Copyright © Oracle Corporation, 2001. All rights reserved.

### **Automatic Segment Free Space Management in Locally Managed Tablespaces**

This feature introduces a new approach to managing space within a segment. With this new feature, a segment's free and used space is tracked using bitmaps as opposed to free lists. A bitmap is a map which describes the status of each block in the segment with respect to how full it is. The map is contained in a separate set of blocks.

# Automatic Segment Free Space Management

- **Methodology:**
  - When a new row is inserted, the map is searched for a block with sufficient space.
  - When the block becomes more full (or less full), its state is reflected in the bitmap.
- **Benefits include:**
  - Ease of management
  - Better space utilization
  - Better performance for concurrent INSERT operations

ORACLE

3-36

Copyright © Oracle Corporation, 2001. All rights reserved.

## Methodology

When a new row is inserted, the map is searched for a block with sufficient space. When the block becomes more full or less full, its new state is reflected in the bit map.

### **Ease of Management**

The use of PCTUSED, FREELISTS, and FREELIST GROUPS is no longer required, making tuning of space within a segment easier to manage.

### **Better Space Utilization**

This is true especially for objects with rows which vary widely in size because blocks are made available for further row inserts automatically and not based on a fixed percentage of the block size (as with PCTUSED).

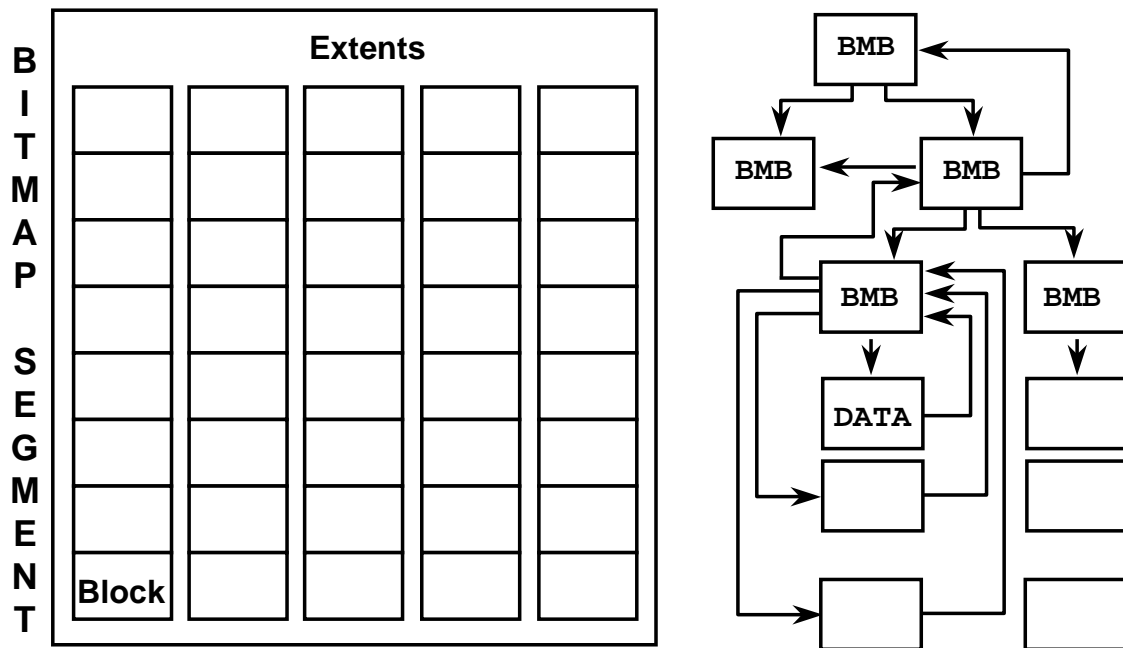
### **Better Concurrency Handling**

Better run-time adjustment to variations in concurrent access through the use of hash algorithms (discussed on the next page).

### **Better Performance**

Improved multiinstance behavior in terms of performance and space utilization. This is also based on a hashing algorithm rather than by assigning users to a free list group based on the instance number.

## Automatic Segment Free Space Management at Work



ORACLE

3-37

Copyright © Oracle Corporation, 2001. All rights reserved.

### Automatic Segment Free Space Management at Work

Automatic Space Management segments have a set of bitmap blocks (BMBs) describing the space utilization of the data blocks in that segment. For each data block, the bitmap indicates if the block is full and, if not, what percentage of space is available.

BMBs are organized in a tree hierarchy. The root level of the hierarchy, which stores the references to all intermediate BMBs, is in fact stored in what was called the segment header. An intermediate level of BMBs contain metadata about the BMB leaves reducing the number of BMB blocks that have to be scanned by a process. Each leaf BMB identifies the space information for a set of contiguous data blocks for the segment.

A process uses a hash algorithm based on the instance number and process id to determine which leaf BMB it will use to locate blocks with available space. This reduces contention between Real Application Clusters instances and processes on the same instance.

During an INSERT statement, the process determines which BMB blocks it should use, through the hash algorithm, then starts at the root level BMB and traverses the tree to locate a BMB that points to data blocks containing free space. These BMBs are acquired in shared mode so that multiple processes can search the same BMB.

## Creating a Segment with Automatic Segment Free Space Management

- **The Automatic Segment Free Space Management is enabled at the tablespace level:**

```
SQL> CREATE TABLESPACE tbs_1
2  DATAFILE '$ORACLE_HOME/dbs/tbs1.dbf'
3  SIZE 10M EXTENT MANAGEMENT LOCAL SEGMENT
4  SPACE MANAGEMENT AUTO;
```

- **Subsequent segments are created with automatic segment free space management.**
- **PCTUSED, FREELIST, and FREELIST GROUPS specifications are ignored.**

ORACLE

3-38

Copyright © Oracle Corporation, 2001. All rights reserved.

### Creating a Segment Managed with Bitmaps

Automatic Segment Free Space Management is enabled at the tablespace level. The tablespace must be of permanent type and locally managed. Bitmapped segments are specified through the `SEGMENT SPACE MANAGEMENT AUTO` clause, which cannot be subsequently altered. The specifications of `PCTUSED`, `FREELIST`, and `FREELIST GROUPS`, if defined, are ignored.

After a tablespace is created with automatic segment free space management, the specification applies to all segments subsequently created in the tablespaces. The `SYSTEM` tablespace is not managed by bitmap segments, because the `SYSTEM` tablespace cannot be locally managed.

Segments that can be managed by bitmaps are heap tables, indexes, IOTs, and LOBs.

## Modifications to Views

- **DBA\_TABLESPACES and USER\_TABLESPACES:**  
**New column called segment\_space\_management**
- **DBA\_TABLES:** **New interpretation for the blocks, empty\_blocks, and pctused columns**
- **DBA\_SEGMENTS:** **Values modified for FREELISTS and FREELIST\_GROUPS**

ORACLE

3-39

Copyright © Oracle Corporation, 2001. All rights reserved.

### **DBA\_TABLESPACES and USER\_TABLESPACES**

A new column, `segment_space_management`, has been added to these two views to indicate whether the segment in the tablespace is managed using free lists or automatic segment free space management. This column can have the following values:

- **MANUAL:** Indicates segment space is managed by free lists
- **AUTO:** Indicates segment space is managed by automatic segment free space management

### **DBA\_TABLES**

The interpretation of the following columns have been modified:

- **blocks:** The number of blocks read by a full table scan
- **empty\_blocks:** Remainder of blocks not read by the full table scan
- **pctused:** Value set to null when using automatic segment free space management

### **DBA\_SEGMENTS**

The values for the following columns have been modified:

- **freelists:** Value set to null when using automatic segment free space management
- **freelist\_groups:** Value set to null when using automatic segment free space management

### Example

```
SQL> select tablespace_name, contents, extent_management,  
2         allocation_type, segment_space_management  
3    from   dba_tablespaces;
```

TABLESPACE_NAME	CONTENTS	EXTENT_MAN	ALLOCATIO	SEGMENT
SYSTEM	PERMANENT	DICTIONARY	USER	MANUAL
RBS	UNDO	LOCAL	SYSTEM	MANUAL
INDX	PERMANENT	LOCAL	SYSTEM	MANUAL
TBS_1	PERMANENT	LOCAL	SYSTEM	AUTO

## Migration and Compatibility

- To migrate existing free list segments to automatic segment free space management, use the `ALTER...MOVE`, or `CREATE...AS SELECT` commands.
- To downgrade, tablespaces with automatic segment free space management must be dropped.
- Tablespaces with automatic segment free space management can be found using the following query:

```
SQL> SELECT TABLESPACE_NAME
2     FROM DBA_TABLESPACES
3     WHERE SEGMENT_SPACE_MANAGEMENT = 'AUTO';
```

ORACLE

3-41

Copyright © Oracle Corporation, 2001. All rights reserved.

### Migration and Compatibility

Automatic segment free space management is enabled only if the `COMPATIBLE` parameter is set to 9.0.0 or higher.

In order to convert existing data, you first create tablespaces with automatic segment free space management, and then move the objects into the new tablespaces. Existing mechanisms, such as `ALTER . . . MOVE` and `CREATE . . . AS SELECT`, can be used for this purpose. They work automatically, because the segments inherit automatic segment free space management from the tablespace in which they are created.

### Downgrading

In practice, the existing objects (if any) must be moved to tablespaces with freelist segment management, because no explicit migration is provided from the automatic free space management format to the freelist format.

# Summary

**In this lesson, you should have learned how to:**

- **Create and manage Oracle-managed files**
- **Remove associated operating-system files when removing a non-OMF tablespace**
- **Create and alter default temporary tablespaces**
- **Manage undo tablespaces**
- **Manage tablespaces with multiple block sizes**
- **Specify segment space management**

ORACLE



# 4

## Managing Resumable Space Allocation

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# Objectives

**After completing this lesson, you should be able to:**

- **Identify suspendable and resumable operations**
- **Enable and disable resumable sessions**
- **Detect suspended statements:**
  - Using `ALERT.log`
  - Using a trigger
- **Monitor resumable statements**

ORACLE

## Resumable Space Allocation

- **With resumable space allocation you can suspend, fix, and resume the execution of large operations when an out of space error occurs.**
- **Benefits include:**
  - **Support for errors related to space limits and out-of-space conditions**
  - **Opportunity to take the corrective steps to resolve errors**
  - **Automatically continues operation after the problem is fixed**

ORACLE

4-3

Copyright © Oracle Corporation, 2001. All rights reserved.

### Resumable Space Allocation

Large operations routinely run out of space or reach space limits after being executed for a long time. When this condition occurs, the operation is rolled back and an error is returned to the user.

With the resumable space allocation feature, the Oracle server suspends these large operations that run into correctable problems. Suspending the operation gives the database administrator an opportunity to take the corrective steps to resolve the error condition. After the error condition disappears, the suspended operation automatically resumes execution.

# Resumable Space Allocation

**Resumable operation is suspended under one of the following conditions:**

- **Out of space errors, for example:**

```
ORA-1653: unable to extend table ...  
         by ... in tablespace ...
```

- **Space limit errors, for example:**

```
ORA-1631: max # extents reached in table ...
```

- **Space quota errors, for example:**

```
ORA-1536: space quota exceeded for  
tablespace ...
```

ORACLE

## Correctable Errors

There are three classes of correctable errors, as shown on the slide.

The following list shows additional errors as a result of which a resumable operation can be suspended:

- ORA-1562: No more space in tablespace; failure to extend rollback segment
- ORA-1628: Maximum extents reached for rollback segment
- ORA-1630: Maximum extents reached for temp segment
- ORA-1631: Maximum extents reached for table
- ORA-1632: Maximum extents reached for index
- ORA-1650: No more space in tablespace, that is, the system cannot extend rollback segment
- ORA-1652: No more space in tablespace, that is, the system cannot extend temporary segment
- ORA-1653: No more space in tablespace, that is, the system, cannot extend table

### **Correctable Errors (continued)**

- ORA-1654: No more space in tablespace, that is, the system cannot extend index
- ORA-1655: No more space in tablespace, that is, the system cannot extend cluster
- ORA-1656: Maximum extents reached in cluster
- ORA-1683: No more space in tablespace, that is, the system cannot extend index partition
- ORA-1684: Maximum extents reached in table partition
- ORA-1685: Maximum extents reached in index partition
- ORA-1688: No more space in tablespace, that is, the system cannot extend table partition.
- ORA-1691: No more space in tablespace, that is, the system cannot extend LOB
- ORA-1692: No more space in tablespace, that is, the system cannot extend LOB partition
- ORA-1693: Maximum extents reached in LOB segment
- ORA-1694: Maximum extents reached in LOB segment partition
- ORA-3233: No more space in tablespace, that, is the system cannot extend table subpartition.
- ORA-3234: No more space in index subpartition, that is, the system cannot extend index subpartition
- ORA-3235: Maximum extents reached in table subpartition.
- ORA-3236: Maximum extents reached in index subpartition.
- ORA-3238: No more space in tablespace, that, is the system cannot extend LOB subpartition
- ORA-3239: Maximum extents reached in LOB subpartition
- ORA-1536: Space quota exceeded in tablespace.

## Enabling and Disabling Resumable Space Allocation Mode

- Enable or disable resumable space allocation:

```
ALTER SESSION {ENABLE RESUMABLE  
               [TIMEOUT timeout][NAME name]  
               |DISABLE RESUMABLE}
```

- Example:

```
SQL> ALTER SESSION DISABLE RESUMABLE;
```

```
SQL> ALTER SESSION ENABLE RESUMABLE NAME  
      2 'insert from table' TIMEOUT 5400;
```

- New system privilege: RESUMABLE

ORACLE

### Enabling and Disabling Resumable Space Allocation Mode

The new `ALTER SESSION ENABLE RESUMABLE [TIMEOUT timeout] [NAME name]` statement enables SQL statements to be resumable when they are invoked within this session.

`ALTER SESSION DISABLE RESUMABLE` disables the resumability of subsequent SQL statements invoked in the same session.

A suspended resumable operation is aborted automatically if the error is not fixed within *timeout* seconds (the default is 2 hours; the maximum is 24,855 days). *name* is a user-defined text string which is inserted to help users to identify a specific resumable operation.

### RESUMABLE System Privilege

The RESUMABLE system privilege is added to allow a database administrator to decide who can execute resumable operations. Suspending an operation automatically results in suspending the transaction; therefore, all transactional resources are preserved while the operation is suspended and resumed.

# Altering Resumable Space Allocation

- **Alter the timeout by using ALTER SESSION or DBMS\_RESUMABLE:**

```
ALTER SESSION {ENABLE|DISABLE} RESUMABLE  
TIMEOUT timeout
```

```
DBMS_RESUMABLE.SET_SESSION_TIMEOUT  
(sessionID, timeout)
```

```
DBMS_RESUMABLE.SET_TIMEOUT(timeout)
```

- **Alter the name within a session:**

```
ALTER SESSION {ENABLE|DISABLE} RESUMABLE  
NAME 'new name'
```

ORACLE

4-7

Copyright © Oracle Corporation, 2001. All rights reserved.

## Altering Resumable Space Allocation

The default timeout is two hours, and it can be set to any value by using the ALTER SESSION ENABLE RESUMABLE TIMEOUT *timeout period in seconds* command, or by calling upon the new DBMS\_RESUMABLE package.

### DBMS\_RESUMABLE Package

The DBMS\_RESUMABLE package helps to control the resumable operation by means of the following procedures:

- ABORT(*sessionID*): Aborts a suspended resumable operation
- GET\_SESSION\_TIMEOUT(*sessionID*): Returns the current timeout value in seconds for resumable operations in the session with that session ID
- SET\_SESSION\_TIMEOUT(*sessionID*, *timeout*): Sets a new timeout setting immediately for resumable operations in the session with that session ID
- GET\_TIMEOUT( ): Returns the current timeout value for resumable operations in the current session
- SET\_TIMEOUT(*timeout*): Sets the timeout for resumable operations in the current session

**Note:** It is not possible to use this package to enable resumable operation in a particular session. Instead, you can create a logon trigger that enables this feature by using an ALTER SESSION command.

# Life Cycle of Resumable Space Allocation

1. Resumable space allocation is enabled.
2. Operation begins.
3. Out-of-space error occurs.
4. Error is written to `ALERT.log`.
5. Error is fixed; for example, space is added to the tablespace.
6. Suspended operation resumes automatically.
7. Operation ends successfully.

ORACLE

4-8

Copyright © Oracle Corporation, 2001. All rights reserved.

## Life Cycle of Resumable Space Allocation

1. The resumable space allocation feature is enabled by using the `ALTER SESSION ENABLE RESUMABLE` command.
2. A resumable operation begins.
3. The resumable operation is suspended under one of the following conditions: An out-of-space occurs, the maximum extents are reached, or the space quota is exceeded.
4. The error is reported in the `ALERT.log` file.
5. The error condition is fixed.
6. The suspended operation resumes automatically.

**Note:** A resumable operation may be suspended and resumed multiple times during its execution. A DBA can abort a suspended operation at any time using `DBMS_RESUMABLE.ABORT( )` or `ALTER SESSION KILL`.

Keep in mind that `DBMS_RESUMABLE.ABORT( )` only aborts the suspended operation. `ALTER SESSION KILL` kills the session.

7. The operation is completed.



# Which Operations are Resumable?

- **Queries**
- **DML statements**
- **SQL\*Loader operations**
- **Import and Export operations**
- **DDL statements**

ORACLE

## Resumable Operations

The following operations are resumable:

- **Queries:** `SELECT` statements that run out of temporary space (for sort areas) are candidates for resumable execution. When using OCI, the `OCIStmtExecute()` and `OCIStmtFetch()` calls are resumable.
- **DML statements:** `INSERT`, `UPDATE`, and `DELETE` operations may be resumable. It does not matter whether the interface used to execute them is OCI, SQLJ, or PL/SQL. Also, `INSERT` and `SELECT` statements from external tables may be resumable.
- **SQL\*Loader operations:** SQL\*Loader, when invoked with the resumable option, performs its operations as resumable operation. The option can also be specified in the SQL\*Loader control file. When the resumable option is not specified and a correctable error occurs, SQL\*Loader commits all data until and excluding the failed row insert. When the resumable option is specified, SQL\*Loader is instead suspended and the error is generated using the resumable notification infrastructure.
- **Import and Export operations:** Resumable operations are an option in both Import and Export, as in SQL\*Loader.
- **DDL statements:** Resumable DDL statements include `CREATE TABLE AS SELECT`, `CREATE INDEX`, `ALTER INDEX REBUILD`, `ALTER TABLE MOVE PARTITION`, `ALTER TABLE SPLIT PARTITION`, `ALTER INDEX REBUILD PARTITION`, `ALTER INDEX SPLIT PARTITION`, `CREATE MATERIALIZED VIEW`, and so on.

## Detecting Suspended Statements: The ALERT.log File

```
Tue Jul 03 15:37:48 2001
statement in resumable session 'User RES(68), Session
15, Instance 1' was suspended due to ORA-01631: max #
extents (18) reached in table RES.DEST
```

```
Tue Jul 03 15:38:38 2001
statement in resumable session 'User RES(68), Session
15, Instance 1' was resumed
```

```
Tue Jul 03 15:38:41 2001
...
```

ORACLE

## Detecting Suspended Statements: The After Suspend System Event

- An After Suspend trigger is fired when a resumable operation encounters a correctable error.
- SQL statements executed within an After Suspend trigger are always nonresumable.

```
CREATE OR REPLACE TRIGGER RES_DEFAULT
AFTER SUSPEND ON DATABASE
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
    /* SEND AN EMAIL TO NOTIFY DBA */
    COMMIT;
END;
```

ORACLE

4-11

Copyright © Oracle Corporation, 2001. All rights reserved.

### After Suspend System Event

A user can create a trigger on the After Suspend system event. Whenever a resumable operation is suspended, the After Suspend trigger is executed. Inside the trigger, the user can access information regarding the suspended operation from the DBMS\_RESUMABLE package.

The trigger must be declared as an autonomous transaction. The autonomous transaction uses an undo segment in the SYSTEM tablespace. This is done to reduce the chance of running into the same error as the user statement.

The trigger may encounter the following errors:

- Deadlock with a lock held by the user transaction, which is executing the resumable operation. This may happen if the trigger attempts to update a row locked by the user transaction. The Oracle server detects the deadlock, rolls back work done in the trigger, and generates the original exception to the user. Thus, if trigger deadlocks (self-locks) with the user transaction, the result is that the operation is not suspended.
- Out of space or space limit reached condition. If a trigger generates these errors, it is not suspended; instead, the trigger and the resumable operation are aborted.

**Note:** You can place an exception handler inside the trigger to clear the error, so that the suspended operation is not aborted.

# Monitoring Resumable Space Allocation

- **Two views are created to support resumable space allocation:**
  - **USER\_RESUMABLE:** Lists the resumable operation being executed by the user
  - **DBA\_RESUMABLE:** Lists and monitors the progress on all resumable operations being executed
- **Contains all currently executing or suspended resumable operations**

ORACLE

4-12

Copyright © Oracle Corporation, 2001. All rights reserved.

## Columns in the USER\_RESUMABLE and DBA\_RESUMABLE Views

Note that the information in these views is not persistent.

- **user\_id:** ID number of resumable operation owner (in DBA\_RESUMABLE only)
- **session\_id:** Session ID of resumable operation
- **instance\_id:** Instance ID of resumable operation
- **coord\_session\_id:** Session ID of Parallel Coordinator
- **coord\_instance\_id:** Instance ID for running Parallel Coordinator
- **sql\_text:** Resumable operation selected from V\$SQL view
- **name:** Name given in the resumable operation clause to identify resumable operation
- **status:** Can be RUNNING, SUSPENDED, ABORTING, ABORTED, or TIMEOUT
- **error\_number:** Error code of the last correctable error
- **error\_msg:** Error message corresponding to error\_number
- **error\_parameter1 – 5:** First through fifth parameter for the error message; null if there is no error.
- **start\_time:** Start time of the transaction which invoked the resumable operation

# Monitoring Resumable Space Allocation

```
SQL> SELECT session_id,timeout,start_time,  
2          name,sql_text  
3 FROM      dba_resumable;  
  
SESSION_ID TIMEOUT START_TIME  
-----  
NAME  
-----  
SQL_TEXT  
-----  
          9      3600 03/13/01 12:32:54  
User RES(68), Session 9, Instance 1  
insert into dest select /* full(a) full(b) */ a.n,a.s  
from src a,src b where a.n!=b.n and rownum<1000000
```

ORACLE

## Monitoring Resumable Space Allocation


- suspend\_time: Last time the resumable operation was suspended
- resume\_time: Last time the resumable operation was resumed
- timeout: Timeout of resumable operation

# Summary

**In this lesson, you should have learned how to:**

- **Identify suspendable and resumable operations**
- **Enable and disable resumable sessions**
- **Detect suspended statements:**
  - Using `ALERT.log`
  - Using an After Suspend trigger
- **Monitor resumable statements**

ORACLE



# **Database Resource Manager Enhancements**

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# Objectives

**After completing this lesson, you should be able to:**

- **Set an active session pool for a consumer group**
- **Set a maximum estimated execution time for each consumer group**
- **Set parameters to automatically switch the consumer group of a running session**
- **Set a maximum limit on the total amount of undo information generated per resource consumer group**

ORACLE



## Database Resource Manager Overview

- Gives more control over resource management decisions
- Helps circumvent problems resulting from inefficient operating system management
- Is implemented by `DBMS_RESOURCE_MANAGER` and the `DBMS_RESOURCE_MANAGER_PRIVS` package
- The `ADMINISTER_RESOURCE_MANAGER` privilege is required.

ORACLE

5-3

Copyright © Oracle Corporation, 2001. All rights reserved.

### Database Resource Manager

The main goal of the Database Resource Manager is to give the Oracle database server more control over resource management decisions, thus circumventing problems resulting from inefficient operating system management. The Database Resource Manager was provided in Oracle8i and is now enhanced in Oracle9i.

## Active Session Pool

- **Allows you to define the number of concurrent active sessions per resource consumer group.**
- **Benefits of an active session pool:**
  - **Allows database administrators to meet performance objectives by limiting the concurrent system workload**
  - **Reduces the number of servers using resources in the system**

ORACLE

5-4

Copyright © Oracle Corporation, 2001. All rights reserved.

### Active Session Pool

Historically, the Oracle Database Resource Manager provided the mechanism to partition CPU resources. These resources were typically allocated at the beginning of a transaction or query and not freed until the transaction was committed or the query was finished. There was no way to free the resources associated with such operations until the operation was complete. As a result, newly arriving operations either significantly worsened system performance, or caused individual operations to generate errors when unable to allocate a resource such as a temporary space.

The active session pool allows the DBA to control the maximum number of concurrently active sessions per resource consumer group by defining an active session pool.

### Benefits of the Active Session Pool

With this functionality, you can indirectly control the amount of resources that any resource consumer group uses, because resource consumption is proportional to the number of active sessions. When the active session pool is filled, the Database Resource Manager queues all subsequent requests, and runs them only after the existing active sessions are completed.

The main goal of the active session pool is to reduce the number of servers using resources in the system, thus avoiding inefficient paging, swapping, and other types of resource depletion (memory, temporary space, and so on) which result from attempting to run too many jobs simultaneously. In essence, it is an attempt to guarantee minimum resources for an operation, and to establish limits on the resource consumer group.

## Active Session Pool Mechanism

- **You set an active session pool size per resource consumer group:**
  - The active session pool is the maximum number of concurrently active sessions.
  - An active session is defined as a session currently part of an active transaction, query, or parallel operation.
  - Only one active session pool size is allowed per consumer group.
- **When the pool is filled with active sessions, all subsequent sessions are queued.**

ORACLE

5-5

Copyright © Oracle Corporation, 2001. All rights reserved.

### Active Session Pool Queuing Method

When the active session pool is filled with active sessions, the Database Resource Manager queues all subsequent sessions attempting to become active until other active sessions are completed or become inactive.

In Oracle9i, there is only one queue per resource consumer group, and the queuing method is first in, first out (FIFO) with a timeout.

## Active Session Pool Parameters

- **The active session pool is defined by setting the following parameters:**
  - **ACTIVE\_SESSION\_POOL\_P1**
  - **QUEUEING\_P1**
- **ACTIVE\_SESSION\_POOL\_P1: Identifies the number of active sessions that establishes the threshold of the resource consumer group**
- **QUEUEING\_P1: Indicates how long any session can wait on the queue before the current operation is aborted**

ORACLE

5-6

Copyright © Oracle Corporation, 2001. All rights reserved.

### Active Session Pool Parameters

When you create a resource plan directive, you can specify the following parameters:

- **ACTIVE\_SESSION\_POOL\_P1:** Identifies the number of active sessions that establishes the threshold of the resource consumer group, and therefore its active session pool
- **QUEUEING\_P1:** This parameter is optional, and indicates how long any session can wait on the queue. If a session waits on the consumer group's queue for longer than the specified value, the session is aborted with an error.

The default value for both is UNLIMITED.

## Setting the Active Session Pool

### Example:

GROUP	ACTIVE SESSION POOL
OLTP	
BATCH	ACTIVE_SESSION_POOL_P1 = 5 QUEUEING_P1 = 600

- **OLTP: No limit on concurrent active sessions**
- **BATCH: Limit of five concurrent active sessions, and to all sessions waiting on the queue for more than ten minutes are aborted**

ORACLE

### Example of Setting the Active Session Pool

In the above example, the OLTP resource consumer group has no limit on the number of concurrent active sessions, because the `ACTIVE_SESSION_POOL_P1` parameter was not set.

The BATCH resource consumer group has a limit of five concurrent active sessions (the `ACTIVE_SESSION_POOL_P1` parameter is set to 5). The BATCH group also has the `QUEUEING_P1` parameter set to 600 seconds (ten minutes). All sessions waiting on the queue for more than ten minutes are aborted with an error.

## Maximum Estimated Execution Time

- **Estimates the execution time of an operation proactively**
- **You can specify a maximum estimated execution time for an operation at the resource consumer group level.**

**MAX\_ESTIMATED\_EXEC\_TIME:** Maximum estimated time an operation can take. The operation will not start if estimate is longer than the value of this parameter.
- **The benefit of this feature is the elimination of exceptionally large jobs that use too many system resources.**

ORACLE

5-8

Copyright © Oracle Corporation, 2001. All rights reserved.

### Maximum Estimated Execution Time

The Database Resource Manager in Oracle9i can estimate the execution time of an operation proactively. You can define the maximum estimated execution time any operation can take at any given time by setting the `MAX_ESTIMATED_EXEC_TIME` resource plan directive parameter. When this parameter is set, the Database Resource Manager estimates the time a specific job will take. If the estimate is more than the value of `MAX_ESTIMATED_EXEC_TIME`, then the operation does not start, and the exceptionally large job that would utilize too many system resources is eliminated. The default value is `UNLIMITED`.

If a resource consumer group has more than one plan directive referring to it, then the group may have more than one `MAX_ESTIMATED_EXEC_TIME`. The Database Resource Manager then chooses the most restrictive of all incoming values.

# Automatic Consumer Group Switching

- **You can specify the resource plan directives:**
  - **SWITCH\_GROUP:** Group to which you have switched
  - **SWITCH\_TIME:** Time (in seconds) within which a session must execute before it is switched to another consumer group
  - **SWITCH\_ESTIMATE:** If TRUE, the execution time estimate is used to decide whether to switch an operation even before it starts

If **SWITCH\_TIME** or **SWITCH\_ESTIMATE** is set, a session's consumer group is switched automatically.
- **The benefit of this feature is that it can limit the resources consumed by long-running operations.**

ORACLE

## Automatically Changing Resource Manager Groups

The Database Resource Manager switches a running session to the group defined by the **SWITCH\_GROUP** parameter if the session is active for more than the number of seconds defined by the **SWITCH\_TIME** parameter. *Active* means the session is running and consuming resources, not waiting idly for user input or waiting for CPU cycles. When the session finishes its operation and becomes idle, it is switched back to its original group.

If a resource consumer group has more than one plan directive referring to it, it may have more than one **SWITCH\_GROUP** and **SWITCH\_TIME**. The Database Resource Manager chooses the most restrictive of all incoming values.

The default value of **SWITCH\_GROUP** is null, and that of **SWITCH\_TIME** is **UNLIMITED**.

If **SWITCH\_ESTIMATE** is set to **TRUE**, the Database Resource Manager uses a predicted estimate of how long the operation will take to complete in order to decide whether to switch a session before an operation even starts running. If this parameter is not set, the operation just starts running, and switches groups only if the other switch criteria are met. The default value is **FALSE**.

You can use this feature to manage the workload better by segregating long-running batch jobs from short OLTP transactions. Batch jobs can be automatically assigned to consumer groups with lower resource allocation during the day, thereby improving the response time for OLTP users.

# Undo Quota

- **Allows you to manage the undo space consumed by long-running transactions**
- **New resource plan directive parameter: UNDO\_POOL**
- **Quota of undo space is defined per resource consumer group**
- **Whenever the total undo space is exceeded, no further INSERT, UPDATE, or DELETE operations are allowed.**

ORACLE

5-10

Copyright © Oracle Corporation, 2001. All rights reserved.

## Undo Quota

Long transactions and improperly written transactions can consume valuable resources. Therefore, the Database Resource Manager now enables a DBA to manage the undo space consumed by long-running transactions.

This feature uses a new resource plan directive parameter called UNDO\_POOL. The undo pool is specified in KB, and the default value is UNLIMITED. You define a quota of undo space per resource consumer group. Whenever the total undo space is exceeded, no further INSERT, UPDATE, or DELETE operations are allowed, and the following error is displayed:

```
ORA-30027: Undo quota violation -  
failed to get %s (bytes)
```

You must increase the undo quota or wait for other transactions to be committed before proceeding.

When undo space is freed by another session in the same group, or the undo quota is increased, the DML operations begin again.



## Undo Quota

- **SELECT statements are allowed even when the UNDO\_POOL limit is exceeded.**
- **The default value for the UNDO\_POOL is UNLIMITED.**
- **UNDO\_POOL is specified in KB.**
- **The UNDO\_POOL can be used with Automatic Undo Management as well as with Manual Undo Management.**

ORACLE

## Changing the Undo Quota

- The undo quota can be created using the following resource plan directive:

```
EXECUTE dbms_resource_manager.create_plan_directive
(plan          => 'bugdb_plan'
,group_or_subplan => 'support', comment => 'comment'
,undo_pool      => 10000);
```

- The undo quota can be changed at any time during database operation using the following resource plan directive:

```
EXECUTE dbms_resource_manager.update_plan_directive
(plan          => 'bugdb_plan'
,group_or_subplan => 'support'
,new_undo_pool  => 50000);
```

ORACLE

### Changing the Undo Quota

In the first example, the resource plan `bugdb_plan` limits undo usage to 10,000 Kbytes for the `support` group. In the second example, the value is changed to 50,000 Kbytes.

## Modified Views to Support Database Resource Manager Extensions

```
SQL> select name, current_undo_consumption  
2 from V$RSRC_CONSUMER_GROUP;
```

NAME	CURRENT_UNDO_CONSUMPTION
SYS_GROUP	0
OTHER_GROUPS	1320
LOW_GROUP	0

ORACLE

5-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### Modified Views to Support Database Resource Manager Extensions

The following column has been added to the V\$SESSION and V\$MYSESSION views:

- **current\_queue\_duration**: The current amount of time in seconds that the session has been queued. If not currently queued, the value is zero.

The following columns have been added to V\$RSRC\_CONSUMER\_GROUP:

- **queue\_length**: Number of sessions waiting on the queue
- **current\_undo\_consumption**: Current amount of undo space consumed by the consumer group

In addition, the **queuing\_mth** column has been added to the DBA\_RSRC\_PLANS view. This column defines the queuing resource allocation method for the plan.

# Summary

**In this lesson, you should have learned how to:**

- **Set parameters to define an active session pool**
- **Set a maximum estimated execution time for each consumer group**
- **Set parameters to automatically switch the consumer group of a running session**
- **Set an undo pool size for each resource consumer group**

ORACLE

# 6

## Enterprise Manager Enhancements

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# Objectives

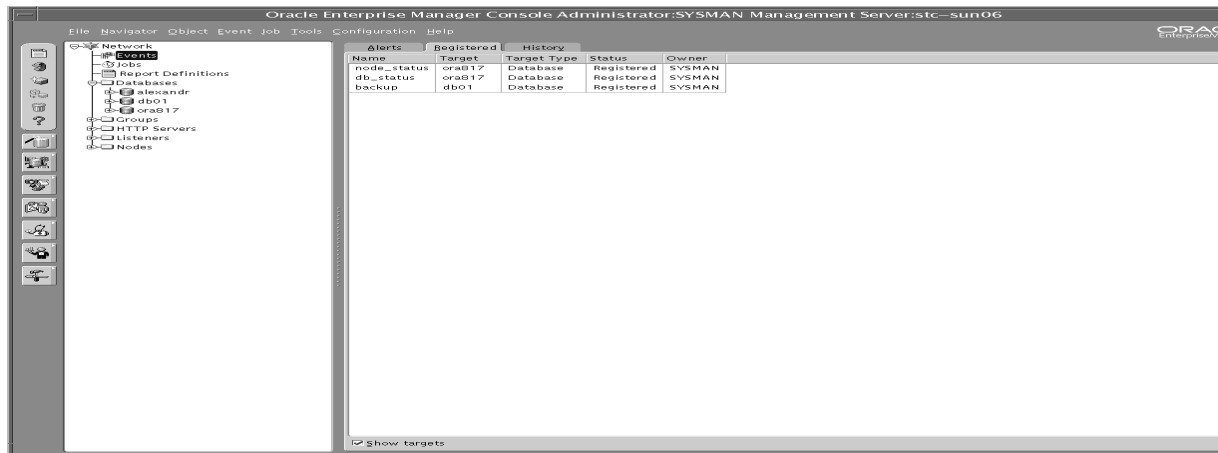
**After completing this lesson, you should be able to:**

- **Describe the new look and feel of the Enterprise Manager Console**
- **Use the Console in Standalone mode**
- **Use management regions**
- **Explain the Enterprise Manager functionality which supports Oracle9i database features**
- **Generate HTML reports**
- **Use user-defined events**

ORACLE

## New Console Look-and-Feel

- Two pane, master/detail view of the environment
- Events, jobs, and groups functionality integrated in the navigator tree
- Database administration features fully integrated with the Console



6-3

Copyright © Oracle Corporation, 2001. All rights reserved.

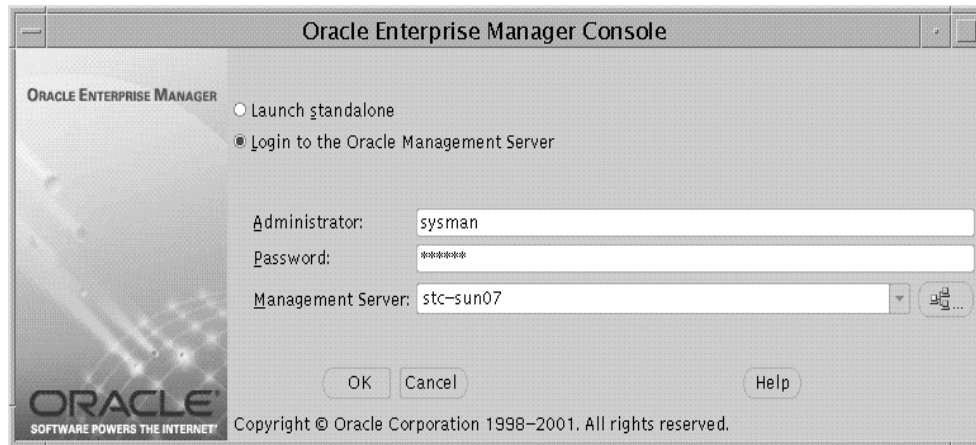
ORACLE

### New Console Look and Feel

When you click an item in the navigator tree, the details of what has been selected appear on the right-hand side of the console.

For example, when you click Events in the navigator tree on the left, the detail side shows the Alerts, Registered, and History tabs, which contain details of the events for all targets.

# Launching Enterprise Manager Console



## Launching Enterprise Manager Console

When you launch the Console, you choose between stand-alone mode and connecting to the middle-tier Oracle Management Server (OMS).

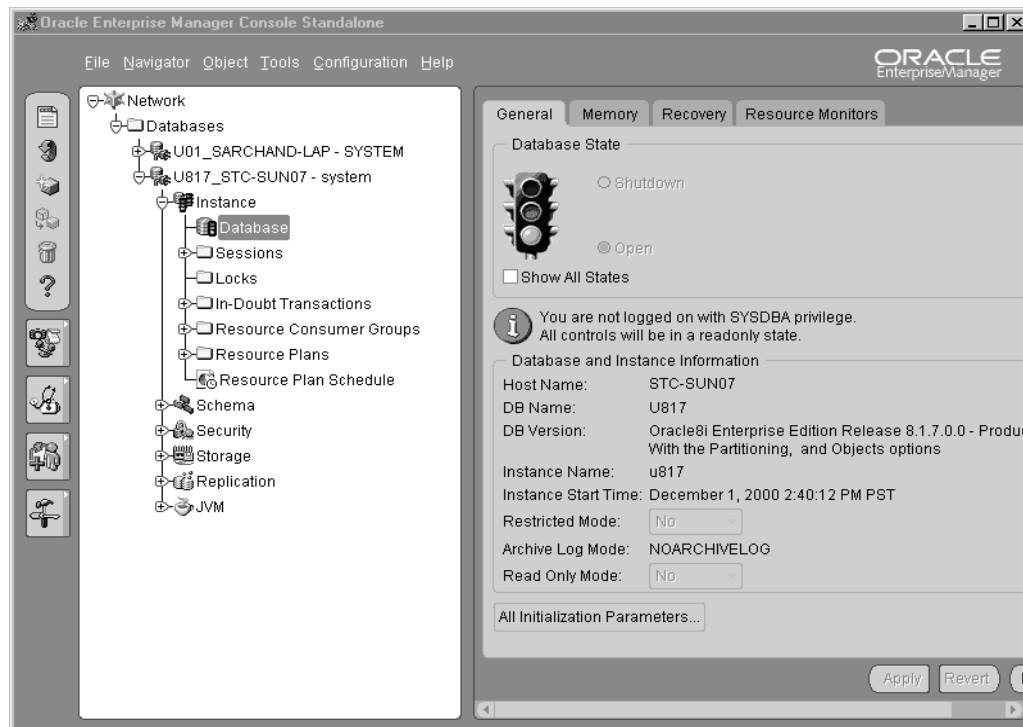
If OMS has not yet been installed or configured, then you can launch the Console in stand-alone mode and connect directly to target databases to perform administrative tasks such as deleting a table or creating a new database user. When launching stand-alone, you can connect only to Oracle databases; no other types of targets are currently supported for Oracle9i.

If OMS has been installed and configured, then you can launch the Console by logging into that Management Server. By logging into a Management Server, you have access to more comprehensive management capabilities.

**Note:** The choice between launching stand-alone or through the Oracle Management Server is also available in other Enterprise Manager applications.



## Console Launched in Standalone Mode



6-5

Copyright © Oracle Corporation, 2001. All rights reserved.

### Console Launched in Standalone Mode

When you launch the Oracle9i Console in Standalone mode, it looks like DBA Studio in release 2.1 or 2.2, but with more features and enhancements.

The slide shows a screenshot of the Console in Standalone mode. The only targets in the navigator tree are databases.

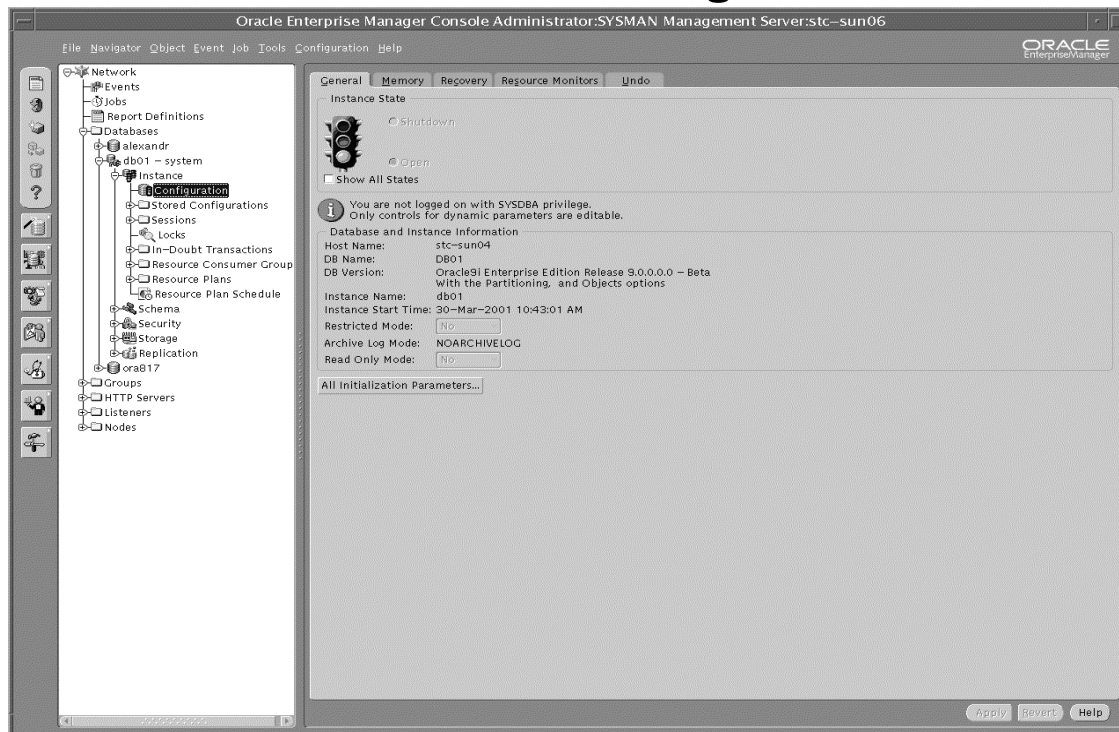
### Adding Databases to the Navigator Tree

You add databases to the navigator tree as follows:

1. Launch the Console in Standalone mode.
2. Use the Add Database to Tree dialog box, which appears automatically when you start the Console in Standalone mode for the first time. It is also available from the Navigator main menu.
3. Add the databases you wish to manage. You can manually enter the Net service names, or add them from the local tnsnames.ora file.

There is no support for discovering nodes with intelligent agents when the Console is launched stand-alone.

# Connections using OMS



6-6

Copyright © Oracle Corporation, 2001. All rights reserved.

## Connections Using OMS

The slide shows a screenshot of the Console connected to the middle-tier Oracle Management Server. There are several types of discovered targets in the navigator tree (databases, HTTP servers, listeners, and so on), as well as definitions of events, jobs, and reports. These additional types of targets and the associated functionality would not be available if the Console were launched in Standalone mode.

## Standalone Connection Benefits

- **Available out of the box**
- **Does not require installing or configuring the middle-tier Management Server**
- **Does not require installing intelligent agent on the administered database**
- **You can connect directly to the managed target**

ORACLE

6-7

Copyright © Oracle Corporation, 2001. All rights reserved.

### Standalone Connection Benefits

Standalone mode enables a single user to use one or more applications without the need of Oracle Management Server or intelligent agent. If you want to perform certain administrative tasks that do not require the job, event, or group system, the stand-alone Console can be used.

## Standalone Connection Restrictions

- Only supports direct administration of Oracle databases (no management of other targets)
- No access to events, jobs, and groups
- Cannot share administrative information
- Cannot use Web-enabled applications
- No paging
- No e-mail blackouts
- Cannot store and access preferences (for example, enhanced notifications or schedule)
- Cannot use backup and data management tools
- Cannot customize, schedule, and publish reports

ORACLE

6-8

Copyright © Oracle Corporation, 2001. All rights reserved.

### Standalone Connection Restrictions

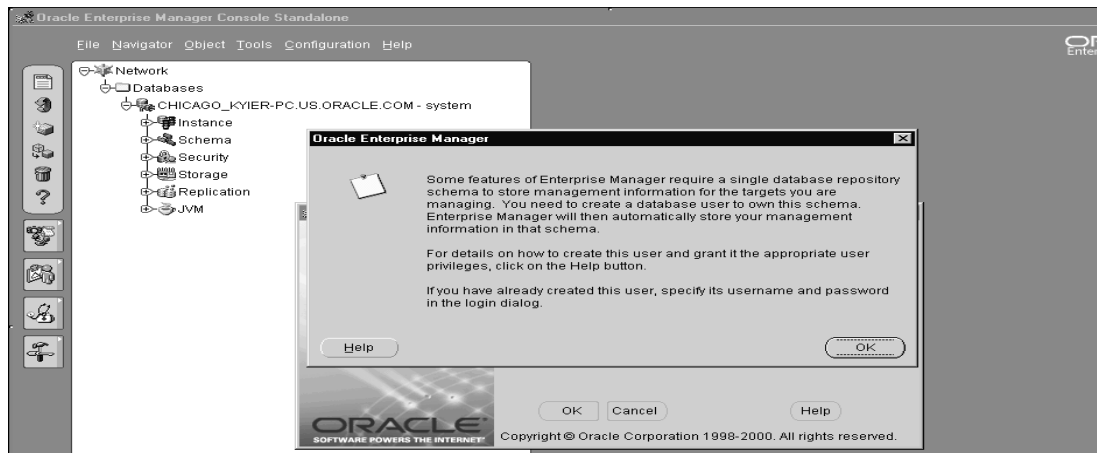
If you need access to any features that are not available in Standalone mode then you must install and configure the middle-tier Management Server and launch the Console through OMS.

Because there is no use of a Management Server and its repository in Standalone mode, information such as paging and blackouts cannot be shared among administrators. Moreover, you cannot launch Enterprise Manager applications using a Web browser.

**Note:** Because only databases are supported in Oracle9i, Management Pack for Oracle Applications and Management Pack for SAP R/3 do not support Standalone mode.

# Standalone Repository

**When launching the Console in Standalone mode, some functionality requires a stand-alone repository.**



## Stand-Alone Repository

A stand-alone repository is a single user database repository schema that stores management information for the targets you are managing while using the Console in Standalone mode.

When launching the Console in Standalone mode and accessing certain Management Pack applications for the first time (such as Oracle Change Manager, Oracle SQL Analyze, Oracle Index Tuning Wizard, Oracle Tablespace Map, and Oracle Expert), you are prompted to create a single-user database repository schema to store management information for the targets being managed. Once this stand-alone repository schema is created, it can be used by all five applications listed above; each application does not require its own stand-alone repository schema.

This single-user database repository schema is separate from the repository that is created when you install and configure a Management Server. This schema is for a single administrator and does not require a Management Server, whereas the repository used by a Management Server is for multiple administrators and is required by the middle tier. Furthermore, interaction between the standalone repository schema and the Management Server repository, including the migration or sharing of repository data, is not supported. Database releases that support an Enterprise Manager release 9i stand-alone repository schema include:

- Enterprise Edition or standard edition, release 9i
- Enterprise Edition or standard edition, release 8.1.7 and 8.1.6
- Enterprise Edition, release 8.0.6 (Objects Option must be installed and enabled)

# Management Regions

- **A management region is a grouping of discovered nodes and management servers.**
- **Multiple management regions can be defined in a single repository.**
- **Only Super Administrators have access to management regions functionality.**
- **Management regions are useful for:**
  - **Global deployments of EM framework**
  - **Deployments across LANs and WANs**
  - **Mapping nodes within firewall boundaries**

ORACLE

6-10

Copyright © Oracle Corporation, 2001. All rights reserved.

## Management Regions

There can be multiple management regions defined in a single repository, but a particular node is a member of one and only one management region. You can assign the management server to be in one and only one management region. However, a management region can contain multiple management servers as long as all the management servers share the same repository.

## Benefits of Management Regions

You use management regions when you have global deployments of the Enterprise Manager framework, or when you have deployments of the framework across a mixture of LANs and WANs. In these two scenarios, management regions can improve performance over slow networks by allowing you to assign a subset of management servers and a subset of discovered nodes to a management region, in order to prevent cross-regional or cross-network communication.

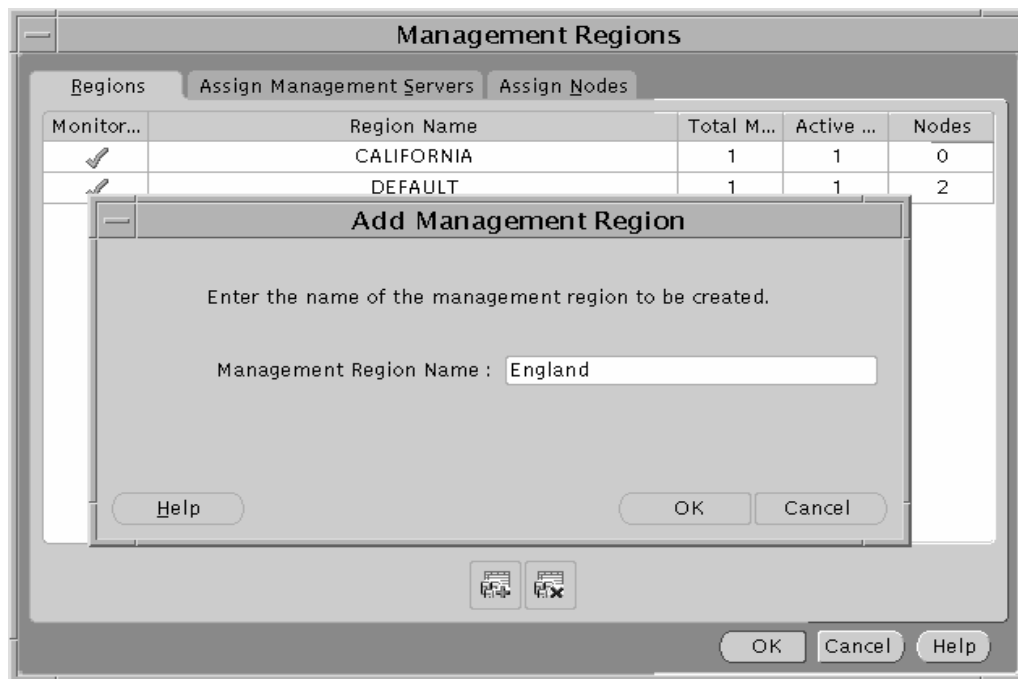
## **Example**

A company has an OMS running in England and monitored targets in England. It also has an OMS running at its headquarters in California, as well as monitored targets in California.

With previous releases of Enterprise Manager, the management server in England could actually monitor the targets in California. There was no way to bind a management server to the targets. Additionally, if there is a firewall between the management server in England and the monitored node in California, the management server and the nodes cannot interact with each other.

Using management regions prevents this cross-regional communication. Now you can specify that the management server in England monitors only the targets in England (within a firewall), not the targets in California.

# Creating a Management Region



ORACLE

6-12

Copyright © Oracle Corporation, 2001. All rights reserved.

## Creating Management Regions

To create or edit management regions, perform the following steps:

1. Select Define Management Regions from the Configuration menu. If you are not a Super Administrator, this menu item does not appear.
2. Click the Add Region icon and enter the name of the management region to be created.

A default management region called DEFAULT is created by Enterprise Manager Configuration Assistant (EMCA).

EMCA can also be used to create a new management region while configuring a management server.

**Note:** Only management regions without an active management server and assigned nodes can be deleted.



## **Enterprise Manager Support for Oracle9i Database Features**

- **Server-managed parameter files ( SPFILEs )**
- **Automatic undo management**
- **Unicode**
- **Dynamic memory management and buffer cache sizing advice**
- **Default temporary tablespace**
- **Multiple block sizes**
- **The Mean Time to Recovery (MTTR) setting**
- **Backup and recovery enhancements**
- **Advanced Queuing**

ORACLE

6-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### **Enterprise Manager Support for Oracle9i Database Features**

The Console has been enhanced to support new features within the database management system.

## Creating the SPFILE



ORACLE

6-14

Copyright © Oracle Corporation, 2001. All rights reserved.

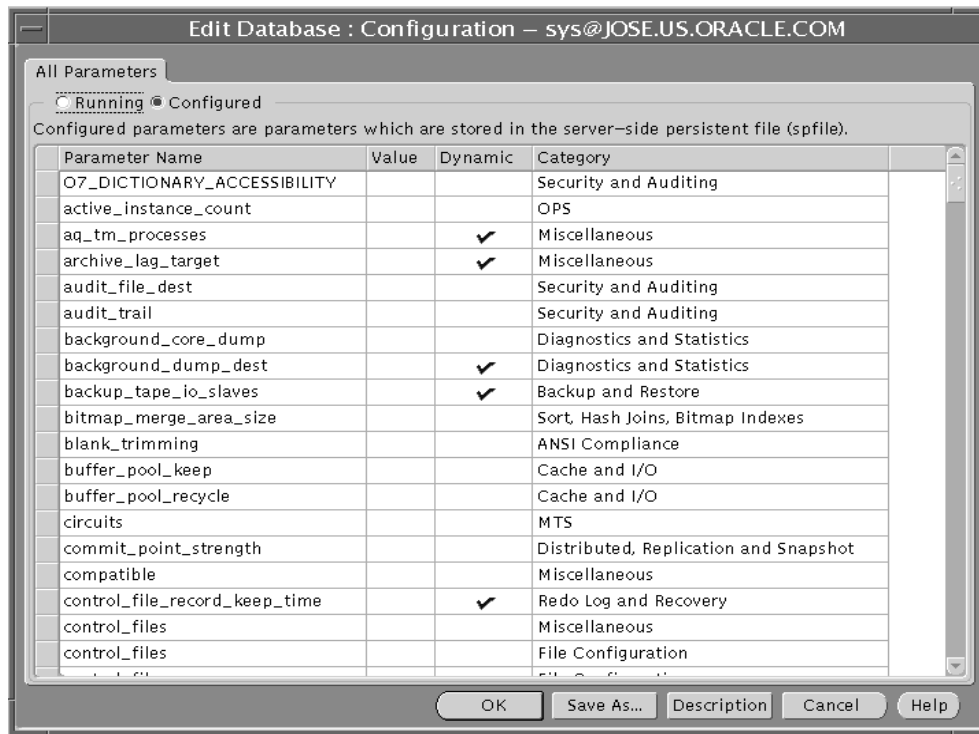
### Creating the SPFILE

The SPFILE is initially created from the initialization parameter file. The SYSDBA or SYSOPER privilege is required to create an SPFILE.

To create the SPFILE:

1. Right-click on the Configuration folder under Instance and click Import spfile. Or, select the Configuration folder and click Import spfile from the Object menu.
2. Enter the location of the parameter file (PFILE). Use the Browse button to search for the files on the database machine. Enter the name of the SPFILE you want to create. Both the PFILE and the SPFILE must be located on the same machine as the database.
3. Click the OK button to create the SPFILE.

# Changing Parameters



ORACLE

6-15

Copyright © Oracle Corporation, 2001. All rights reserved.

## Changing Parameters

The All Parameters page lists all initialization parameters.

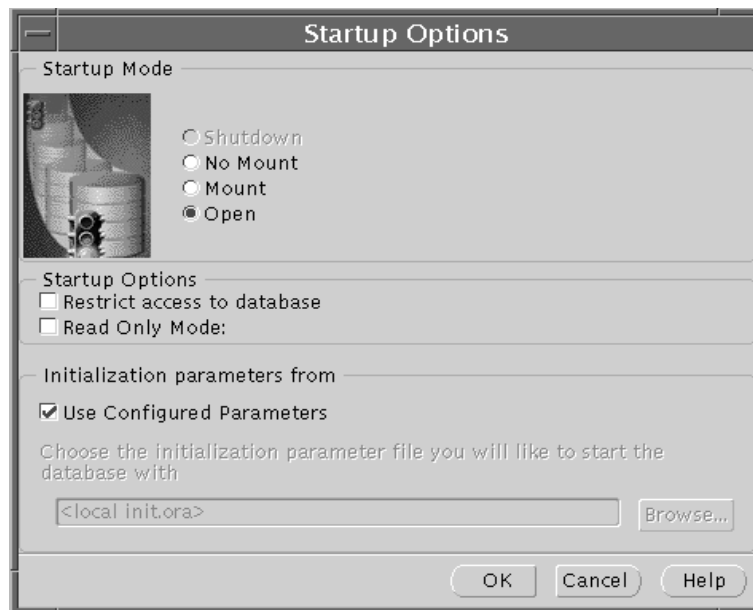
- For Oracle9i databases, you must be connected as SYSDBA to edit static initialization parameters. To edit dynamic parameters, the SYSDBA privilege is not required.
- For non-Oracle9i databases, you must be connected as SYSDBA to edit all parameters.

### Running Parameters and Configured Parameters

Running Parameters button shows the parameters that are currently running. If you change a dynamic parameter, the change is effected immediately. If the database is restarted, it starts up with the original parameter value, because the value has not been saved to the SPFILE. If you change a static parameter, however, the changes are saved to a PFILE or SPFILE and the database is restarted.

Configured parameters are stored in the server-side parameter file. Here, if you change a dynamic parameter, the changes are saved to an SPFILE and take place immediately.

# Startup Using the SPFILE



ORACLE

6-16

Copyright © Oracle Corporation, 2001. All rights reserved.

## Startup Using the SPFILE

To use the Startup dialog box to start the instance, perform the following steps:

1. Right-click on the database folder and select Startup, or elect Startup from the Object menu.
2. If you want to start up the database using SPFILE, select the Use Configured Parameters check box. The instance reads the initialization parameters from a server parameter file in a platform-specific default location.
3. If you want to use a `init.ora` file to start the database, clear the Use Configured Parameters check box and supply the `init.ora` file you want to use.

# Undo Tablespace Support

- **The Undo tablespace simplifies and automates the management of rollback segments**
- **The database administrator can manage rollback segments manually, or have Oracle automatically manage undo data in an undo tablespace.**
- **The rollback mode is set by the `UNDO_MANAGEMENT` initialization parameter.**
- **There is a new Undo option button in the Create Tablespace dialog box.**
- **The Instance Management Undo tabbed page displays all details about the undo tablespace.**

ORACLE

6-17

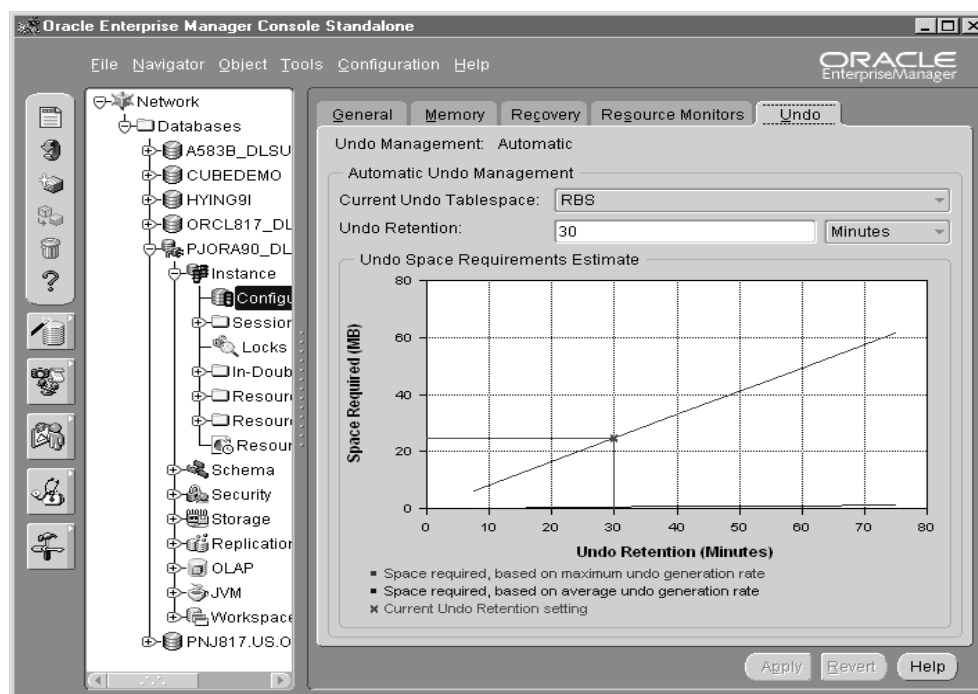
Copyright © Oracle Corporation, 2001. All rights reserved.

## Undo Tablespace

Enterprise Manager allows you to determine whether a database is in Automatic Undo Management mode. You specify whether the database uses rollback segments or an undo tablespace in the `init.ora` file. You should also specify which tablespace is the undo tablespace.

When you create a new tablespace, you can make it an undo tablespace by selecting the Undo option button in the Create Tablespace dialog box. For an undo tablespace, all storage options are disabled because they are handled automatically.

# Undo Tab

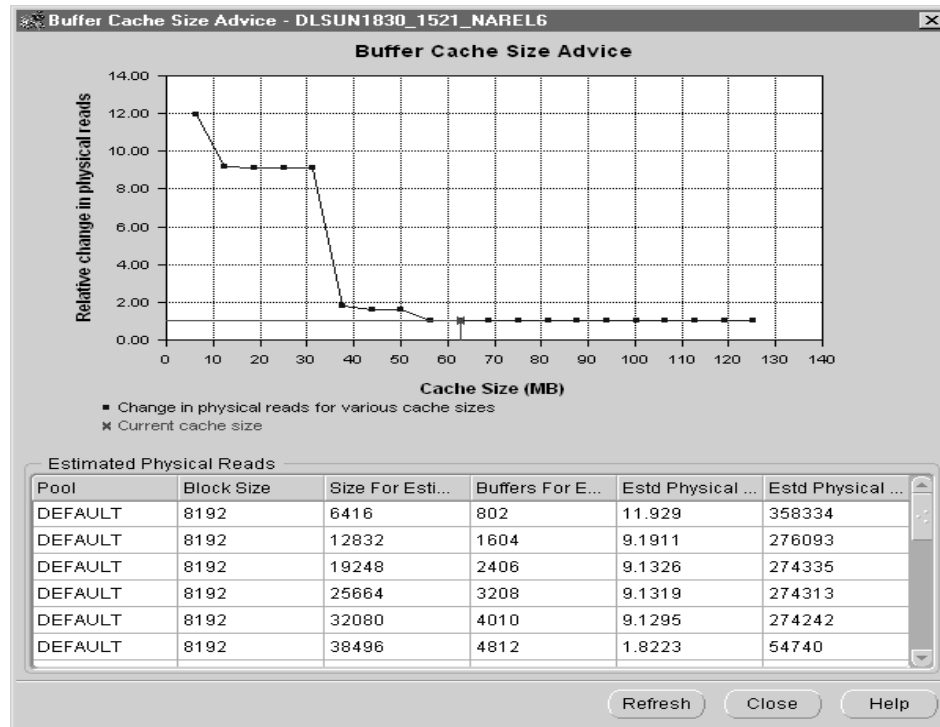


ORACLE

## Creating an Undo Tablespace

The Undo tabbed page in Instance Management displays all the details about the undo tablespace, including its name and the retention time. The retention time is the length of time to retain undo information. Committed undo information is normally lost when the undo space is overwritten by newer transactions. For consistent read consistency if long running queries might require old undo information, the Undo Retention feature provides a means of specifying the amount of undo information to retain.

# Buffer Cache Size Advice View



ORACLE

6-19

Copyright © Oracle Corporation, 2001. All rights reserved.

## Buffer Cache Size Advice View

You can view the Buffer Cache Size Advice information in Enterprise Manager as an alternative to querying the V\$DB\_CACHE\_ADVICE view. The Enterprise Manager display includes a chart of the information, which makes it much easier to view and interpret the data.

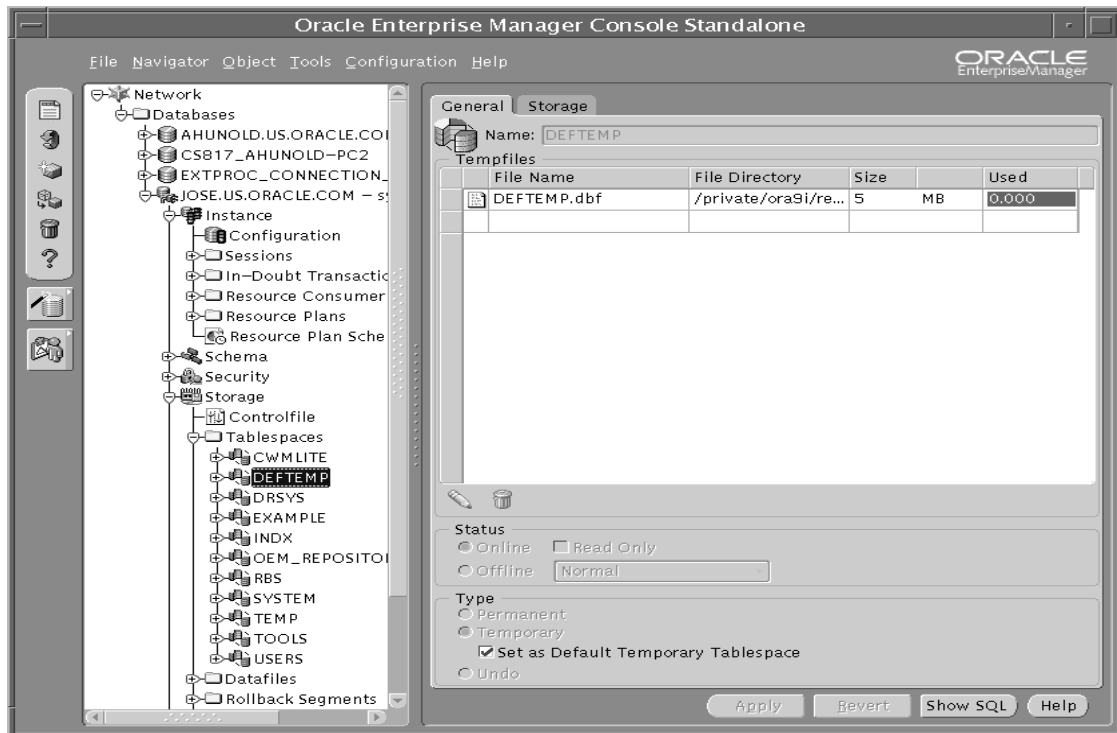
### How to Launch Buffer Cache Size Advice

1. Navigate to Instance > Configuration and click the Memory tab.
2. Click on Buffer Cache Size Advice to see the advice for choosing the size of the buffer cache.
3. If the DB\_CACHE\_ADVICE initialization parameter has not been changed, a dialog box appears, asking whether you want to change the \_CACHE\_ADVICE initialization parameter. If you select Yes, the parameter is changed.

### How to Interpret the Graphical Display

As you move the mouse over the graphical display of Physical Reads vs. Cache Size, text appears helping you interpret the data. For example, "If you set cache size to 5 MB the physical reads will increase by 2%."

# Creating Default Temporary Tablespace



6-20

Copyright © Oracle Corporation, 2001. All rights reserved.

## Creating Default Temporary Tablespace

Using Storage Management, you can either create a new tablespace and set it as the default temporary tablespace, or define an existing temporary tablespace as the default.

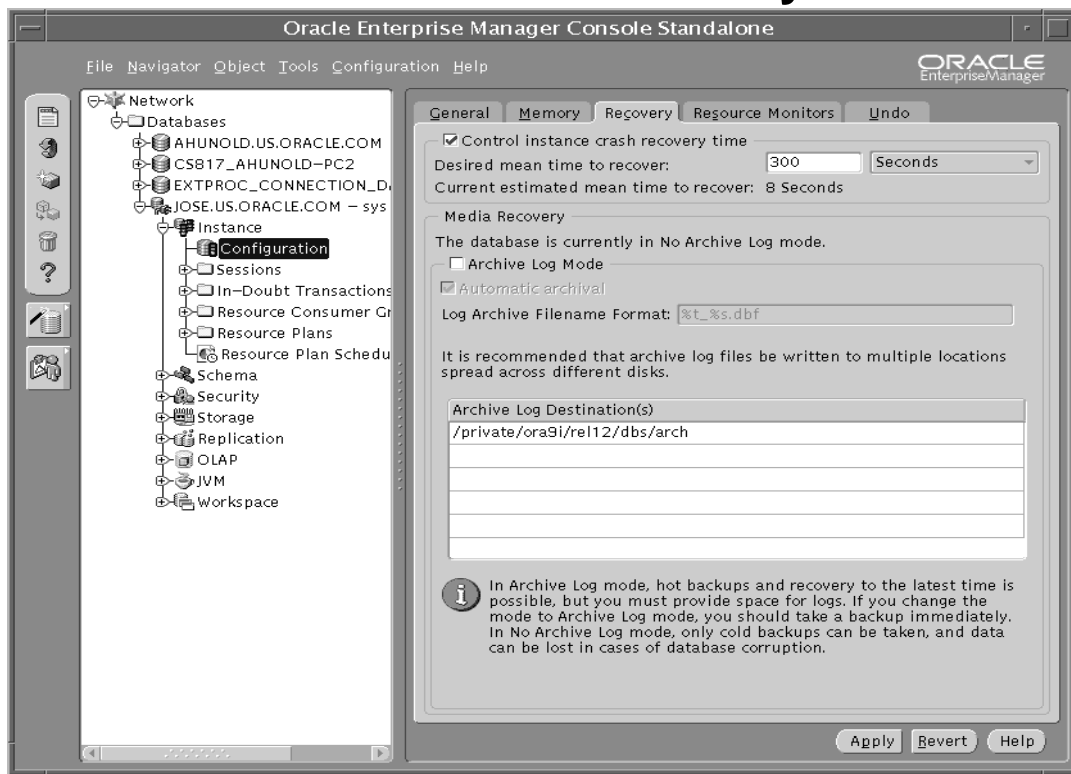
To create a default temporary tablespace:

1. Under the Create Tablespace folder, select the type as Temporary. This enables the Set as Default Temporary Tablespace check box.
2. Select this check box to make this tablespace the default temporary tablespace. You can also reassign the default temporary tablespace to another one.

You cannot take the default temporary tablespace offline or make it permanent.



# Mean Time to Recovery



6-21

Copyright © Oracle Corporation, 2001. All rights reserved.

## Mean Time to Recovery

- Controls the time required to recover from an instance crash
- Specifies the maximum time you want to spend recovering from an instance crash
- When you specify a value for this field, the `FAST_START_MTTR_TARGET` initialization parameter is changed automatically.

For example, you are running an e-commerce site and your instance crashes. You know there must be no more than five minutes of unavailability. You therefore set the MTTR to 5 minutes; the database then guarantees that it can recover within that period of time.

**Note:** Setting this field to too short a time can have a negative impact on database performance.

## Backup and Recovery Enhancements

- **Recovery Manager (RMAN) scripts can be submitted and scheduled by means of the Job system**
- **Supports the RMAN Image Copy option**
- **Requires Oracle9i intelligent agent**
- **Can be enabled in pre-Oracle9i agents by using the `rman.tcl` file**

ORACLE

6-22

Copyright © Oracle Corporation, 2001. All rights reserved.

### Backup and Recovery Enhancements

To enhance the flexibility of the backup and recovery facility, the Job system supports an RMAN-specific job task. You can simply input any RMAN script and submit it through the Job system.

To enable a RMAN script in pre-Oracle9i agents, add the `rman.tcl` file to the `$ORACLE_HOME/network/agent/jobs/oracle/rdbms/general` directory.

## Advanced Queuing

- **A topology map displays individual queues and their states on the selected database.**
- **You can view errors and drill down to diagnostic detail.**
- **Schedule details and messages can also be viewed.**

ORACLE

6-23

Copyright © Oracle Corporation, 2001. All rights reserved.

### Advanced Queuing

Advanced Queuing support is provided by introducing a new topology map that displays individual queues and their states on the selected database, including dblink status. You can see where there are errors and drill down to additional diagnostic detail.

# HTML Database Reports

**Report generation enhancements enable you to generate fully formatted reports. You can generate:**

- **A complete database configuration report**
- **A report of the properties of only the object selected in the navigator tree**
- **A report of object dependencies**

ORACLE

6-24

Copyright © Oracle Corporation, 2001. All rights reserved.

## HTML Database Reports

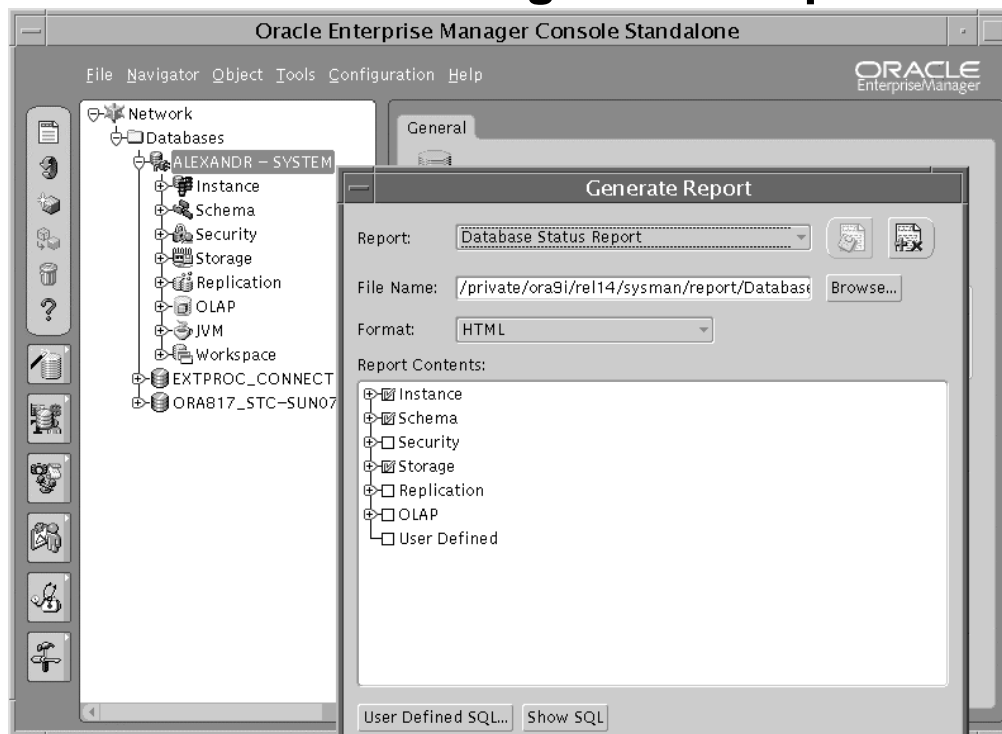
Report generation enhancements enable you to generate fully formatted reports with just the information you need.

You can generate complete database configuration properties reports, or reports that contain the properties of only the object selected in the navigator tree.

When you launch report generation, the dialog boxes that appear are appropriate to the currently selected object in the navigator tree.

These reports are available in Standalone mode.

# Database Configuration Report



6-25

Copyright © Oracle Corporation, 2001. All rights reserved.

## Database Configuration Reports

You can extract information from the database and save it in different formats (text, HTML, or Comma-Separated Values). For example, you can save the instance, storage, and schema information to an HTML file.

### How to Generate an HTML Report

1. Highlight Database in the navigator tree.
2. Click the Create Report icon in the toolbar. Or, right-click the database folder and click Create Report. Or, select Create Report from the Object menu.
3. Specify the desired contents of the report.
4. Click OK to generate the report, or click View to generate and immediately view the report.

### User Defined SQL Statement

The User Defined SQL enables you to optionally provide a SQL script to be added as an item under the report content's User Defined category.

**Note:** Each user-defined SQL statement must be no more than 2 KB.

## User-Defined Events

- **Allow you to integrate any event monitoring script (in any language) with the Event system**
- **Allow customization of monitoring**
- **Extend the benefits of the Event system to existing user-defined monitoring scripts**
- **Require Oracle9i agent**

ORACLE

6-26

Copyright © Oracle Corporation, 2001. All rights reserved.

### User-Defined Events

User-defined events enable you to integrate any monitoring script with the Event system. The agent runs any specified script which you define, to check for conditions specific to your environment. This provides great flexibility and customization to the type of event conditions monitored by the agent. It also extends the benefits of the Event system (cooperative monitoring among different administrators, notifications, history, and so on) to existing monitoring scripts.

The monitoring script itself should include the following:

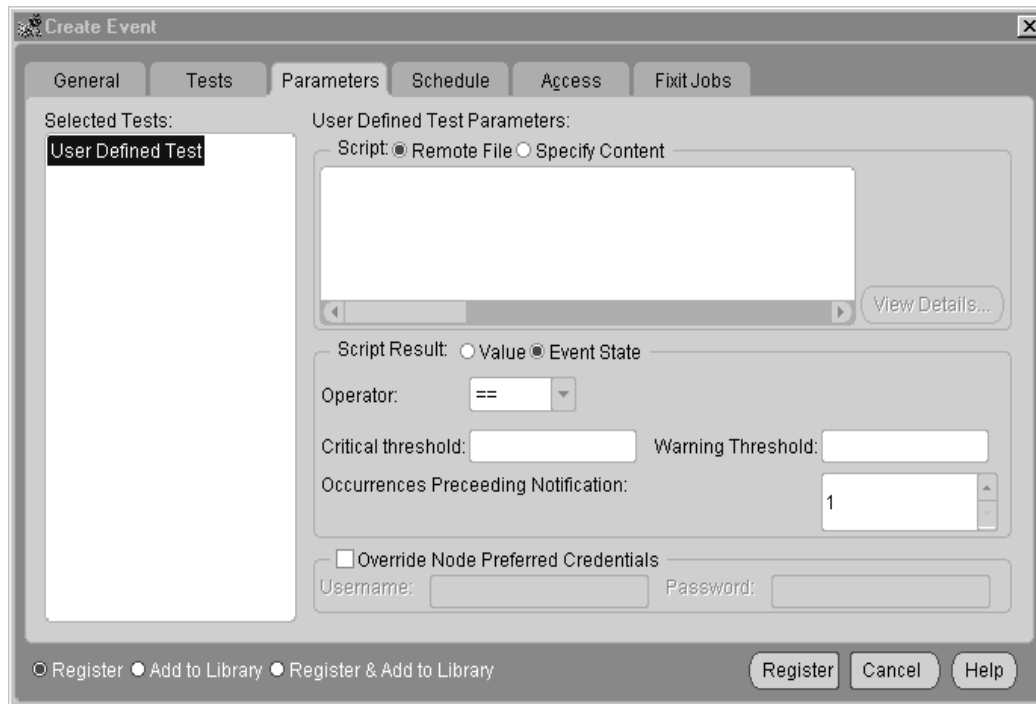
- Logic to check the specific event condition or value of the monitored metric (for example, the amount of free disk space)
- Logic to return the current value of the metric or status of the event and any associated message

If the current value of the metric is returned, then the user must also specify, in the Create Event window, critical and warning thresholds against which the current value is compared.

Alternatively, the script can evaluate and return the status of the event (clear, critical, warning, error), which is then propagated back and triggered accordingly.

The script itself can be written in any language, as long as the run-time requirements needed by the agent to run the script are available on the monitored node.

# User-Defined Event Tests



ORACLE

6-27

Copyright © Oracle Corporation, 2001. All rights reserved.

## User-Defined Event Tests

### Script

1. Specify the monitoring script. It can either reside on the monitored node, or you can enter it directly in the Create Event window.
2. If the script resides on the monitored node, enter the fully-qualified name of the script to be executed.

To enter the script directly, type in the script commands. Alternatively, you can use your favorite editor to enter the script and save it to a file, then load it into the Create Event window by clicking the Browse button.

### Script Result

The script should return either the value of the monitored metric or the status of the event. From the Script Result option buttons:

- Select Value if the script returns the value of the monitored metric.
- Select Event State if the script evaluates the event condition and returns an event status: Clear, Critical, Warning, or Error.

If Value is chosen for Script Result, the following additional parameters are required to determine how the event is evaluated:

### **Script Result (continued)**

- **Operator:** Specify the operator that Enterprise Manager should use to compare the monitored metric value against the thresholds. Comparison operators include == (equal to), < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to), and != (not equal to).
- **Critical Threshold:** The value against which the monitored metric is compared. If the comparison holds true, then the event triggers at a Critical level.
- **Warning Threshold:** The value against which the monitored metric is compared. If the comparison holds true, the event triggers at a Warning level.
- **Occurrences Preceding Notification:** The number of times the event condition must hold true before a notification is sent.

### **Override Node Preferred Credentials**

Node credentials are required because the agent executes the script as the user specifies in the node credentials. Default preferred credentials for the target node are used unless overridden in this field.

Refer to the online Help for more details.



# Event Handler

- **Extends the Event system by allowing customizable responses**
- **Enables the user to respond to a triggered event by:**
  - **Logging event information to a file, or**
  - **Executing any operating-system command**
- **Allows integration of third-party systems**
- **Takes advantage of OMS load-balancing and scalability features**

ORACLE

6-29

Copyright © Oracle Corporation, 2001. All rights reserved.

## Event Handler

All responses in the previous releases of the Event system were provided by the system itself. Now you can customize your own responses.

The event handler allows further processing when an event is triggered. Specifically, it allows you to log event information to a file, or execute any operating-system command.

These two generic mechanisms provide great flexibility in allowing third-party integration. Any third-party application can then parse the event log for further processing.

Additionally, if you have a trouble-ticketing system that has a command line interface, the event handler can be used to execute the command required to open a trouble ticket when the event triggers.

# Summary

**In this lesson, you should have learned how to:**

- **Use the Console in Standalone mode**
- **Use the Enterprise Manager functionality which supports Oracle9i database features**
- **Generate HTML reports**
- **Use user-defined events**

ORACLE



# **Performance Enhancements**

ORACLE<sup>®</sup>

Copyright © Oracle Corporation, 2001. All rights reserved.

# Objectives

**After completing this lesson, you should be able to:**

- **Use the new cost-based optimizer (CBO) model**
- **Use new first rows optimization**
- **Use the stored outlines editor**
- **Use the extensions to cursor sharing**
- **Access cached execution plans**
- **Use index skip scanning optimization**
- **Monitor index usage**

ORACLE

# Optimizer Cost Model Enhancements

**More meaningful cost estimates are available.**

- **The `plan_table` table contains three new columns:**
  - **`cpu_cost`: Estimated CPU cost of the operation**
  - **`io_cost`: Estimated I/O cost of the operation**
  - **`temp_space`: Estimated temporary space (in bytes) used by the operation**
- **Estimates include CPU and network usage**
- **Estimates account for the effect of caching**
- **Estimates account for index prefetching**

ORACLE

## Optimizer Cost Model Enhancements

The role of a query optimizer is to produce the best performing execution plan for a given query. This process includes selecting access paths for single tables, the join order if more than one table is involved in the query, and the join methods.

Currently, you have a choice between using the rule-based optimizer (RBO) and the cost-based optimizer (CBO). The CBO uses a cost model to choose between alternative access paths, join order, and join methods, whereas the RBO uses a set of simple rules.

The CBO compares the cost of several alternatives and selects the one with the lowest cost. In addition to the cost model, the CBO also uses a size model in order to derive statistics on intermediate tables; for example, the cardinality of the result of a join operation.

Both the cost and the size model use statistics on the objects manipulated by the query. Those statistics are produced by using the `DBMS_STATS` package, and are stored in the database dictionary.

The quality of the execution plan produced by the optimizer is dependent on the accuracy of both the cost and the size model. The current version of both models contains several limitations in terms of both accuracy and completeness. For example, the size model assumes column independence when computing the selectivity of multiple predicates on different columns, and the cost model accounts only for I/O activities.

## Optimizer Cost Model Enhancements (continued)

The cost model in Oracle9i has been extended as follows:

- Users or developers can convert the cost into more meaningful information. The `plan_table` table contains three new columns:
  - `cpu_cost`: The CPU cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. The value of this column is proportional to the number of machine cycles required for the operation
  - `io_cost`: The I/O cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. The value of this column is proportional to the number of data blocks read by the operation.
  - `temp_space`: The temporary space, in bytes, used by the operation, as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, or for operations that use no temporary space, this column is null.
- It includes CPU and network usage. CPU usage is estimated for SQL functions and operators; network usage is estimated when data is shipped between query servers running on different nodes of a cluster.
- It accounts for the effect of caching on the performance of nested-loops joins.
- It accounts for index prefetching: Fetching multiple leaf blocks in a single I/O operation.

## New Statistics-Gathering Estimates

- **New `estimate_percent` value available:**  
`AUTO_SAMPLE_SIZE`
- **New histogram size options:**
  - **REPEAT:** Same number of buckets
  - **AUTO:** Based on data distribution and application workload
  - **SKEWONLY:** Based on data distribution

```
EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS(... ,  
estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,  
method_opt       => 'for all columns size AUTO');
```

ORACLE

### New Statistics-Gathering Estimates

Because the cost-based approach relies on statistics, users should generate statistics for all tables and clusters, and all indexes accessed by SQL statements, before using the cost-based approach. If the size and data distribution of the tables change frequently, then users should generate these statistics regularly to ensure that the statistics accurately represent the data in the tables.

Exact statistics computation requires enough space to perform scans and sorts of involved objects. If there is not enough space in memory, then temporary space may be required. Thus, it is also possible to compute only estimations in order to reduce the resources needed to gather statistics.

The difficulty in computing estimated statistics is finding the best sample size. Some statistics are always computed exactly, such as the number of data blocks currently containing data in a table or the depth of an index from its root block to its leaf blocks. However, this is not true for all statistics.

With Oracle9i, you should set the `ESTIMATE_PERCENT` parameter of the `DBMS_STATS` gathering procedures to the new `DBMS_STATS.AUTO_SAMPLE_SIZE` value. This value is introduced to maximize performance gains while achieving necessary statistical accuracy, thereby avoiding the extremes of collecting inaccurate statistics and wasting valuable time.

## New Statistics-Gathering Estimates (continued)

Oracle9i introduces some new possible values for the METHOD\_OPT parameters of the DBMS\_STATS gathering procedures:

- If the size option is set to REPEAT and the column currently has a histogram with b buckets, the Oracle server attempts to create a new histogram with b buckets. If the column has no histogram, no new statistics are gathered. This option is used to maintain the same “class” of statistics (histogram/no-histogram) when looking at new data.
- If the size is set to AUTO, the Oracle server creates a histogram based on the data distribution *and* the way the column is being used by the application. This means that the Oracle server looks not only at nonuniformity in value repetition counts (skew), but also at nonuniformity in range (sparsity). If the application has yet to be run for an amount of time sufficient to capture the workload involving this column, it is better to use the SKEWONLY option temporarily.
- If the size is set to SKEWONLY, the Oracle server creates a histogram based solely on the data distribution (regardless of how the application uses the column). This option is useful when gathering statistics for the first time (before the workload has had time to run). Using SKEWONLY can add quite a bit of overhead to statistics collection, so you should use AUTO after the application has run for a while.

The example in the slide shows you how to collect all table, column, and index statistics for the OE schema, where the Oracle server decides what the sampling percentage should be and when histograms are necessary (assuming that the workload has run for a while).

**Note:** The Oracle server captures workload information for a cursor when it is hard parsed. Information is stored in the SGA and regularly flushed to disk. No access to these memory and disk structures is provided in Oracle9i.



## Gathering System Statistics

- **System statistics enable the CBO to consider CPU and I/O characteristics when deciding on a plan.**
- **System statistics must be gathered on a regular basis; this does not invalidate cached plans.**
- **Gathering system statistics means analyzing system activity for a specified period of time.**
- **New procedures:**
  - `DBMS_STATS.GATHER_SYSTEM_STATS`
  - `DBMS_STATS.SET_SYSTEM_STATS`
  - `DBMS_STATS.GET_SYSTEM_STATS`

ORACLE

7-7

Copyright © Oracle Corporation, 2001. All rights reserved.

### Gathering System Statistics

System statistics allow the optimizer to consider a system's I/O and CPU performance and utilization. For each candidate plan, the optimizer computes estimates for I/O and CPU costs. It is important to know the system characteristics to pick the most efficient plan with the optimal proportion between I/O and CPU cost.

System CPU and I/O characteristics depend on many factors and do not stay constant all the time. Using system statistics management routines, database administrators can capture statistics in the interval of time when the system has the most common workload. For example, database applications can process online transaction processing (OLTP) transactions during the day and run online analytical processing (OLAP) reports at night. Administrators can gather statistics for both states and activate the appropriate OLTP or OLAP statistics when needed. This allows the optimizer to generate relevant costs with respect to available system resource plans.

When the Oracle server generates system statistics, it analyzes system activity in a specified period of time. Unlike when table, index, or column statistics get updated, the Oracle server does not invalidate already-parsed SQL statements when system statistics get updated. All new SQL statements are parsed using new statistics. It is highly recommended that you gather system statistics.

The `DBMS_STATS.GATHER_SYSTEM_STATS` routine collects system statistics in a user-defined timeframe. You can also set system statistics values explicitly using `DBMS_STATS.SET_SYSTEM_STATS`. Use `DBMS_STATS.GET_SYSTEM_STATS` to verify system statistics.

## Example of Gathering System Statistics

- **First day:**

```
EXECUTE DBMS_STATS.GATHER_SYSTEM_STATS(  
interval => 120,  
stattab => 'mystats', statid => 'OLTP');
```

- **First night:**

```
EXECUTE DBMS_STATS.GATHER_SYSTEM_STATS(  
interval => 120,  
stattab => 'mystats', statid => 'OLAP');
```

- **Subsequent days:**

```
EXECUTE DBMS_STATS.IMPORT_SYSTEM_STATS(  
stattab => 'mystats', statid => 'OLTP');
```

- **Subsequent nights:**

```
EXECUTE DBMS_STATS.IMPORT_SYSTEM_STATS(  
stattab => 'mystats', statid => 'OLAP');
```

ORACLE

### Example of Gathering System Statistics

The above example shows database applications processing OLTP transactions during the day and running reports at night.

First, system statistics must be collected during the day. Here, gathering ends after 120 minutes and is stored in the `mystats` table.

Then, system statistics are collected during the night. Gathering ends after 120 minutes and the statistics are also stored in the `mystats` table.

Generally, you use the above syntax to gather the system statistics. In that case, you must be sure, before invoking the `GATHER_SYSTEM_STATS` procedure with the `INTERVAL` parameter specified, to activate job processes using a command such as the following:

```
SQL> alter system set job_queue_processes = 1;
```

Alternatively, you can invoke the same procedure with different arguments to enable manual gathering instead of using jobs. For syntax information, refer to the *Oracle9i Supplied PL/SQL Packages Reference Release 9.0.0*.

If appropriate, you can switch between the statistics gathered. Note that it is possible to automate this process by submitting a job to update the dictionary with appropriate statistics: During the day, a job can import the OLTP statistics for the daytime run; during the night, another job can import the OLAP statistics for the nighttime run.

## New First Rows Optimization

```
OPTIMIZER_MODE = {FIRST_ROWS_1  
                  |FIRST_ROWS_10  
                  |FIRST_ROWS_100  
                  |FIRST_ROWS_1000}
```

```
SQL> ALTER SESSION SET  
      2 OPTIMIZER_GOAL = FIRST_ROWS_10
```

```
SQL> SELECT /*+ FIRST_ROWS(25) */ ...  
      2 FROM ...
```

ORACLE

### New First Rows Optimization

Oracle9i introduces a new way of doing first rows optimization. The old mode was partially based on heuristics, for example: Always use an index if possible. These heuristics could sometimes lead to very bad plans.

The new mode is completely cost based, and allows optimizing for a particular number of first rows; for example, the first 10 rows.

It is invoked as an argument to the `OPTIMIZER_MODE` initialization parameter or the `OPTIMIZER_GOAL` session parameter, which allows the following values:

- `first_rows_1`
- `first_rows_10`
- `first_rows_100`
- `first_rows_1000`

In addition, the `FIRST_ROWS` hint now takes a numeric argument that is not limited to the values for the parameter. For instance, you can specify `/*+FIRST_ROWS(25)*/`.

Without a numeric argument, both the parameter and the hint imply the old first-rows behavior, which is retained for backwards compatibility and plan stability.

## Outline Editing Overview

- **Oracle8i introduced stored outlines; Oracle9i enables you to tune execution plans without having to change the application.**
- **The idea is to clone the outline in a staging area where the outline can be safely edited without impacting the user community.**
- **Once satisfied by the result, you can publicize the result to the user community.**

ORACLE

7-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### Outline Editing Overview

Stored outlines were introduced in Oracle8i as a way to preserve execution plan stability across releases of the database and across different operating environments. In Oracle9i, outline editing extends the usefulness of stored outlines by introducing a user-editing interface. Users can tune their execution plans by editing the stored outlines that influence the optimizer.

This feature benefits both application developers and customer-support personnel. Although the optimizer usually chooses optimal plans for queries, there are times when the user knows something about the execution environment that is inconsistent with the heuristics that the optimizer follows. Sometimes an execution plan is acceptable in one environment but not in another. By editing the outline directly, the user can tune the query without having to change the application.

An application developer can generate outlines in a staging area and notice that some plan did not take advantage of an index which could improve performance. It might be easier to simply edit the outline to use the index, rather than searching through the application code and tuning the SQL until it eventually yields the desired result.

If there is a problem query at a customer site, customer-support personnel can often solve the problem quickly by creating an outline and then editing it with some simple edit, such as changing the join order. In this way, the customer's problem is solved immediately, without anyone needing to go through the process of debugging and updating the application itself.

## Outline Editing Overview (continued)

For the customer whose environment has unique characteristics that might cause an outline to yield a less-than-optimal execution plan, the ability to make minor adjustments to the outline enhances the ability to support specific customer needs. In this sense, stored outlines are made more adaptive, because users can make finely tuned adjustments to the saved plan.

Stored outline metadata is maintained in the `outln` schema and maintained by the Oracle server. Users should not update these tables directly, just as they are advised not to update any other system tables.

You can safely edit outlines without compromising the integrity of the outline; to accomplish this, the outline is cloned into your schema at the onset of the outline editing session. All subsequent editing operations are performed on that clone until you are satisfied with the edits and choose to publicize them. In this way, any editing done by you does not impact the rest of the user community, who continue to use the public version of the outline until the outline edits are explicitly saved.

## Attributes Which Can Be Edited

- **Join order**
- **Join methods**
- **Access methods**
- **Distribution methods**
- **Query rewrite**
- **View and subquery merging and summary rewrite**

ORACLE

7-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### Attributes Which Can Be Edited

- **Join order:** Join order defines the sequence in which tables are joined during query execution. This includes tables produced by evaluating subqueries and views, as well as tables appearing in the FROM clauses of subqueries and views.
- **Join methods:** Join methods define the methods used to join tables during query execution. Examples are nested-loop joins and sort-merge joins.
- **Access methods:** Access methods define the methods used to retrieve table data from the database. Examples are indexed access and full table scan.
- **Distributed execution plans:** Distributed queries have execution plans that are generated for each site at which some portion of the query is executed. The execution plan for the local site at which the query is submitted can be controlled by plan stability, and equivalent plans must be produced at that site. In addition, driving site selection can be controlled centrally even though it may normally change when certain schema changes are made.
- **Distribution methods:** For parallel query execution, distribution methods define how the inputs to execution nodes are partitioned.
- **View and subquery merging and summary rewrite:** This includes all transformations in which objects or operations that occur in one subquery of the original SQL statement are caused to migrate to a different subquery for execution. Summary rewriting can also cause one set of objects or operations to be replaced by another.

# Outline Cloning

- **Public outlines:**
  - Default setting when creating outlines
  - Stored in the `outln` schema
  - Used when `USE_STORED_OUTLINES = TRUE`
- **Private outlines:**
  - Stored in the user's schema
  - Can be edited
  - Used when `USE_PRIVATE_OUTLINES = TRUE`
  - Changes can be saved to public

ORACLE

7-13

Copyright © Oracle Corporation, 2001. All rights reserved.

## Outline Cloning

### Public Outlines

In Oracle8i, all outlines are public objects indirectly available to all users on the system for whom the `USE_STORED_OUTLINES` configuration parameter setting applies. Outline data resides in the `outln` schema, which can be thought of as an extension to the `sys` schema in the sense that it is maintained and consumed by the system only. Users were discouraged from manipulating this data directly, to avoid the security and integrity issues associated with outline data. Outlines continue to be public by default, and only public outlines are generally available to the user community.

### Private Outlines

Oracle9i introduces the notion of a private outline to aid in outline editing. A private outline is an outline seen only in the current session, and whose data resides in the current parsing schema. Because the outline data for a private outline is stored directly in the user's schema, users can directly manipulate the outline data through DML however they choose. Any changes made to such an outline are not seen by any other session on the system, and a private outline can be applied to the compilation of a statement only in the current session through a new session parameter. Only when a user explicitly chooses to save edits back to the public area can the rest of the users see them.

An outline clone is a private outline that has been created by copying data from an existing outline.

## Outline Administration and Security

- **Privileges required for using `CREATE OUTLINE FROM`:**
  - **`SELECT_CATALOG_ROLE`**
  - **`CREATE ANY OUTLINE`**
- **`dbms_outln_edit.create_edit_tables`**
  - **Creates required tables in user's schema for cloning and editing outlines (equivalent to executing `utledit01.sql`)**
  - **Requires `EXECUTE` privilege on `dbms_outln_edit`**
- **`v$sql`**
  - **`outline_sid` column added: The session ID from which the outline was retrieved**

ORACLE

7-14

Copyright © Oracle Corporation, 2001. All rights reserved.

### Outline Administration and Security

#### **`SELECT_CATALOG_ROLE`**

This role is required for the `CREATE OUTLINE FROM` command, unless the issuer of the command is also the owner of the outline. Any `CREATE OUTLINE` statement requires the `CREATE ANY OUTLINE` privilege. Specification of the `FROM` clause requires the additional `SELECT_CATALOG_ROLE` role, because such a command exposes SQL text to different users who might otherwise not be privileged to read the text.

#### **`dbms_outln_edit.create_edit_tables`**

This is a supporting command procedure that creates the metadata tables in the invoker's schema. It can be called by anyone with `EXECUTE` privilege on `DBMS_OUTLN_EDIT`.

#### **`V$SQL` extension**

A column is added to the `V$SQL` fixed view to help users distinguish whether a shared cursor was compiled while using a private outline or a public outline. `outline_sid` is the name of this new column, which identifies the session ID from which the outline was retrieved. The default is 0, which implies a lookup in the `outln` schema.



## Configuration Parameters

**USE\_PRIVATE\_OUTLINES** is a session parameter that control the use of private outlines instead of public outlines:

```
ALTER SESSION SET USE_PRIVATE_OUTLINES =  
{TRUE|FALSE|category_name}
```

- **TRUE** enables the use of private outlines in the **DEFAULT** category.
- **FALSE** disables the use of private outlines.
- **category\_name** enables the use of private outlines in the named category.

ORACLE

7-15

Copyright © Oracle Corporation, 2001. All rights reserved.

### Configuration Parameters

The **USE\_PRIVATE\_OUTLINES** session parameter is added to control the use of private outlines instead of public outlines. When an outlined SQL command is issued, this parameter causes outlines to be retrieved from the session private area rather than the public area normally consulted as per the setting of **USE\_STORED\_OUTLINES**. If no outline exists in the session private area, no outline is used to compile the command.

You can specify a value for this session parameter by using the following syntax:

```
ALTER SESSION SET USE_PRIVATE_OUTLINES =  
{TRUE|FALSE|category_name}
```

Where:

- **TRUE** enables the use of private outlines and defaults to the **DEFAULT** category.
- **FALSE** disables the use of private outlines.
- **category\_name** enables the use of private outlines in the named category.

When a user begins an outline editing session, the parameter should be set to the category to which the outline being edited belongs. This enables the feedback mechanism, allowing the private outline to be applied to the compilation process.

Upon completion of outline editing, this parameter should be set to **False** to restore the session to normal outline lookup, as dictated through the **USE\_STORED\_OUTLINES** parameter.

## CREATE OUTLINE Syntax Changes

- **PUBLIC:** Create for public use (this is the default)
- **PRIVATE:** Create for private use by the current session only
- **FROM:** Create an outline by copying an existing one
- **source\_outline\_name:** The name of the outline being cloned

```
CREATE [OR REPLACE] [PUBLIC|PRIVATE] OUTLINE [outline_name]  
[FROM [PUBLIC|PRIVATE] source_outline_name]  
[FOR CATEGORY category_name] [ON statement]
```

ORACLE

7-16

Copyright © Oracle Corporation, 2001. All rights reserved.

### CREATE OUTLINE Syntax Changes

The new syntax elements are highlighted in blue in the above slide:

- **PUBLIC:** The outline is created for systemwide use. This is the default.
- **PRIVATE:** The outline is created for private use by the current session only, and its data is stored in the current parsing schema. When specified, the prerequisite outline tables and indices must exist in the local schema.
- **FROM:** This construct provides a way to create an outline by copying an existing one.
- **source\_outline\_name:** The outline being cloned. By default, it is found in the public area, but if preceded by the **PRIVATE** keyword it is found in the local schema.

The addition of the **PRIVATE** and **FROM** keywords enable outline cloning. If you want to edit an outline, you do so on a private copy which is created by specifying the **PRIVATE** keyword. In the **FROM** clause, the source outline to be edited is named, and that source is found in the public area unless preceded by the **PRIVATE** keyword, in which case you are copying a private version of the named outline.

When specifying the **FROM** clause, existing semantics apply to outline name and category; if unspecified, an outline name is generated under the **DEFAULT** category. When a **PRIVATE** outline is being created, an error is returned if the prerequisite outline tables to hold the outline data do not exist in the local schema.

**Note:** It is also possible to use the **PUBLIC** and **PRIVATE** keywords with the **ALTER OUTLINE** command.

## Examples of Outline Cloning

```
SQL> CREATE PRIVATE OUTLINE private_outline  
2 FROM public_outline;
```

```
SQL> CREATE PRIVATE OUTLINE private_outline2  
2 FROM PRIVATE private_outline1  
3 FOR CATEGORY cat2;
```

```
SQL> CREATE OR REPLACE OUTLINE public_outline  
2 FROM PRIVATE private_outline
```

ORACLE

7-17

Copyright © Oracle Corporation, 2001. All rights reserved.

### Examples of Outline Cloning

```
SQL> CREATE PRIVATE OUTLINE private_outline  
2 FROM public_outline;
```

This example shows how to clone a public outline to the session private area for editing purposes. In this case, the `public_outline` outline is cloned into the private area and given the name `private_outline`.

```
SQL> CREATE PRIVATE OUTLINE private_outline2  
2 FROM PRIVATE private_outline1 FOR CATEGORY cat2;
```

This example shows how to clone a private copy of an outline to another private copy. In this case, the `private_outline2` outline is created in category `cat2` from the `private_outline1` outline, which is local to the current session. This can be done if the user wants to save several versions of the outline. The category must be specified here to avoid namespace collision.

```
SQL> CREATE OR REPLACE OUTLINE public_outline  
2 FROM PRIVATE private_outline;
```

This example shows how to copy a private outline back to the public area for general use. This is a way to save local edits back to the master copy of the outline. This operation is also called “publicizing.”

## Examples of Outline Cloning

```
SQL> CREATE PRIVATE OUTLINE private_outline  
2 ON select count(*) from employee;
```

```
SQL> CREATE OR REPLACE OUTLINE public_outline2  
2 FROM public_outline1 FOR CATEGORY cat2;
```

```
SQL> CREATE OR REPLACE  
2 PRIVATE OUTLINE private_outline1  
3 FROM PRIVATE private_outline1;
```

ORACLE

7-18

Copyright © Oracle Corporation, 2001. All rights reserved.

### Examples of Outline Cloning (continued)

```
SQL> CREATE PRIVATE OUTLINE private_outline ON  
2 select count(*) from employees;
```

This example demonstrates how to create a private outline independently without copying from a pre-existing outline. The `private_outline` outline is created in the session private area. This is useful for users who want to experiment with an outline before creating it for public consumption.

```
SQL> CREATE OR REPLACE OUTLINE public_outline2  
2 FROM public_outline1 FOR CATEGORY cat2;
```

This example shows how to copy one public outline to another. Here the `public_outline1` public outline is copied to `public_outline2` and placed in category `cat2`. The category must be specified here to avoid namespace collision.

```
SQL> CREATE OR REPLACE PRIVATE OUTLINE private_outline1  
2 FROM PRIVATE private_outline1;
```

This example shows how to replace an outline with itself. This may seem like a useless operation, but it actually serves the dual purpose of invalidating dependent shared cursors and bringing the shared pool view of the object in line with what is on disk due to the user's edits. The feedback mechanism relies on this aspect of the command.

## Example of Manual Outline Editing

```
SQL> EXECUTE DBMS_OUTLN_EDIT.CREATE_EDIT_TABLES;
```

```
SQL> CREATE PRIVATE OUTLINE OL1 FROM OL1;
```

```
SQL> UPDATE OL$HINTS  
2 SET HINT_TEXT = 'INDEX(T1 I1)'  
3 WHERE HINT# = 5;
```

ORACLE

7-19

Copyright © Oracle Corporation, 2001. All rights reserved.

### Example of Manual Outline Editing

1. Connect as scott (for example), who must have both the CREATE ANY OUTLINE privilege and the SELECT\_CATALOG\_ROLE role in order to edit an outline.
2. Create the outline editing tables in that schema:  

```
SQL> EXECUTE DBMS_OUTLN_EDIT.CREATE_EDIT_TABLES;
```
3. Clone the outline for editing:  

```
SQL> CREATE PRIVATE OUTLINE OL1 FROM OL1;
```
4. Edit the outline by performing DML operations against the local ol\$hints table; for example:  

```
SQL> UPDATE_OL$HINTS  
2 SET HINT_TEXT='INDEX(T1 I1)'  
3 WHERE HINT#=5;
```

## Example of Manual Outline Editing

```
SQL> CREATE PRIVATE OUTLINE OL1  
2 FROM PRIVATE OUTLINE OL1;  
  
SQL> ALTER SESSION SET  
2 USE_PRIVATE_OUTLINES = true;
```

```
SQL> CREATE OR REPLACE OUTLINE OL1  
2 FROM PRIVATE OUTLINE OL1;
```

ORACLE

7-20

Copyright © Oracle Corporation, 2001. All rights reserved.

### Example of Manual Outline Editing (continued)

5. Prepare to test the edits:

```
SQL> CREATE PRIVATE OUTLINE OL1 FROM PRIVATE OUTLINE OL1;
```

This step is needed to re-read the edited version of the outline on disk and to invalidate all corresponding cursors in order to recompile them.

```
SQL> ALTER SESSION SET USE_PRIVATE_OUTLINES=true;
```

6. Execute the outlined query; save, or publicize the edits

```
SQL> CREATE OR REPLACE OUTLINE OL1  
2 FROM PRIVATE OUTLINE OL1;
```

### Outline Editor

Oracle9i gives users the ability to manage their outlines directly. With new outline management functionality, the Outlines window, you can browse, sort, delete, export, and import outline. You can also tell whether an outline, once defined or imported, has been used, or continues to be used after a database or application upgrade, and you can change the category of a group of outlines.

The Outlines window is accessible through SQL Analyze (the primary interface for dealing with outlines) and the Console. Because outlines are primarily used for tuning SQL statements, users create (other than through SQL), edit, analyze, and compare them using the Outline Editor functionality of SQL Analyze.

# Cursor Sharing Enhancements

- **Literal replacement:** Literal values are replaced by bind variables before optimizing
- **Safe literal replacement:** Regardless of the literal value provided in the statement, the optimizer develops the same execution plan
- **Unsafe literal replacement:** The optimizer can develop different execution plans for different literal values

ORACLE

7-21

Copyright © Oracle Corporation, 2001. All rights reserved.

## Literal Replacement

The term *literal replacement* refers to an operation performed by the optimizer to reduce the overhead of parsing SQL statements containing literal values. Rather than developing an execution plan each time a similar statement is executed with a different literal value, the optimizer substitutes a bind variable for the literal when the statement is first parsed. The execution plan developed using the bind variable is used for all subsequent executions of the statement.

## Safe Literal Replacement

Consider a query such as this one, where EMPLOYEE\_ID is the primary key:

```
SQL> SELECT * FROM employees WHERE employee_id = 153;
```

The substitution of any value would produce exactly the same execution plan. It is therefore safe for the optimizer to use literal replacement and substitute a bind variable for the value 153 before generating the execution plan for this statement.

## Unsafe Literal Replacement

Assume the same `employees` table has a wide range of values in its `DEPARTMENT_ID` column. For example, department 50 contains over one-third of all employees, and department 70 contains just one or two. Given the following two queries:

```
SQL> SELECT * FROM employees WHERE department_id = 50;
```

```
SQL> SELECT * FROM employees WHERE department_id = 70;
```

Replacing either value with a bind variable is not a safe literal replacement if you have histogram statistics (and there is skew in the data) on the `department_id` column. In this case, depending on which statement was executed first, the execution plan may contain a full table (or fast full index) scan, or it may use a simple index range scan.



## The CURSOR\_SHARING Parameter

- **CURSOR\_SHARING parameter values:**
  - **FORCE**
  - **EXACT** (the default)
  - **SIMILAR** (new in Oracle9i)
- **CURSOR\_SHARING can be changed using:**
  - **ALTER SYSTEM**
  - **ALTER SESSION**
  - **INIT.ORA**

ORACLE

7-23

Copyright © Oracle Corporation, 2001. All rights reserved.

### The CURSOR\_SHARING Parameter

The value of the CURSOR\_SHARING initialization parameter determines how the optimizer processes statements with bind variables:

- **EXACT:** Literal replacement is disabled completely
- **FORCE:** Sharing is used for all literals
- **SIMILAR:** Sharing is used for safe literals only

In previous releases, you could choose only the EXACT or the FORCE option. The SIMILAR option is new in Oracle9i. It causes the optimizer to examine the statement to ensure that replacement occurs only for safe literals. In doing this, it can use information about the nature of any available index (unique or non-unique), and any statistics collected on the index or underlying table, including histograms.

The value of CURSOR\_SHARING in the initialization file can be overridden with an ALTER SYSTEM SET CURSOR\_SHARING or an ALTER SESSION SET CURSOR\_SHARING command.

## Cached Execution Plans

- **Cached execution plans preserve the actual execution plan of a cached SQL statement in memory.**
- **Benefits include:**
  - Use of actual execution plans
  - Better diagnosis of query performance
  - An easy-to-use interface for monitoring bottlenecks
- **Cached execution plans are removed when the SQL statement is aged out.**

ORACLE

### Cached Execution Plans

Prior to Oracle9i, the V\$SQLTEXT query text could be used to gather useful information about cached cursors and execution statistics, such as the number of executions and the number of buffer gets from V\$SQL. In order to determine the execution plan corresponding to a cursor, the SQL text was fetched from V\$SQLTEXT and an EXPLAIN PLAN command was run on the SQL text. With this approach, the execution plan obtained by the EXPLAIN PLAN command can differ from the execution plan used to execute the cursor, because the cursor may have been compiled with different session parameter values. With the Cached Execution Plan feature, the execution plan information for the cached cursors is made available through a new dynamic performance table.

The performance of a database application depends on the performance of the database, so tuning SQL statements is important in tuning the application performance. Cached Execution Plan information helps greatly during the SQL statement tuning process. For example, you can determine the effect of creating an index on a table, search cursors containing a certain access path, or identify the indexes that are or are not selected by the optimizer.

You can also track changes in the performance of SQL statements and correlate those changes with changes in execution plans. Such changes can happen after migrating the application or the database to a new release, switching from the rule-based optimizer to the cost-based optimizer, running the ANALYZE command on the database objects, dropping or creating indexes, or changing parameter values.

Once the SQL statement is aged out of the library cache, the corresponding cached execution plan is removed.

## New View to Support Cached Execution Plans

- A new dynamic performance view, `V$SQL_PLAN`, stores actual execution plan information for a cached cursor.
- The view contains all of the `plan_table` columns (except the `level` column), in addition to four new columns.
- Columns that are also present in the `plan_table` table have the same value as the columns in `V$SQL_PLAN`.

ORACLE

7-25

Copyright © Oracle Corporation, 2001. All rights reserved.

### New View to Support Cached Execution Plan History

The `V$SQL_PLAN` view stores the actual execution plan information for a cached cursor. The view contains all of the `plan_table` columns except the `level` column, as well as the following four new columns:

- `address`: Cursor parent handle address
- `hash_value`: Parent statement hash value in library cache
- `child_number`: Number using this execution plan
- `depth`: Level of the operation in the tree

The columns which are also present in `plan_table` have the same values in `V$SQL_PLAN`.

## **New `plan_hash_value` Column in `V$SQL`**

**This new column contains the hashed value obtained from the corresponding execution plan. You can use it for comparisons:**

- **Take a snapshot of cursor plan information (`plan_hash_value`) for cached SQL statements**
- **Execute your application again**
- **Find queries whose plans have changed (that is, which have a different value for `plan_hash_value`)**

ORACLE

7-26

Copyright © Oracle Corporation, 2001. All rights reserved.

### **New `plan_hash_value` Column in `V$SQL`**

A new column, `plan_hash_value`, is added to the `V$SQL` view; it contains a hash value built from the cursor plan information.

This new column should be used to compare cursor plans the same way the `hash_value` column is used to compare cursor SQL texts.

For example, if you expect changes in execution plans following a particular event (such as changing the value of a major parameter or migrating to a new release of a database or application), you can take a snapshot of cursor plan information (including the `plan_hash_value` value and `V$SQL_PLAN` content). After the event, you can find out about queries whose plan has changed; that is, for which `plan_hash_value` is different.

## Identifying Unused Indexes

- **Monitoring is done at query parse time.**
- **Turn monitoring on or off when system activity is light.**
- **Main benefits include:**
  - **Space conservation**
  - **Improved performance by eliminating unnecessary overheads during DML operations**

ORACLE

7-27

Copyright © Oracle Corporation, 2001. All rights reserved.

### Identifying Unused Indexes

You can monitor indexes to determine whether they are being used. If an index is not being used, then it can be dropped, thus eliminating unnecessary statement overhead.

Index usage statistics are obtained at query parse time, because that is when you know which indexes are accessed by a particular query.

When users turn the monitor on or off, you can invalidate and recompile those already-compiled cursors dependent on this index, in order to re-collect or stop collecting statistics about which indexes are accessed. It costs some time to activate the monitor in the middle of application execution. Hence, you should turn this feature on or off before starting or after stopping the application, or in periods of light activity.

**Note:** Because Oracle9i collects the statistics at parse time, then the indexes accessed by a SQL statement are not counted if that statement is already being executed when you activate the monitoring feature.

## Enabling and Disabling Index Usage Monitoring

- **Start monitoring the usage of an index:**

```
SQL> ALTER INDEX EMPLOYEES_idx  
2 MONITORING USAGE;
```

- **Stop monitoring the usage of an index:**

```
SQL> ALTER INDEX EMPLOYEES_idx  
2 NOMONITORING USAGE;
```

ORACLE

7-28

Copyright © Oracle Corporation, 2001. All rights reserved.

### Enabling and Disabling Monitoring Index Usage

To start monitoring the usage of an index, issue this statement:

```
ALTER INDEX index_name MONITORING USAGE
```

Later, issue the following statement to stop the monitoring:

```
ALTER INDEX index_name NOMONITORING USAGE
```

## The V\$OBJECT\_USAGE View

- **Displays information about index usage:**
  - **Monitoring on (Yes/No)**
  - **Index is used (Yes/No)**
  - **Start, and begin monitoring, the timeframe**
- **Each time the MONITORING USAGE clause is specified, the view is reset for the specified index.**
- **The view is never purged unless the index is dropped.**

ORACLE

7-29

Copyright © Oracle Corporation, 2001. All rights reserved.

### New View to Support Identifying Unused Indexes

The V\$OBJECT\_USAGE view can be queried for the index being monitored to see whether the index has been used. The view contains a used column whose value is YES or NO, depending on whether the index has been used within the time period being monitored. The view also contains the start and stop times of the monitoring period, and a monitoring column (YES/NO) to indicate whether usage monitoring is currently active.

Each time that you specify MONITORING USAGE, the V\$OBJECT\_USAGE view is reset for the specified index. The previous usage information is cleared or reset, and a new start time is recorded. When you specify NOMONITORING USAGE, no further monitoring is performed, and the end time is recorded for the monitoring period.

Until the next ALTER INDEX . . . MONITORING USAGE statement is issued, the view information is left unchanged unless you drop the corresponding index.

**Note:** The V\$OBJECT\_USAGE view is based on a real data dictionary table. This means that its contents are consistent even after an instance crash. Also note that the view is dynamic; the contents depend on the user querying the view.

## **New View to Support Identifying Unused Indexes (continued)**

V\$OBJECT\_USAGE contains the following columns:

- `index_name`: The index name
- `table_name`: The corresponding table
- `monitoring`: Indicates whether monitoring is ON or OFF
- `used`: Indicates (YES or NO) whether the index has been used during the monitoring time
- `start_monitoring`: The time at which monitoring began on the index
- `end_monitoring`: The time at which monitoring stopped on the index



## Skip Scanning of Indexes

- **Use a composite index when the leading column value is unknown.**
- **Skip scanning is supported for:**
  - **Cluster indexes**
  - **Descending index scans**
  - **CONNECT BY clauses**
- **It is not supported for reverse key or bitmap indexes.**

ORACLE

7-31

Copyright © Oracle Corporation, 2001. All rights reserved.

### Skip Scanning

In prior releases, a composite index is used only if the index prefix (leading) column was included in the predicate of the statement. With Oracle9i, the optimizer can use a composite index even if the prefix column value is not known. The optimizer uses an algorithm called skip scanning to retrieve row IDs for values that do not use the prefix column.

Skip scans reduce the need to add an index to support occasional queries which do not reference the prefix column of an existing index. This can be useful when high levels of DML activity would be degraded by the existence of too many indexes used to support infrequent queries. The algorithm is also valuable in cases where there are no clear advantages to using any particular column as the prefix column in a composite index. The prefix column should be the most discriminating, but also the most frequently referenced in queries. Sometimes, these two requirements are met by two different columns in a composite index, forcing a compromise or the use of multiple indexes.

During a skip scan, the B-tree is probed for each distinct value in the prefix column. Under each prefix column value, the normal search algorithm takes over. The result is a series of searches through subsets of the index, each of which appears to result from a query using a specific value of the prefix column. However, with the skip scan, the value of the prefix column in each subset is obtained from the initial index probe rather than from the command predicate.

The optimizer can use skip scans for processing not only standard B-tree indexes, but also cluster indexes, descending scans, and statements with CONNECT BY clauses. Reverse key and bitmap indexes do not support the skip scan algorithm.

## Example of Skip Scanning

**LANGUAGE and TERRITORY combinations:**

LANGUAGE	TERRITORY
ENGLISH	AMERICA
ENGLISH	CANADA
ENGLISH	UK
FRENCH	CANADA
FRENCH	FRANCE
FRENCH	SWITZERLAND
GERMAN	GERMANY
GERMAN	SWITZERLAND
PORTUGUESE	BRAZIL
PORTUGUESE	PORTUGAL

ORACLE

7-32

Copyright © Oracle Corporation, 2001. All rights reserved.

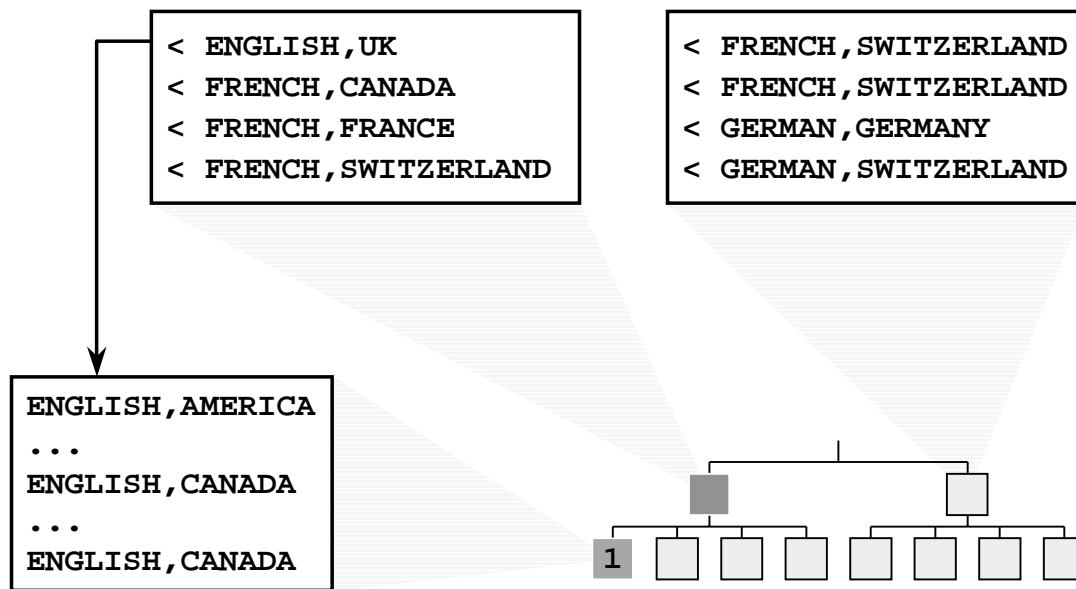
### Example of Skip Scanning

In the example, suppose a composite index exists on the two columns LANGUAGE and TERRITORY, with LANGUAGE as the prefix column. The data values stored in the underlying table result in the combinations of values shown in the table. Each combination can occur multiple times in the table and the resulting index.

In previous releases without the skip scan algorithm, a query on a value in the TERRITORY column was forced to execute a full table scan or a fast full index scan. If such a query is infrequent, this might be acceptable. If the query is more common, then you may have to add a new index on the TERRITORY column alone. This new index, however, could negatively impact the performance of DML operations on the table.

The skip scan solution provides an improvement without the need for the second index. While not as fast as a direct index lookup, the skip scan algorithm is faster than a full table scan in cases where the number of distinct values in the prefix column is relatively low.

## Example of Skip Scanning: Search for SWITZERLAND



ORACLE

7-33

Copyright © Oracle Corporation, 2001. All rights reserved.

### Example of Skip Scanning (continued)

Here is part of the index built on the LANGUAGE and TERRITORY columns, as shown on the previous slide. This portion of the index contains two branch blocks, each of which points to four leaf blocks. The highest value on the leaf block determines the boundary condition for the branch pointers. For example, the first leaf block referenced by the first branch block contains a number of entries with only two distinct values, ENGLISH,AMERICA and ENGLISH,CANADA, both less than ENGLISH,UK. The pointer from the branch block entry to this leaf block is shown on the slide.

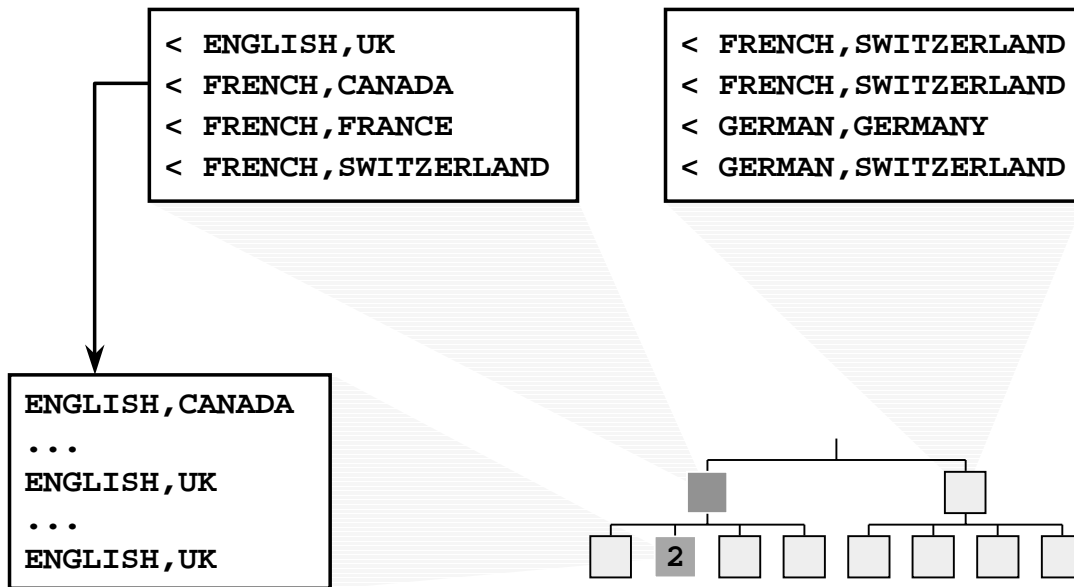
In the next few slides, you will see how the skip scanning algorithm bypasses unneeded leaf blocks when searching for the value SWITZERLAND in the TERRITORY column.

#### First Leaf Block

The index scan begins with the first branch block, which indicates that the first leaf block has values less than ENGLISH,UK.

The specific values on this first leaf block are unknown at this time, so all of the entries are scanned. At the end of the scan of the first leaf block, the potential value ENGLISH,SWITZERLAND, has not been reached.

## Example of Skip Scanning: Search for SWITZERLAND



ORACLE

7-34

Copyright © Oracle Corporation, 2001. All rights reserved.

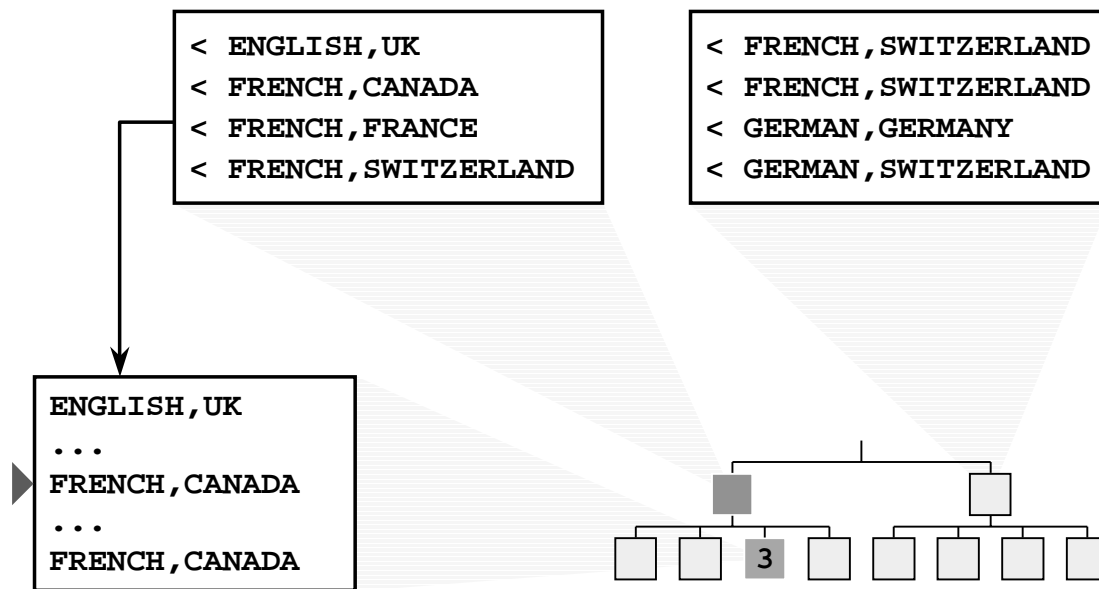
### Example of Skip Scanning (continued)

#### Second Leaf Block

The second entry in the first branch block, combined with the last value read on the first leaf block, imply that the entries on the second leaf block are bounded by the values ENGLISH,CANADA and FRENCH,CANADA.

Since there could be a number of LANGUAGE values, including ENGLISH and FINNISH, in this range, the second leaf block is also scanned for any entries with an TERRITORY value of SWITZERLAND.

## Example of Skip Scanning: Search for SWITZERLAND



ORACLE

7-35

Copyright © Oracle Corporation, 2001. All rights reserved.

### Example of Skip Scanning (continued)

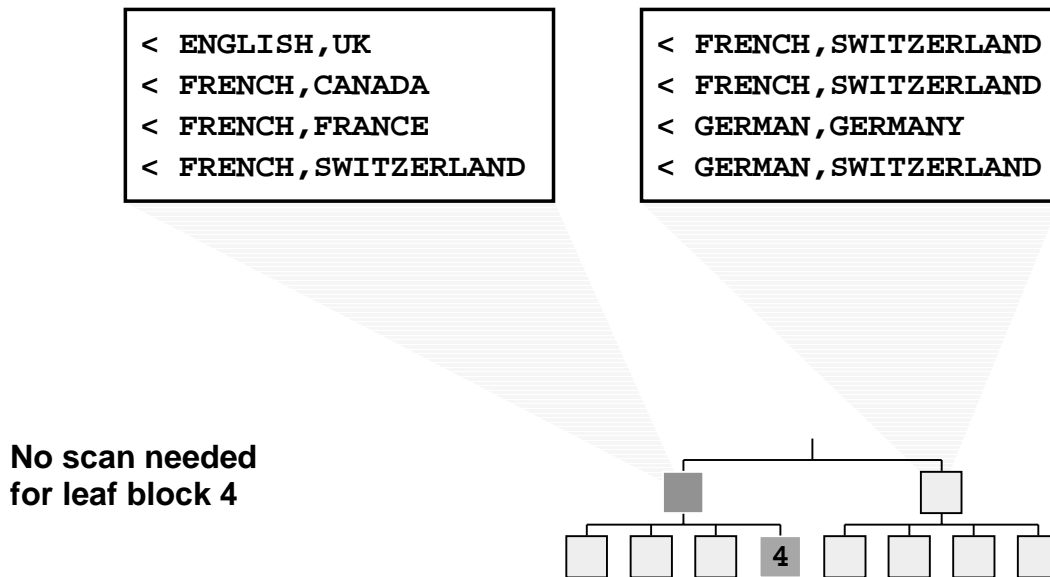
#### Third Leaf Block

The third entry in the first branch block, combined with the last value read on the second leaf block, imply that entries on the third leaf block are bounded by the values ENGLISH,UK and FRENCH,FRANCE.

Although the value ENGLISH,SWITZERLAND would not be in this range, there is still the possibility of values such as FINNISH,SWITZERLAND appearing on this leaf block.

However, as the third leaf block is scanned, the value FRENCH,CANADA is encountered. Because the block does not contain values greater than FRENCH,FRANCE, the value FRENCH,SWITZERLAND cannot be in this leaf block. Furthermore, there can be no other LANGUAGE values besides FRENCH, and so the remainder of the third leaf block can be skipped.

## Example of Skip Scanning: Search for SWITZERLAND



ORACLE

7-36

Copyright © Oracle Corporation, 2001. All rights reserved.

### Example of Skip Scanning (continued)

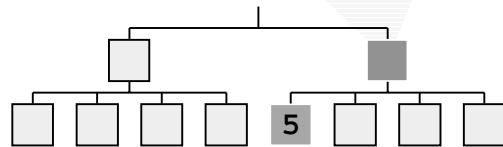
#### Fourth Leaf Block

The fourth entry in the first branch block indicates that the final value on the fourth leaf block must be less than FRENCH,SWITZERLAND. The last value on the third leaf block also has a LANGUAGE value of FRENCH, a fact that can be determined from the previous two branch entries. This implies that the entire block must consist of entries with an LANGUAGE of FRENCH, none of which has an TERRITORY value of “SWITZERLAND.” The scan therefore skips this entire block.

## Example of Skip Scanning: Search for SWITZERLAND

< FRENCH , SWITZERLAND  
< FRENCH , SWITZERLAND  
< GERMAN , GERMANY  
< GERMAN , SWITZERLAND

No scan needed  
for leaf block 5



ORACLE

### Example of Skip Scanning (continued)

#### Fifth Leaf Block

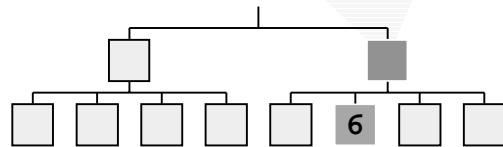
The entries on the first branch block have all been read, so the second branch block is read to find the upper possible value on the fifth leaf block. This entry, just like the last entry on the first branch block, implies that the entire fifth leaf block must consist of entries with a `LANGUAGE` value of `FRENCH`, none of which has a `TERRITORY` value of `SWITZERLAND`.

The scan therefore skips the fifth leaf block.

## Example of Skip Scanning: Search for SWITZERLAND

< FRENCH , SWITZERLAND  
< FRENCH , SWITZERLAND  
< GERMAN , GERMANY  
< GERMAN , SWITZERLAND

No scan needed  
for leaf block 6



ORACLE

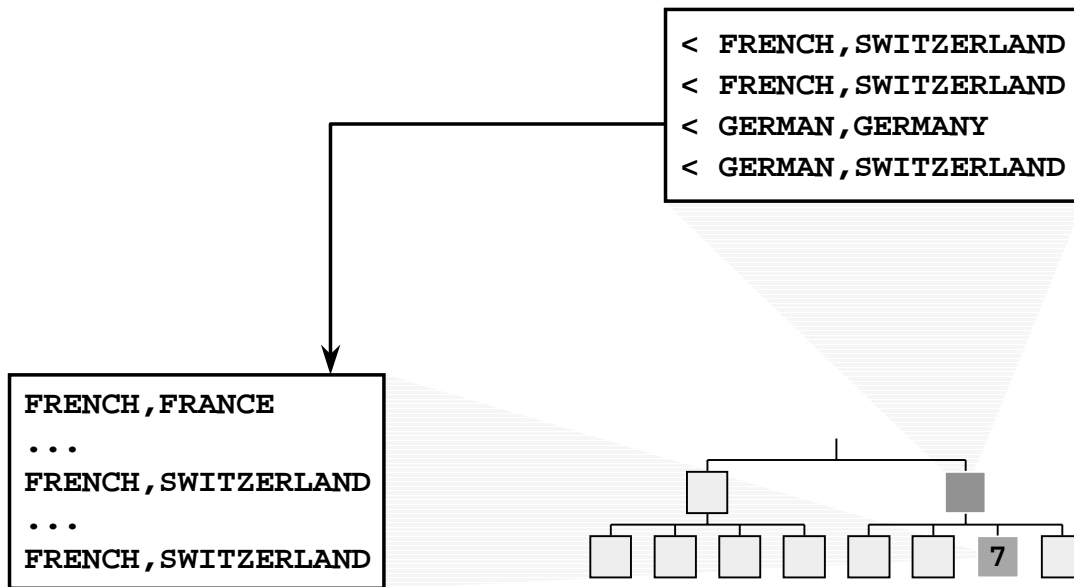
### Example of Skip Scanning (continued)

#### Sixth Leaf Block

The second branch value on the second branch block is exactly the same as the first branch entry. By the same reasoning used for the fifth leaf block, the sixth leaf block is also skipped.



## Example of Skip Scanning: Search for SWITZERLAND



ORACLE

7-39

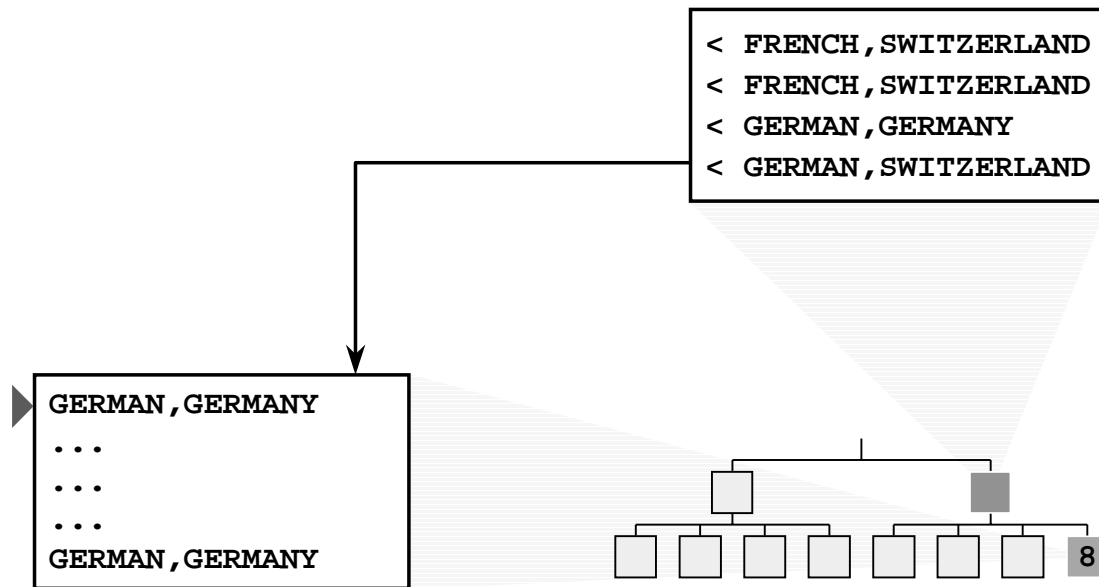
Copyright © Oracle Corporation, 2001. All rights reserved.

### Example of Skip Scanning (continued)

#### Seventh Leaf Block

The third value on the second branch block indicates that the seventh branch block could contain at least two different LANGUAGE values, bracketed by FRENCH and GERMANY. Any one of these LANGUAGE values could have the required value of SWITZERLAND in the TERRITORY column, except GERMANY, which is excluded by the upper boundary value. The seventh leaf block must therefore be read.

## Example of Skip Scanning: Search for SWITZERLAND



ORACLE

7-40

Copyright © Oracle Corporation, 2001. All rights reserved.

### Example of Skip Scanning (continued)

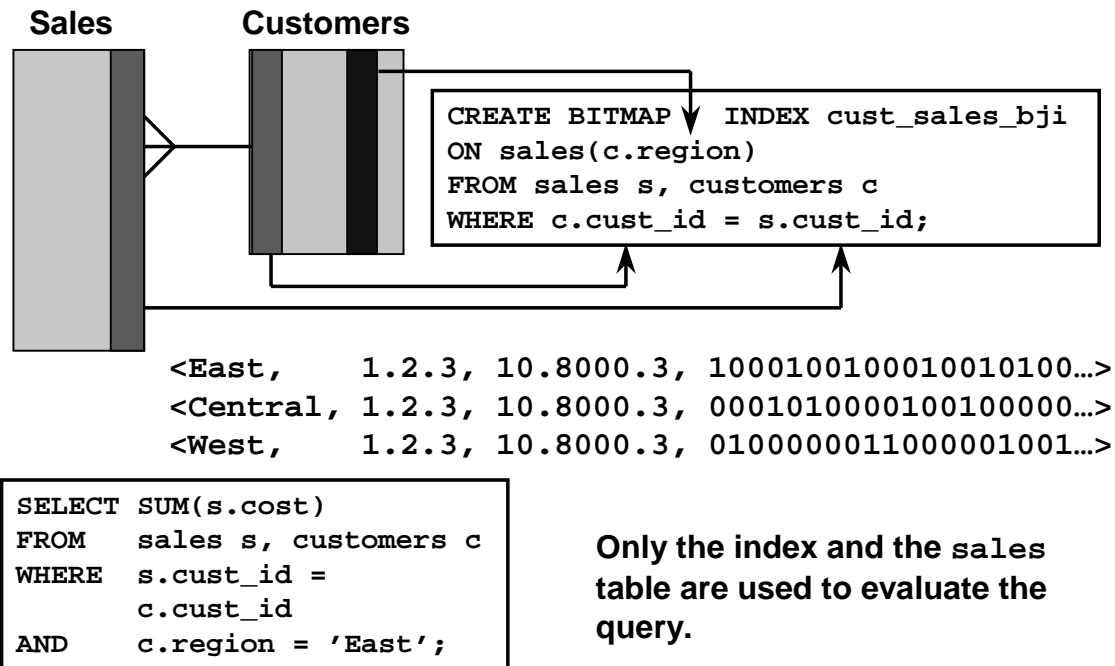
#### Eighth Leaf Block

The final value on the second leaf block indicates that entries on eighth leaf block with a LANGUAGE value of GERMANY do not contain the required value of SWITZERLAND in the TERRITORY column. However, it is not possible to tell whether there are more entries with another language value in this eighth leaf block, so the scan reads the block. But, after finding that the first entry has the value of GERMANY in the LANGUAGE column, the scan can skip the rest of the eighth leaf block.

#### Summary

Even in this very small example, you can see that almost half the leaf blocks are skipped by the index skip scanning algorithm, and the entire index has not been examined. Obviously, different results are obtained depending on the range and distribution of the values in the index columns. The optimizer uses statistics to determine whether a skip scan retrieval is more efficient than a full table scan, or other possible retrieval paths, when parsing SQL statements.

## Bitmap Join Indexes (BJIs)



ORACLE

7-41

Copyright © Oracle Corporation, 2001. All rights reserved.

### Bitmap Join Indexes (BJIs)

In addition to a bitmap index on a single table, you can create a bitmap join index (BJI) in Oracle9i. A BJI is a bitmap index for the join of two or more tables. A bitmap join index is a space-efficient way of reducing the volume of data that must be joined by performing restrictions in advance.

As you can see on the above slide, Oracle9i introduces a new CREATE BITMAP INDEX syntax allowing you to specify a FROM and a WHERE clause.

Here, we create a new BJI named cust\_sales\_bji on the sales table. The key of this index is the region column of the customers table.

This example assumes that there is a primary key–foreign key relationship between sales and customers. The cust\_id column is the primary key of customers but is also a foreign key inside sales.

The FROM and WHERE clauses in the CREATE statement allow Oracle9i to make the link between the two tables. They represent the natural join condition between the two tables.

The middle part of the above graphic shows you a theoretical implementation of this BJI. Each entry or key in the index represents a possible region found in the customers table. A bitmap is then associated to one particular key. The meaning of the bitmap is obvious, because it is the same representation as for traditional bitmap indexes. Each bit in a bitmap corresponds to one row in the sales table. The first key above (East) shows that the first row in the sales table corresponds to a product sold to an East customer, while the second bit represents a product not sold to an East customer.

### **Bitmap Join Indexes (BJIs) (continued)**

The interest of this structure becomes clear with the last part of the slide. Indeed, when the user tries to find what the total cost is of all sales for the East customers, Oracle9i can just use the above BJI and the `sales` table to answer the question. In this case, there is no need to compute the join between the two tables explicitly. Using the BJI in this case is much faster than computing the join at query time.

## Advantages and Disadvantages of BJIs

- **Advantages:**
  - Bitmap join indexes provide good performance for join queries, and are space-efficient.
  - They are especially useful for large dimension-tables in star schemas.
- **Disadvantages:**
  - More indexes are required: Up to one index per dimension-table column, rather than one index per dimension table
  - Maintenance costs are higher: Building or refreshing a bitmap join index requires a join

ORACLE

7-43

Copyright © Oracle Corporation, 2001. All rights reserved.

### Advantages and Disadvantages of Bitmap Join Indexes

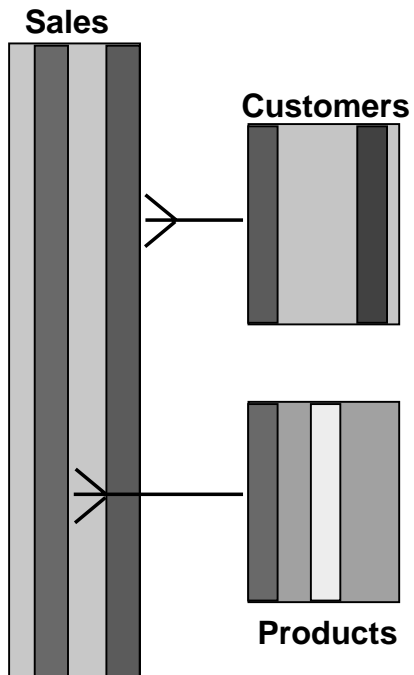
The volume of data that must be joined can be reduced if BJIs which are used as joins have already been precalculated. In addition, BJIs which contain more than one dimension table can eliminate bitwise operations, which are necessary in the star transformation's use of bitmap indexes.

An alternative to a BJI is a materialized join view, which is the materialization of a join in a table. Compared to a materialized join view, a BJI is much more space-efficient, because it compresses row IDs of the fact tables.

Also, queries using BJIs can be sped up by means of bitwise operations.

On the other hand, you may need to create more BJIs on the fact table to satisfy the maximum number of different queries. In fact, you may have to create one BJI for each column of the corresponding dimension tables. Of course, having many indexes on one table results in higher maintenance costs, especially when the fact table is updated.

## BJI: A More Complex Example



```
CREATE BITMAP INDEX c_s_p_bji
ON    sales(c.gender,p.category)
FROM  sales s, customers c
      , products p
WHERE  c.cust_id = s.cust_id
AND    p.prod_id = s.prod_id
```

```
SELECT SUM(s.cost)
FROM    sales s, customers c
      , products p
WHERE   s.cust_id = c.cust_id
AND     s.prod_id = p.prod_id
AND     c.gender  = 'MALE'
AND     p.category = 'MOBILE';
```

ORACLE

### BJI: A More Complex Example

You can create a BJI on more than one table, as in the above slide, where `c_s_p_bji` is a bitmap join index on the `sales` table which indexes two columns in two different tables: Gender in `customers` and category in `products`.

The above `SELECT` statement that retrieves the total mobile sales for male customers can benefit the `c_s_p_bji` bitmap join index.

## Restrictions on BJIs

- **Parallel DML is supported only on the fact table.**
- **Only one table can be updated concurrently.**
- **No table can appear twice in the join.**
- **There can be no index-organized tables or temporary tables.**
- **Keys must all be part of the dimension tables.**
- **The dimension-table join columns must either be primary key columns or have unique constraints.**
- **If a dimension table has a composite primary key, then each key column must be part of the join.**

ORACLE

7-45

Copyright © Oracle Corporation, 2001. All rights reserved.

### Restrictions on Bitmap Join Indexes

Due to the necessity of storing the results of join results, BJIs have the following restrictions:

- Parallel DML is currently only supported on the fact table. Parallel DML on one of the participating dimension tables marks the BJI as unusable.
- Only one table can be updated concurrently by different transactions when using a BJI.
- No table can appear twice in the join.
- You cannot create a BJI on an index-organized table or a temporary table.
- The BJI columns must all be columns of the dimension tables.
- The dimension-table join columns must either be primary key columns or have unique constraints.
- If a dimension table has composite primary key, each column in the primary key must be part of the join.

# Summary

In this lesson, you should have learned how to:

- Use the new cost-based optimizer (CBO) model
- Use new first rows optimization
- Use the stored outlines editor
- Use the extensions to cursor sharing
- Access cached execution plans
- Use index skip scanning optimization
- Monitor index usage

ORACLE



# 8

## Partitioning Enhancements

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# Objectives

**After completing this lesson, you should be able to:**

- **Use list partitioning**
- **Maintain global indexes during DDL execution**
- **Use parallel direct-load enhancements**

ORACLE

## List Partitioning Overview and Benefits

- **List partitioning is a new partition method introduced in Oracle9i, where you control how rows map to partitions.**
- **It allows for the distribution of data based on discrete column values.**
- **Unordered and unrelated sets of data can be grouped together.**
- **There is no relationship between partitions.**
- **This method is ideal for columns that consist of discrete values; it also provides powerful data-management capabilities.**

ORACLE

### List Partitioning Overview and Benefits

The Partitioned Objects feature has incrementally added new partition methods to the Oracle RDBMS over two releases: Oracle8 and Oracle8i. There are, however, constantly emerging requirements generated by customers who cannot take full advantage of the Oracle8i partition methods provided so far because their data model does not dovetail with those methods.

Thus, Oracle9i adds a new partitioning model called *list partitioning* to the set of partition methods already being supported in the Oracle RDBMS. The list method allows explicit control over how rows map to partitions, by specifying a list of discrete values for the partitioning column in the description of each partition. This is different from *range partitioning*, where a range of values is associated with a partition, and *hash partitioning*, where the user has no control of row-to-partition mapping. This partition method has been specifically added to model data distributions that follow discrete values, which cannot be done easily in Oracle8i.

## Example of List Partitioning

```
CREATE TABLE locations
( location_id, street_address, postal_code
, city, state_province, country_id)
PARTITION BY LIST (state_province)
STORAGE(INITIAL 10K NEXT 20K) TABLESPACE tbs5
( PARTITION region_east
VALUES ('MA','NY','CT','NH','ME','MD','VA','PA','NJ')
STORAGE (INITIAL 20K NEXT 40K PCTINCREASE 50)
TABLESPACE tbs8
, PARTITION region_west
VALUES ('CA','AZ','NM','OR','WA','UT','NV','CO')
PCTFREE 25 NOLOGGING
, PARTITION region_south
VALUES ('TX','KY','TN','LA','MS','AR','AL','GA')
, PARTITION region_central
VALUES ('OH','ND','SD','MO','IL','MI', NULL, 'IA')
)
```

ORACLE

8-4

Copyright © Oracle Corporation, 2001. All rights reserved.

### Example of List Partitioning

The details of list partitioning can best be described with an example. In this case, you want to partition the `locations` table by region, that is, group states together according to their geographical location.

A row is mapped to a partition by checking whether the value of the partitioning column for a row falls within the set of values that describes the partition.

For example, the following rows are inserted as follows:

- (1500, '2011 Interiors Blvd', '99236', 'South San Francisco', 'CA', 'US') maps to the `region_west` partition.
- (5000, 'Chemin de la fanee', '13840', 'ROGNES', 'BR', 'FR') does not map to any partitions in the table.

In the above example, the partitions with specified physical attributes override the table-level defaults; however, any partitions with unspecified attributes inherit their physical attributes from the table-level defaults.

**Note:** For formatting reasons, the column data types are not specified. Refer to the sample schema definition for more details on those data types.

## List Partitioning Pruning

Here are the three pruning types supported for list-partitioned tables:

```
SQL> SELECT * FROM LOCATIONS  
2 WHERE state_province = 'CA';
```

```
SQL> SELECT * FROM LOCATIONS  
2 WHERE state_province IN ('CA','NY');
```

```
SQL> SELECT * FROM LOCATIONS  
2 WHERE state_province <= 'AZ';
```

ORACLE

### List Partitioning Pruning

One feature of list partitioning is that there is no apparent order of partitions (unlike in range partitioning). Nevertheless, Oracle9i supports partition pruning for objects partitioned by the list method, for queries involving the following predicates on the partitioning key:

- Equality: Only the corresponding partition is accessed. Based on the `locations` table and the first query above, only the `region_west` partition is accessed.
- In-List: Only the corresponding partitions are accessed. Based on the `locations` table and the second query above, only the `region_west` and `region_east` partitions are accessed.
- Range: Only partitions that contain literal values in their list that correspond to the range predicate are accessed. Based on the `locations` table and the third query above, only the `region_west` partition and the `region_south` partition are accessed.

## ALTER TABLE ADD PARTITION Example

**For this example to work, no value in the set of literal values that describes the partition being added can exist in any of the other partitions of the table:**

```
SQL> ALTER TABLE locations
  2  ADD PARTITION region_nonmainland
  3          VALUES ( 'HI', 'PR' )
  4  STORAGE (INITIAL 20K NEXT 20K)
  5  TABLESPACE tbs_3
  6  NOLOGGING;
```

ORACLE

8-6

Copyright © Oracle Corporation, 2001. All rights reserved.

### ALTER TABLE ADD PARTITION Example

In Oracle9i, the syntax and semantics of this DDL statement is modified to support adding a single partition to a table partitioned using the list method. This just means that the user is adding a new partition to the set of partitions of a table. However, there is no ordering among the partitions. The newly created partition has the following characteristics:

- It has no data.
- A partition name, a set of literal values describing the partition value list, physical attributes, and logging attributes can be specified.
- Every literal value in the set that describes the partition value list for a partition has to be a unique value specified for that object. If there is a duplicate entry, an error is returned.
- If any physical attributes of the partition are not specified, they are derived by combining default table-level attributes with the default attributes of the tablespace in which the partition will be placed.
- If a partition name is not specified, a system-generated name of form SYS\_P### is assigned.
- Adding a partition to a table also add a corresponding index partition (with the same value list) to all local indexes defined on the table. Global indexes are not affected.

### **ALTER TABLE ADD PARTITION Example (continued)**

- ALTER TABLE ADD PARTITION locks the list-partitioned table in Exclusive mode (X) for the duration of the operation. This is a fast dictionary operation, and no other DDL or DML operation is permitted on the table.

The above example adds a new partition to the `locations` table partitioned by the list method. The example specifies some new physical attributes for this partition, and uses the table-level defaults for the others.

## ALTER TABLE MERGE PARTITION Example

**This DDL command can be used to merge the contents of any two arbitrary partitions of a list-partitioned table:**

```
SQL> ALTER TABLE locations
  2  MERGE PARTITIONS
  3  region_northwest, region_southwest
  4  INTO PARTITION region_west
  5  PCTFREE 50 STORAGE(MAXEXTENTS 20);
```

ORACLE

8-8

Copyright © Oracle Corporation, 2001. All rights reserved.

### ALTER TABLE MERGE PARTITION Example

The syntax of this operation remains unchanged for a list-partitioned table compared to a range-partitioned table. But the semantics are modified to operate on tables partitioned using the list method. The statement can be used to merge the contents of any two arbitrary partitions of a table; the candidate partitions do not necessarily have to be adjacent, because list partitions do not assume any order for partitions. The resulting partition consists of the union of the set of values that formed the two partitions being merged.

Usage:

- If a name for the resulting partition is not specified, a name of form SYS\_P# is assigned.
- Any two partitions can be merged.
- The resulting partition value list is the union of the partition value lists of the two partitions being merged.
- The data in the resulting partitions consists of data from both the partitions.
- When merging the partitions of tables partitioned using the list method, the tablespace in which the resulting partition is located and the partition's attributes are determined by the table-level default attributes, except for those specified explicitly.
- The corresponding local index partitions are also merged, and the resulting local index partition is marked Index Unusable.



### **ALTER TABLE MERGE PARTITION Example (continued)**

- All global indexes defined on the table are marked Index Unusable. These include both partitioned and nonpartitioned indexes.
- ALTER TABLE MERGE PARTITIONS locks the table in SX mode and the partitions being merged in X mode.

The above example merges two partitions of the `locations` table, partitioned using the list method, into a single partition which inherits all of its attributes from the table-level default attributes, except for `PCTFREE` and `MAXEXTENTS` which are specified in the statement.

If:

- The value list for the `region_northwest` partition is (`'CA'` , `'OR'` , `'NV'` , `'UT'` )
- The value list for the `region_southwest` partition is (`'AZ'` , `'NM'` , `'CO'` , `'WA'` )

then the resulting partition value list is the union of these two lists; in other words, the `region_west` partition value list is

(`'CA'` , `'OR'` , `'NV'` , `'UT'` , `'AZ'` , `'NM'` , `'CO'` , `'WA'` ).

**Note:** This example supposes that at some point the initial partition `region_west` was split into `region_northwest` and `region_southwest`.

## ALTER TABLE MODIFY PARTITION ADD VALUES Example

**Specified new literal values must not already exist in any of the partition's value lists:**

```
SQL> ALTER TABLE locations
2  MODIFY PARTITION region_south
3  ADD VALUES ( 'OK' , 'KS' );
```

ORACLE

8-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### ALTER TABLE MODIFY PARTITION ADD VALUES Example

This is a new statement which can be used to extend the partition value list of an existing partition to contain additional literal values. The statement is used to describe the literal values that are being added to an existing partition value list. None of the new literal values can have been previously specified in any of the partition's value lists that describe the partition table.

The above example adds a new set of state codes ( 'OK' , 'KS' ) to the existing `region_south` partition list.

Usage:

- The set of literal values being added to the partition must be unique within the set of literal values that describes the partition value list, for all existing partitions. If duplicate values are found, an error is returned.
- The partition literal value list for the corresponding local index partition is also naturally extended.
- The status (Index Unusable) of the Local and Global index partitions are not affected by this operation.
- This is a fast operation, and during the operation a DML Share-Exclusive (SX) lock is acquired on the table and a DML Exclusive (X) lock is acquired on the partition being modified.

## ALTER TABLE MODIFY PARTITION DROP VALUES Example

**There must be no rows in the partition for the literal values being dropped:**

```
SQL> DELETE FROM locations
      2 WHERE state_province
      3       IN ('OK', 'KS');
```

```
SQL> ALTER TABLE locations
      2 MODIFY PARTITION region_south
      3 DROP VALUES ('OK', 'KS');
```

ORACLE

8-11

Copyright © Oracle Corporation, 2001. All rights reserved.

### ALTER TABLE MODIFY PARTITION DROP VALUES Example

This is a new statement which can be used to prune the partition value list of an existing partition. It removes a set of literal values from an existing set of literal values that describes the partition's value list. It is important to note that this operation expects no rows to exist in the partition for the literal values being dropped; however, it checks for the existence of rows in the partition that correspond to the literal values being dropped, and fails with an error message if any such rows are found. Thus, the user must drop the corresponding rows first.

The above statement drops a set of state codes ('OK', 'KS') from the existing `region_south` partition value list. This operation also drops the corresponding values from the dictionary.

Usage:

- The set of literal values being dropped from the partition has to be a subset of the set of literal values that describes the current partition value list. Otherwise an error message is returned.
- This operation cannot be used to drop all the values that correspond to a partition. You should use `ALTER TABLE DROP PARTITION` instead.

### **ALTER TABLE MODIFY PARTITION DROP VALUES Example (continued)**

- The partition is checked for any row whose value for the partitioning key corresponds to any of the values being dropped. An error is returned and the operation fails if such a row is found. The user must issue a DELETE DML statement to remove the rows that correspond to the values being dropped before re-issuing the command.
- The dictionary is also updated to reflect the new partition value list for the partition being modified.
- During the operation a DML Row Exclusive (SX) lock is acquired on table and a DML Exclusive (X) lock is acquired on partition.
- The DDL timestamps of the table and the partition are updated.
- Because a query is executed to check for the existence of rows in the partition that correspond to the literal value being dropped from the partition, you should create a local prefixed index on the table. This speeds up the execution of the query and the overall operation.
- The status (Index Unusable) of local indexes remains unaffected, however the index partition being affected retains the new partition value list. Global indexes remain unaffected by this operation.

## ALTER TABLE SPLIT PARTITION Example

- You specify the values for the first new partition.
- The second new partition holds the remaining values.

```
SQL> ALTER TABLE locations
  2  SPLIT PARTITION region_east
  3  VALUES ( 'CT' , 'VA' , 'MD' )
  4  INTO ( PARTITION region_east_1
  5          TABLESPACE tbs2
  6          , PARTITION region_east_2
  7          STORAGE (NEXT 2M )
  8          )
  9  PARALLEL 5;
```

ORACLE

8-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### ALTER TABLE SPLIT PARTITION Example

The syntax of this statement is modified to support list-partitioned tables. The semantics of this statement is also extended to allow a single partition value list to be split up into two distinct partitions with nonoverlapping value lists. The syntax is exactly the same as that for splitting range partitions, except that the AT keyword is replaced with the VALUES keyword.

The list of values following the VALUES clause applies to the first partition of the two new partitions being created. The list of values for the second partition is obtained by subtracting the first new partition literal values list from the original literal values list of the partition being split.

Additionally, the new partitions can have specified partition names, physical attributes, and logging clauses.

If the INTO clause is not specified, system-generated names are used for the two new partitions.

The above example splits the `region_east` partition into two partitions: `region_east_1` with a literal value list of ( 'CT' , 'VA' , 'MD' ), and `region_east_2` inheriting the remaining literal values ( 'NY' , 'NH' , 'ME' , 'MA' , 'PA' , 'NJ' ). The individual partitions have new physical attributes specified at the partition level. This operation is run with a parallelism of degree 5.

## **ALTER TABLE SPLIT PARTITION Example (continued)**

### Usage:

- The set of literal values specified for the first new partition must be a subset of the set of literal values that describes the current partition value list being split. Otherwise an error message is returned.
- No new literal values can be added to the partition description other than those that existed for the partition being split.
- After the split, rows with partitioning key values equal to the specified VALUES list will be inserted into the first partition. Rows with partitioning keys not equal to the specified VALUES list are inserted into the second partition.
- The new partition inherits all unspecified physical attributes from the partition being split (not from the table-level default).
- This statement also performs a matching split on the corresponding partition in each local index defined on the table. The index partitions are split even if they are marked Index Unusable.
- New local index partitions are assigned the same name as that of the corresponding new base table partition, except for indexes which already have a partition with such a name, in which case a name is generated automatically with the form SYS\_Pn.
- With the exception of the TABLESPACE attribute, the physical attributes of the local index partition being split are used for both new index partitions. If the parent local index lacks the default TABLESPACE attribute, new local index partitions reside in the same tablespaces as the corresponding new partitions of the underlying table.
- All global indexes are marked Index Unusable if the partition is not empty.
- ALTER TABLE SPLIT PARTITION locks the table in Row Exclusive (SX) mode and locks partition in Exclusive (X) mode.

## List Partitioning Usage

- **Currently supported only for heap tables**
- **Multicolumn partitioning not supported**
- **The specified literal values must be unique across all literal values of all partition value lists of the object**
- **NULL can be specified as a partition literal value**
- **MAXVALUE cannot be specified**
- **All lists must have at least one literal**
- **List of literals cannot exceed 4KB**
- **Partition pruning, partition wise joins, and parallelism are supported**
- **Local indexes and global range partitioned indexes are supported**

ORACLE

8-15

Copyright © Oracle Corporation, 2001. All rights reserved.

### List Partitioning Usage

In general, all semantics that apply for the range method also apply for the list method. Here are some specific features that apply to the list partition method:

- The list method is not extended to IOT's in Oracle9i.
- Multicolumn partitioning is not supported for list partitioning, unlike for range and hash partitioning. If a table is partitioned by list, the partitioning key must consist of a single column of the table. Other than that, all columns that can be partitioned by range or hash can be partitioned by list.
- The partitions do not have any implicit ordering, as in range partitioning, and hence can be specified in any order.
- The specified literal values that define the partitioning criterion of any given partition must be unique across all literal values of all partition value lists of the object. That is, a literal value for a partition value list (such as 'AZ') cannot be specified as a value in any other partition. If there are duplicate values, an error is returned.
- The NULL keyword can be specified as a value element of this list, so that null values for a partitioning key can map to a specific partition. However, be cautious with predicates using the IN-LIST clause that involves null values, because SQL language semantics treats the null literal different from other literal values. Queries that test equality predicates for null literals should be evaluated using existential (is) predicates, as opposed to regular equality (=) predicates.

### **List Partitioning Usage (continued)**

- The MAXVALUE literal can no longer be specified as a partition literal value, because it has no real meaning.
- The set of literal values that describes a partition value list must have at least one element; that is, it cannot be empty.
- The string comprising the list of values for a partition cannot exceed 4 KB.



# Maintaining Global Indexes

- **UPDATE GLOBAL INDEXES provides the capability to update a global index during DDL operations.**
- **Benefits include:**
  - **Usability: Partition DDL becomes more user friendly**
  - **Availability: Database becomes more available**
  - **Manageability: Global indexes are maintained with the base table**

ORACLE

8-17

Copyright © Oracle Corporation, 2001. All rights reserved.

## Maintaining Global Indexes During DDL

Prior to Oracle9i, when DDL operations were issued, index partitions that corresponded to the data partitions affected by the DDL became invalidated. The database administrator was then required to rebuild the indexes. This was not a problem for local indexes, because the DBA only needed to rebuild the local index of the partitions. Unlike local indexes, however, the entire global index became invalidated when a partition DDL was performed, even though the partition DDL might affect only a small fraction of the data in the table. In some cases, this rebuilding the global index was a very costly operation. Being able to update global indexes automatically during DDL execution eliminates this problem.

### Benefits

**Usability:** This feature makes partition DDL much more user friendly, because there is no need to look up the names of all invalid global indexes and then issue DDL commands to rebuild them individually. This feature replaces a two-step process (partitioning DDL, then rebuilding the index) with a one-step process.

**Availability:** Using this feature makes the database more available. If a partition DDL is issued and specifies the UPDATE GLOBAL INDEXES clause, all global indexes are available throughout the operation. Otherwise, global indexes are marked invalid, and remain invalid until they are rebuilt. In practice, this has a severe impact on performance and availability for applications accessing this partitioned table.

**Manageability:** With this feature, global indexes are maintained in conjunction with the base table. Therefore, they do not have to be rebuilt independently.

## Maintaining Global Indexes

- **UPDATE GLOBAL INDEXES** is allowed after the **partition specification clause** and before the **parallel clause**:

```
SQL> ALTER TABLE employees
2 DROP SUBPARTITION EMP1
3 UPDATE GLOBAL INDEXES
4 PARALLEL (DEGREE 4);
```

- The **parallel clause** is allowed in conjunction with the syntax for **DROP**, **TRUNCATE**, and **EXCHANGE**.

ORACLE

8-18

Copyright © Oracle Corporation, 2001. All rights reserved.

### Maintaining Global Indexes During DDL (continued)

The **UPDATE GLOBAL INDEXES** clause is an optional clause and must be placed immediately after the partition specification clause and before the **PARALLEL** clause.

For **EXCHANGE PARTITION**, only global indexes on the table whose partition is being exchanged are updated. Indexes for the table being exchanged continue to be invalidated.

## Maintaining Global Indexes

- **UPDATE GLOBAL INDEXES** is available when executing the following partition DDL statements: **ADD, DROP, MOVE, TRUNCATE, SPLIT, MERGE, EXCHANGE, and COALESCE.**
- **Only valid indexes are updated.**
- **The use of this clause is not supported with index-organized tables (IOTs).**

ORACLE

## Performance Considerations: Updating versus Rebuilding an Index

- **Updating:**
  - Partition DDL takes longer to complete.
  - DROP, TRUNCATE, and EXCHANGE are not fast, data-dictionary-only operations.
  - Updates to the global index are logged.
  - Favorable when the amount of row work is low
  - Ensures that performance does not drastically fall
- **Rebuilding:**
  - Makes index more efficient
  - Enables the user to reorganize the index

ORACLE

8-20

Copyright © Oracle Corporation, 2001. All rights reserved.

### Updating versus Rebuilding an Index

#### Updating

- Partition DDL takes longer to complete because the global indexes that were previously marked invalid are be updated.
- DROP, TRUNCATE, and EXCHANGE are no longer fast, data-dictionary-only operations, because all rows in the partition are scanned.
- Updates to the global index are logged, hence redo and rollback information is generated.
- Updating an index is favorable when the amount of row work is low.
- Updating an index ensures that application performance does not drastically fall until the index is rebuilt.

#### Rebuilding

- Rebuilding the entire index makes the index more efficient.
- Rebuilding the index enables the user to reorganize the index.

## Parallel Direct-Load Inserts versus Partitioned Tables Enhancements

- Since Oracle8, each partition of a table is acted upon by only one slave during a parallel direct-load operation.
- With Oracle9i, multiple slave processes can insert data in one partition.
- The main benefit is that load balancing is possible with data skew amongst partitions. This results in better performance.

ORACLE

8-21

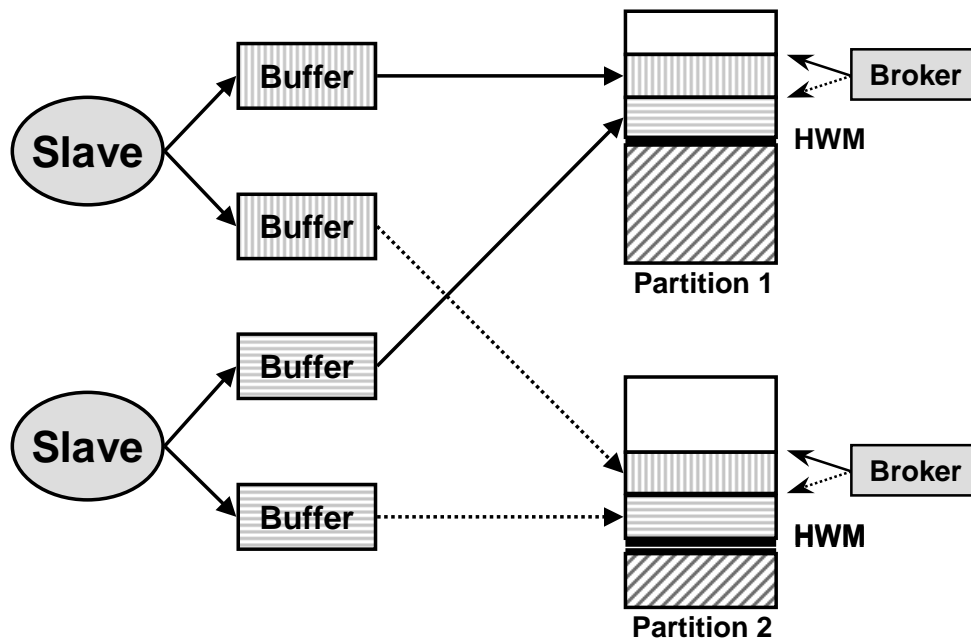
Copyright © Oracle Corporation, 2001. All rights reserved.

### Parallel Direct-Load Inserts Versus Partitioned Tables Enhancements

Since Oracle8, a parallel direct-load insert operation over a partitioned table can use only one slave process per partition. As a result, load balancing is not possible if there is a data skew among the partitions. For example, in many applications the table is partitioned by range on a date column, and the rows are inserted mainly in the last partition. Due to this, the slave working on the last partition needs to do much more work than other slaves.

Allowing multiple slaves to work on a partition helps to alleviate this performance bottleneck. It takes advantage of the dynamic load balancing capabilities of parallel execution.

## DML Intra-Partition Parallelism



ORACLE

8-22

Copyright © Oracle Corporation, 2001. All rights reserved.

### DML Intra-Partition Parallelism

In Oracle8i, during parallel execution of an insert into a partitioned table, there are two active slave sets:

- The query slave set which reads the data
- The DML slave set which inserts the data

Each slave of the DML slave set works on one partition of the table. The inverse is also true; that is, each partition is allocated to at most one slave from this slave set.

The rows read by the query slaves have to be repartitioned and redistributed amongst the DML slaves so that each DML slave gets rows that correspond to the partition it is populating. These rows are inserted above the High Water Mark (HWM) of the segment of the corresponding partition by the DML slave.

With the ability of allowing intra-partition parallelism in inserts, Oracle9i removes this repartitioning phase altogether. The new mode of operation is represented on the slide above. Each slave does the query as well as the DML part of the execution. And as more than one slave can work simultaneously on the target partitions, the repartitioning phase is not required.

### **DML Intra-Partition Parallelism (continued)**

Each slave while executing the query, buffers the results in an in-memory buffer (one for each partition that it can insert into). Whenever a buffer fills up, the slave exactly knows the amount of space it needs for that partition. The slaves coordinate their activities on each segment through a “broker enqueue.” With each segment, Oracle9i maintains an enqueue which holds the address above the HWM above which no space has been reserved by any slave yet, so space allocations above the HWM by multiple slaves cannot cause conflicts.

# Summary

**In this lesson, you should have learned how to:**

- **Use list partitioning**
- **Maintain global indexes during DDL execution**
- **Use parallel direct-load enhancements**

ORACLE





# Data Warehouse Enhancements

ORACLE<sup>®</sup>

Copyright © Oracle Corporation, 2001. All rights reserved.

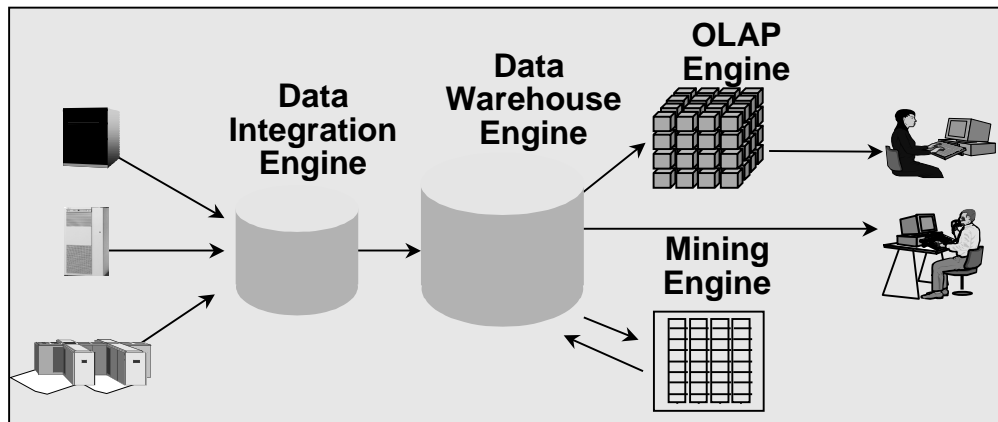
# Objectives

**After completing this lesson, you should be able to:**

- **Use full extraction**
- **Use incremental extraction**
- **Transport data using flat files**
- **Transport data using transportable tablespaces**
- **Load data using pipelined transformations**
- **Access external tables**
- **Transform data with SQL**
- **Use new features of materialized views**

ORACLE

## The Traditional Way: Fragmented Information Supply Chain



**Protracted implementation and maintenance cycle:**

- **Synchronization and currency issues**
- **Information-management chaos**

ORACLE

9-3

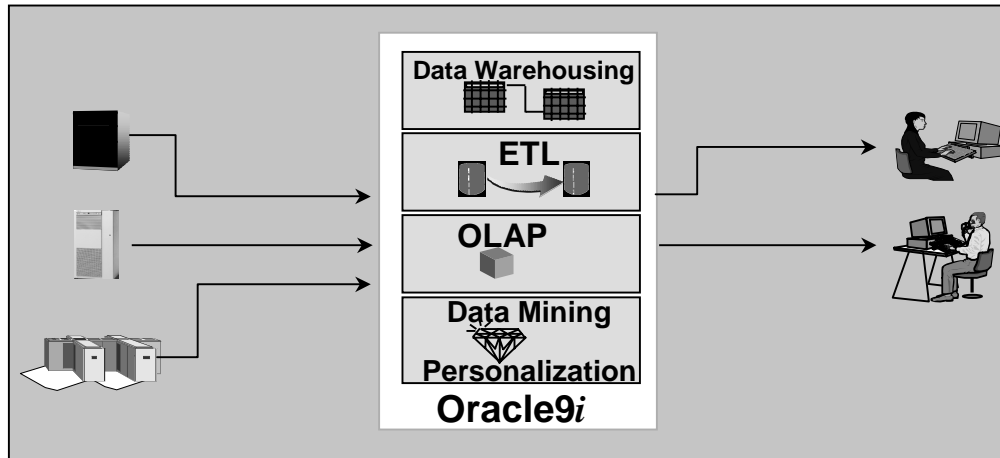
Copyright © Oracle Corporation, 2001. All rights reserved.

### The Traditional Way: Fragmented Information Supply Chain

Prior to Oracle9i, building a business intelligence system required the integration of multiple server products.

The result was that such systems were unnecessarily complex. The integration of multiple servers was costly. Once the system was implemented, there were ongoing administration costs in maintaining different servers and keeping the data synchronized across all servers.

## The New Way: Oracle9i



**Single business intelligence platform:**

- Reduced administration and implementation costs
- Faster deployment
- Improved scalability and reliability

ORACLE

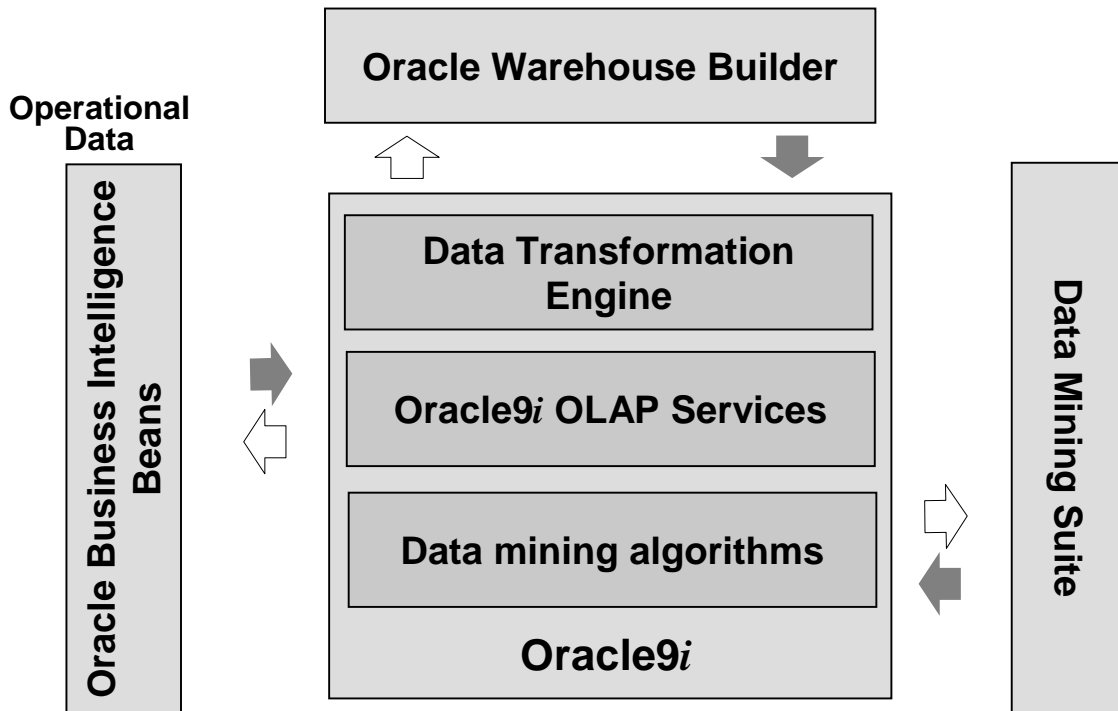
### The New Way: Oracle9i

Oracle9i is the solution to this information-management problem.

Oracle9i provides a single server platform for all business intelligence needs. All data is stored in the relational database. All administration is done through Oracle Enterprise Manager. All business intelligence processes benefit from the scalability and reliability of Oracle9i.

With Oracle9i, implementing and administering business intelligence systems is faster and less costly.

## The New Way: Oracle9i



ORACLE

9-5

Copyright © Oracle Corporation, 2001. All rights reserved.

### The New Way: Oracle9i

With Oracle Warehouse Builder, you can design and manage Extraction Transformation Loading (ETL) processes.

The ETL Transformation Engine is scalable (parallel), extensible (Java, PL/SQL), and efficient (no data staging). Oracle9i has been significantly enhanced to address many of the specific requirements of ETL environments. These enhancements make Oracle9i a fully functional transformation engine.

These new features have four main characteristics:

- **Scalable:** Enterprise data warehouses can transform many gigabytes of data in a single load cycle; the ability to work in parallel is a crucial requirement (and is one that some proprietary transformation solutions are lacking)
- **Efficient:** Each of the new Oracle9i features improves the efficiency of doing common ETL operations in the relational database.
- **Open:** The ETL features are extensions to SQL. Any customer or tool can leverage these new features in their own ETL environments.
- **Extensible:** Transformations can be arbitrarily complex; extensibility is important so that you can implement you own transformations. The Oracle database supports Java and PL/SQL procedures (as well as C and C++ callouts), and in Oracle9i these procedures can be fully parallel, to achieve scalable, extensible transformations.

## The New Way: Oracle9i (continued)

Moreover, the enhancements in Oracle9i are in general open. They are implemented in SQL, so that they can be widely used in all ETL environments.

- **Oracle9i OLAP Services:** Oracle9i is the new platform for analytical applications on the Internet. Oracle9i offers a scalable data platform, analytical functions, an Internet-ready online analytical processing (OLAP) system, and a complete development environment for analytical applications. As a result, you can quickly build applications with high analytical content, easily deploy these applications to large, geographically distributed user communities, and analyze larger data sets than previously possible. In addition, the Business Intelligence Beans provide OLAP-ready application building blocks to support the rapid development of Internet-based business intelligence applications.
- **Data Mining:** Oracle Personalization is Oracle's first product which moves the data mining algorithms into the database and adopting standards. Oracle Personalization uses the Transactional Naïve Bayes and Predictive Association Rules data mining algorithms, because they are algorithms that work well for real-time recommendations on the Web. All of the models built in Oracle Personalization are built using PL/SQL, and Oracle Personalization's API is Java-based.

Oracle Data Mining will continue to move additional data mining algorithms into the database, including decision trees, clustering techniques, and classic Naïve Bayes. Oracle will continue to package these data mining algorithms in thin-client GUIs that focus on solving business problems, and will continue to provide standards-based APIs for custom applications and integration.

Lastly, Oracle is actively adopting and driving open standards in the industry, including the JSR-73 Java data mining API standard. Oracle's implementation of data mining in the database using PL/SQL and Java APIs provides the most powerful and open data mining engine in the industry.

**Note:** Oracle9i's OLAP repository is fully compliant with the Common Warehouse Metamodel (CWM) standard, an Object Management Group (OMG) standard. This means that Oracle9i dimensions, cubes, measures/facts, and data source mappings are fully compliant with CWM, thus allowing Oracle9i to support the emerging CWM standard. This ensures that investments made in Oracle9i analytical application development are preserved as the industry moves toward the CWM standard.

# Full Extraction and Transportation Methods

- **Unload to flat files and use external tables**
- **Use transportable tablespaces:**
  - **From Oracle9i: Source and target databases can have different block sizes. Use the same procedures defined in Oracle8i to transport a tablespace to a database with a different block size.**
  - **This is especially useful for transporting from OLTP to warehouse.**
  - **There are the same limitations as in Oracle8i, except that different block sizes are allowed.**

ORACLE

## Transportable Tablespaces and Oracle9i

Oracle8i introduced an important mechanism for transporting data: Transportable tablespaces. This feature is the fastest mechanism for moving large volumes of data between two Oracle databases.

Previous to Oracle8i, the most scalable data transportation mechanisms relied on moving flat files containing raw data. These mechanisms required that data be unloaded or exported into files from the source database. Then, after transportation, these files were loaded or imported into the target database. Transportable tablespaces entirely bypass the unloading and reloading steps.

Using transportable tablespaces, Oracle data files (containing table data, indexes, and almost every other Oracle database object) can be directly transported from one database to another. Furthermore, like the Import and Export utilities, transportable tablespaces are a mechanism for transporting metadata in addition to data.

Transportable tablespaces have some notable limitations: Source and target systems must be running Oracle8i (or higher), must be running the same operating system, and must use the same character set and national character set. Despite these limitations, transportable tablespaces provide a valuable data transportation technique in many warehouse environments.

The most common applications of transportable tablespaces in data warehouses are in moving data from an OLTP database to a data warehouse. For this case, Oracle9i introduces a new feature which allows the database administrator to transport a tablespace with a certain block size to a database with another block size.

# Overview of Oracle Change Data Capture

- **When extracting data, you need to identify recent changes to the data efficiently.**
- **Oracle Change Data Capture facilitates incremental data extraction:**
  - **Captures DML operations performed on user tables**
  - **Changes are stored in database tables called change tables**
  - **Change data is available to users through views**
  - **Propagation as Publish and Subscribe mechanism**
  - **Uses internal triggers in the same database, which gives the user a synchronous view of changes**
  - **DBAs must ensure that Java is enabled in the database**

ORACLE

## Overview of Oracle Change Data Capture

An important consideration for extraction is incremental extraction, also called change data capture. If a data warehouse extracts data from an operational system on a nightly basis, then the data warehouse requires only the data that has changed since the last extraction (that is, the data that has been modified in the past 24 hours).

When it is possible to accurately identify and extract only the most recently changed data, the extraction process is much more efficient, because it must extract a much smaller volume of data. Unfortunately, in many source systems, identifying the recently modified data is difficult or intrusive to the operation of the system. Change data capture is typically the most challenging technical issue in data extraction.

Oracle Change Data Capture is a new server feature introduced in Oracle9i that quickly identifies and processes only the data that has changed, not entire tables, and makes the change data available for further use. It captures all the INSERT, UPDATE, and DELETE operations performed on user tables. These changes are stored in a new database object called a change table, and the change data is made available to applications in a controlled way using views.

Oracle Change Data Capture (CDC) can be used in synchronous mode only. This mode of data capture employs internal database triggers: Changes are stored in the same database as the one in which changes are tracked. This has a negative impact on performance, but has the advantage of giving the user a synchronous view of the changes.



## **Overview of Oracle Change Data Capture (continued)**

**Note:** Although the user interface is through PL/SQL packages, Oracle Change Data Capture is implemented in Java and C. DBAs must have the Java Runtime installed and enabled on the Oracle9i RDBMS server.

# The Publish and Subscribe Model

- **Publisher:**
  - **Determines and advances the change sets**
  - **Uses the DBMS\_LOGMNR\_CDC\_PUBLISH package supplied by Oracle**
  - **Publishes the change data**
  - **Allows controlled access to subscribers**
- **Subscriber:**
  - **Uses the DBMS\_LOGMNR\_CDC\_SUBSCRIBE package supplied by Oracle**
  - **Extends the window and the create change view**
  - **Prepares the subscriber views**
  - **Views data stored in change tables**
  - **Purges the subscriber views**
  - **Removes the subscriber views**

ORACLE

9-10

Copyright © Oracle Corporation, 2001. All rights reserved.

## The Publish and Subscribe Model

Most Change Data Capture systems have one publisher that captures and publishes change data for any number of Oracle source tables. There can be multiple subscribers accessing the change data. Change Data Capture provides PL/SQL packages to accomplish the publish and subscribe tasks.

### **Publisher**

The publisher (who is usually a database administrator) determines which tables the warehouse application captures changes from. These tables are referred to as source tables. For each source table in the OLTP system, the publisher creates a corresponding change table on the staging system. Change tables are organized into change sets, which keep track of the changes to a number of tables, in a transactional consistent manner. Oracle CDC ensures that none of the updates are missed, or counted twice.

The publisher performs these tasks:

- Determines the source tables from which the data warehouse application captures change data
- Uses the DBMS\_LOGMNR\_CDC\_PUBLISH package, supplied by Oracle, to set up the system to capture data from one or more relational tables (called source tables)
- Publishes the change data in the form of change tables
- Allow controlled access to subscribers by using the GRANT and REVOKE SQL statements to grant and revoke the SELECT privilege on change tables for users and roles

## The Publish and Subscribe Model (continued)

### Subscribers

After the change set has been created, client applications can subscribe to it. The subscribers are consumers of the published change data. Each subscriber has his or her own change view on the change data, so that multiple clients can subscribe to the same change set simultaneously without interfering with one another.

For example, if the change set contains all the changes that occurred between Monday and Friday, application A can be processing data from Tuesday, client B can be looking at the data from Wednesday and Thursday, and so on. Each client has its own subscription window, which contains a block of transactions in the order in which they were committed.

Oracle CDC manages the subscription window on behalf of each subscriber, by creating a database view that returns the range of transactions of interest to that subscriber. The subscriber accesses the change data by performing a `SELECT` operation on the change view that was generated by CDC. As each client finishes processing the data in its subscription window, it calls a procedure to purge the contents of the window. To read additional change data, it calls a procedure to extend the window, and CDC creates a new subscriber view.

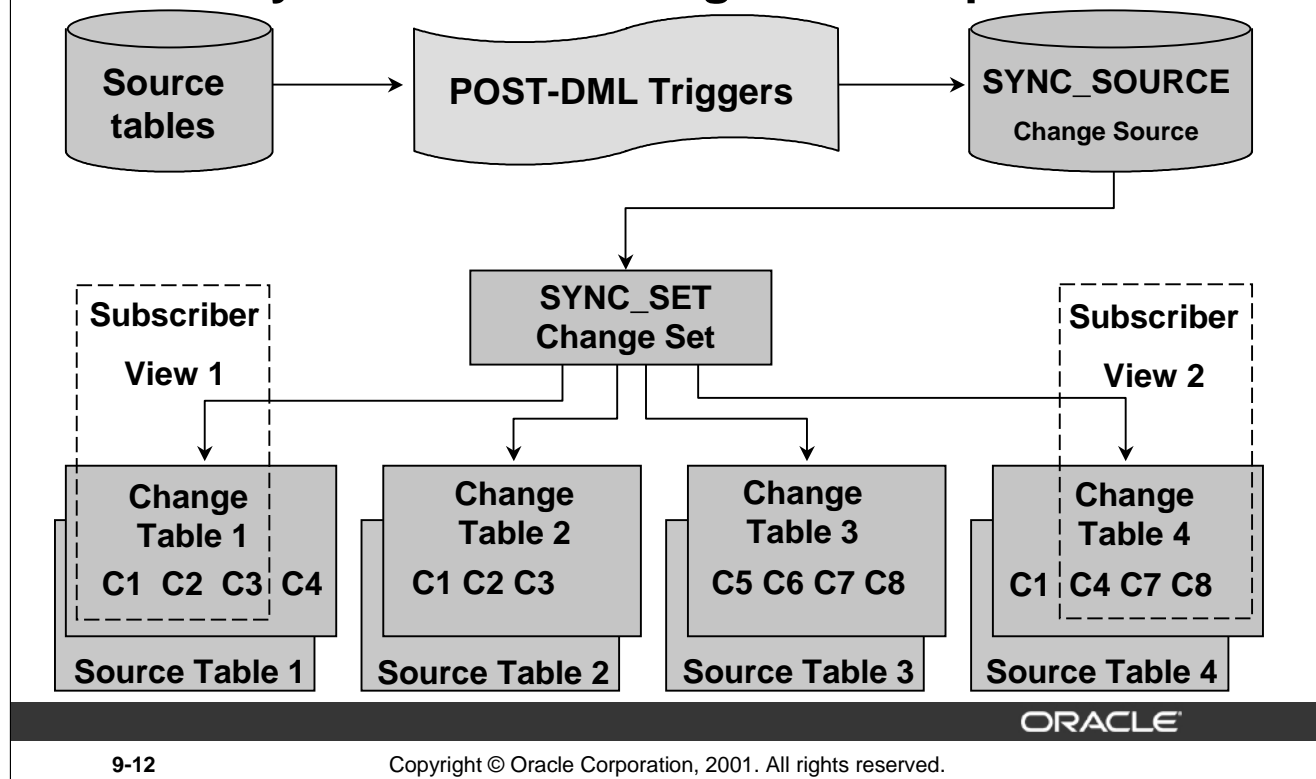
Subscribers go through the data at their own pace, while Oracle CDC takes care of storage management. Data that is no longer in use by any subscriber is automatically purged by CDC. This is necessary to prevent the change set from growing indefinitely.

Subscribers perform the following tasks:

- Use the `DBMS_LOGMNR_CDC_SUBSCRIBE` package, supplied by Oracle, to subscribe to source tables for controlled access to the published change data for analysis
- Extend the window and create a new view when the subscriber is ready to receive a set of change data
- Prepare the subscriber views
- View data stored in change tables through subscriber views, by using `SELECT` statements to retrieve change data
- Purge the subscriber view when it is finished processing a block of changes, effectively making the subscriber view empty
- Remove the subscriber views

**Note:** The following slides show examples of how to use CDC.

## Components and Terminology for Synchronous Change Data Capture



## Components and Terminology for Synchronous Change Data Capture

The following sections describe Change Data Capture components in more detail:

### Source System

A source system is a production database that contains source tables for which changes are captured.

### Source Table

A source table is a database table that resides on the source system, which contains the data you want to capture. Changes made to the source table are immediately reflected in the change table.

### Change Source

A change source represents a source system. There is only one system-generated change source, named SYNC\_SOURCE.

### Change Set

A change set is a collection of change tables. There is only one system-generated change set, named SYNC\_SET. Change tables are contained in the predefined SYNC\_SET change set.

## **Components and Terminology for Synchronous Change Data Capture (continued)**

### **Change Table**

A change table contains the change data resulting from DML statements made to a single source table. A change table consists of two things: The change data itself, which is stored in a database table, and the system metadata necessary to maintain the change table. A given change table can capture changes from only one source table. In addition to published columns, the change table contains control columns that are managed by Change Data Capture.

### **Subscriber View**

A subscriber view is a view created by Change Data Capture, which returns all of the rows in the subscription window. In the example on this slide, the subscribers have created two views: One on columns 1, 2 and 3 of Source Table 1, and one on columns 4, 7, and 8 of Source Table 4. The columns included in the view are based on the actual columns that the subscribers subscribed to in the source table.

### **Subscriber Window**

A subscriber window defines the time range of change rows that the subscriber can currently see. The oldest row in the window is the low-water mark; the newest row in the window is the high-water mark. Each subscriber has a subscription window.

**Note:** With synchronous data capture, internal triggers are used to generate the change data when data manipulation language (DML) operations are made to the database on the source system. Every time a DML operation occurs on a source table, the internal trigger is executed and writes a record of that operation to the change table.

## Data Dictionary Views Supporting CDC

- **CHANGE\_SOURCES** lists existing change sources
- **CHANGE\_SETS** lists existing change sets
- **CHANGE\_TABLES** lists existing change tables
- **DBA\_SOURCE\_TABLES** lists published source tables
- **DBA\_PUBLISHED\_COLUMNS** lists published source table columns
- **DBA\_SUBSCRIPTIONS** lists all registered subscriptions
- **DBA\_SUBSCRIBED\_TABLES** lists published tables to which subscribers have subscribed
- **DBA\_SUBSCRIBED\_COLUMNS** lists the columns of published tables to which subscribers have subscribed

ORACLE

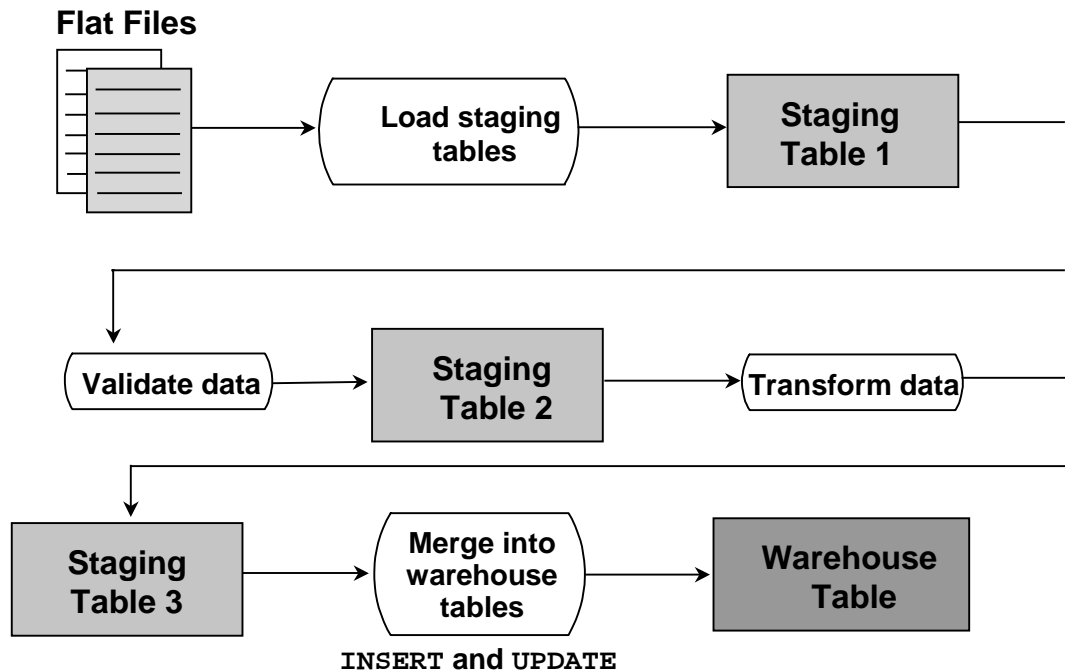
9-14

Copyright © Oracle Corporation, 2001. All rights reserved.

### Data Dictionary Views Supporting CDC

**Note:** For most of these views, there are also the corresponding ALL\_ and USER\_ views.

# Classical ETL Process



ORACLE

9-15

Copyright © Oracle Corporation, 2001. All rights reserved.

## Classical ETL Process

Data transformations are often the most complex and, in terms of processing time, the most costly part of the Extraction Transformation Loading (ETL) process. They range from simple data conversions to extremely complex data scrubbing techniques. Many, if not all, data transformations can occur within an Oracle9i database, although transformations are often implemented outside of the database (for example, on flat files) as well.

From an architectural perspective, you can transform your data in two ways:

- Multistage data transformation
- Pipelined data transformation

### Multistage Data Transformation

The data transformation logic for most data warehouses consists of multiple steps. For example, in transforming new records to be inserted into a sales table, there can be separate logical transformation steps to validate each dimension key. A graphical way of looking at the transformation logic is presented on the above slide.

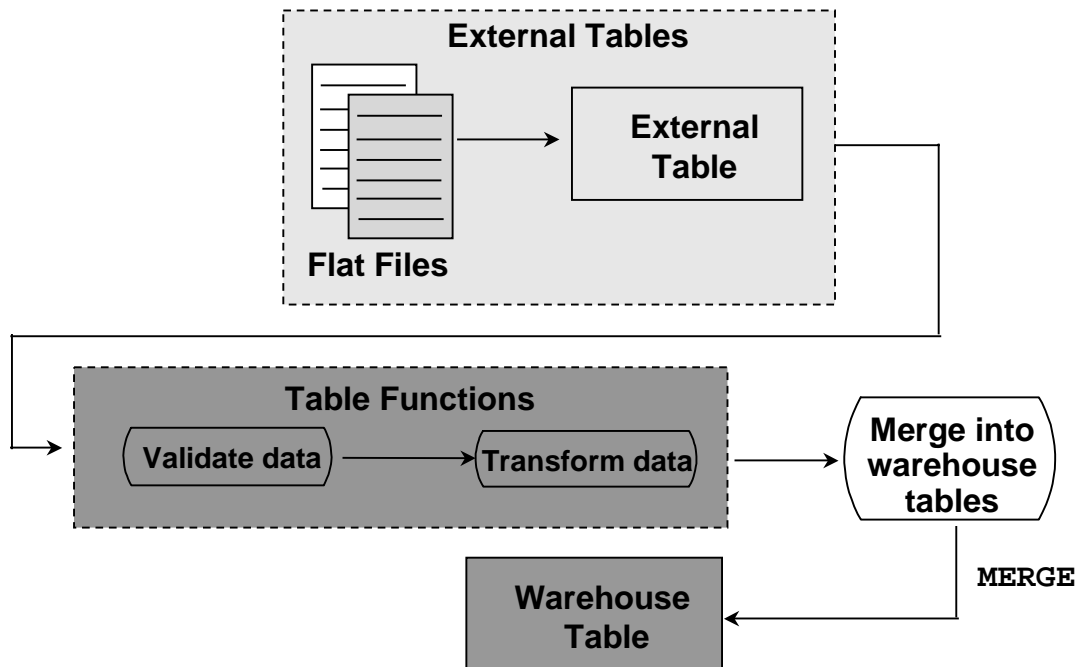
When using Oracle8i as a transformation engine, a common strategy is to implement each different transformation as a separate SQL operation, and to create a separate, temporary staging table to store the incremental results for each step. This load-then-transform strategy also provides a natural checkpointing scheme for the entire transformation process, which enables the process to be more easily monitored and restarted. However, a disadvantage of this multistage approach is that the space and time required increases.

**Classical ETL Process (continued)**

It may also be possible to combine many simple logical transformations into a single SQL statement or single PL/SQL procedure. Doing so can provide better performance than performing each step independently, but it may also introduce difficulties in modifying, adding, recovering, or dropping individual transformations.



## Pipelined Data Transformation in Oracle9i



ORACLE

9-17

Copyright © Oracle Corporation, 2001. All rights reserved.

### Pipelined Data Transformation in Oracle9i

Oracle9i significantly enhances the ability of the Oracle database to address tasks specific to ETL environments. The ETL process flow can be changed dramatically, and the database becomes an integral part of the ETL solution. Taking advantage of the new functionality, some of the formerly necessary process steps become obsolete, and others can be remodeled to make the data flow and the data transformation more scalable and less interruptive. Rather than transforming then loading serially (with most of the tasks done outside the database), or loading then transforming, the enhanced process transforms *while* loading.

It is important to understand that the database offers toolkit functionality rather than trying to provide a one-size-fits-all solution. The underlying database has to enable the ETL process flow for a specific customer need, and not dictate or constrain it from a technical perspective.

The above slide illustrates the new functionality, which is discussed later in this lesson.

# External Tables

- **External tables are read-only tables where the data is stored outside the database in flat files. They also describe how this data is presented to the database.**
- **The data can be queried like a virtual table, using any supported language inside the database.**
- **No DML operations are allowed and no indexes can be created.**
- **The metadata for an external table is created using a `CREATE TABLE` statement.**
- **Access rights are controlled through the `SELECT TABLE` and `READ DIRECTORY` privileges.**

ORACLE

9-18

Copyright © Oracle Corporation, 2001. All rights reserved.

## Overview of External Tables

External tables are like regular SQL tables, except that the data is read-only and does not reside in the database (thus the organization is “external”). The external table can be queried directly and in parallel using SQL. As a result, the external table acts like a view. The metadata for the external table is created using the `CREATE TABLE ... ORGANIZATION EXTERNAL` statement.

No DML operations are possible and no indexes can be created on these tables.

The `CREATE TABLE ... ORGANIZATION EXTERNAL` operation involves only the creation of metadata in the Oracle Dictionary, because the external data already exists outside the database. Once the metadata is created, the external table feature enables the user to easily perform parallel extraction of data from the specified external sources.

# Applications of External Tables

## External tables:

- **Allow external data to be queried and joined directly and in parallel, without being loaded into the database**
- **Eliminate the need for staging the data within the database for ETL in data warehousing applications**
- **Are useful in environments where an external source has to be joined with database objects and then transformed**
- **Are useful when the external data is large and not queried frequently**
- **Complement SQL\*Loader functionalities**

ORACLE

9-19

Copyright © Oracle Corporation, 2001. All rights reserved.

## Applications of External Tables

The external table feature in Oracle9i allows you to use external data as a “virtual table,” which can be queried and joined directly and in parallel without the external data first being loaded into the database.

External tables enable you to merge the transformation process with the loading process without any interruption of the data streaming. It is no longer necessary to stage the data inside the database for comparison or transformation.

The main difference between external tables and regular tables is that externally organized tables are read-only. No DML operations are possible, and no indexes can be created on them. Oracle9i's external tables are a complement to the existing SQL\*Loader functionality, and are especially useful for environments where the complete external source has to be joined with existing database objects and transformed in a complex manner, or where the external data volume is large and used only once.

SQL\*Loader, on the other hand, may still be the better choice for loading data in cases where additional indexing of the staging table is necessary. This is true for operations where the data is used in independent complex transformations, or where the data is only partially used in further processing.

## Example of Defining External Tables

```
CREATE table employees_ext (employee_id NUMBER,  
first_name CHAR(30), last_name CHAR(30))  
ORGANIZATION EXTERNAL (      -- External Table  
TYPE oracle_loader           -- Access Driver  
DEFAULT DIRECTORY delta_dir  -- Files Directory  
ACCESS PARAMETERS           -- Similar to SQL*Loader  
    (RECORDS DELIMITED BY NEWLINE  
     FIELDS TERMINATED BY ','  
     BADFILE 'bad_emp_ext'  
     LOGFILE 'log_emp_ext'  
     MISSING FIELDS ARE NULL)  
    LOCATION ('emp1.txt','emp2.txt'))  
PARALLEL 5  -- Independent from the number of files  
REJECT LIMIT UNLIMITED;
```

ORACLE

9-20

Copyright © Oracle Corporation, 2001. All rights reserved.

### Example of Defining External Tables

An external table can be created with single `CREATE TABLE` DDL command. This creates the meta-information which is necessary for accessing the external data seamlessly from inside the database.

The following information must be provided:

- Columns and data types for access in the database
- Where to find the external data
- Access driver: The access driver and the external table layer must perform the necessary transformations on the data in the data file, so that it matches the external table definition. There is one access driver for every implementation of an external table type.
- Format of the external data, similar to SQL\*Loader
- Degree of parallelism (note that the degree of parallelism is not dependent on the number of external data files)

In the example in the slide, an external table named `employees_ext` is defined. `delta_dir` is the directory where the external flat files reside. This example also use the default access driver for this implementation of external tables, which is called `oracle_loader`. The access parameters control the extraction of data from the flat file using record and file formatting information. The directory object was introduced in Oracle8i.

## Data Dictionary Information for External Tables

### DBA\_EXTERNAL\_TABLES

- OWNER
- NAME
- TYPE\_OWNER
- TYPE\_NAME
- DEFAULT\_DIRECTORY\_OWNER
- DEFAULT\_DIRECTORY\_NAME
- REJECT\_LIMIT

### DBA\_EXTERNAL\_LOCATIONS

- OWNER
- TABLE\_NAME
- LOCATION
- DIRECTORY\_OWNER
- DIRECTORY\_NAME

ORACLE

9-21

Copyright © Oracle Corporation, 2001. All rights reserved.

### Data Dictionary Information for External Tables

DBA\_EXTERNAL\_TABLES lists the specific attributes of all the external tables in the system:

- OWNER: Owner of the external table
- NAME: Name of the external table
- TYPE\_OWNER: Implementation type owner
- TYPE\_NAME: Implementation type name
- DEFAULT\_DIRECTORY\_OWNER: Owner of the default directory for this external table
- DEFAULT\_DIRECTORY\_NAME: Name of the default directory for this external table
- REJECT\_LIMIT: Reject limit

DBA\_EXTERNAL\_LOCATIONS lists the specific flat files and corresponding Oracle Directories:

- OWNER: Owner of the external table
- TABLE\_NAME: Name of the external table
- LOCATION: Flat file name
- DIRECTORY\_OWNER: Owner of the directory for this external table
- DIRECTORY\_NAME: Name of the directory for this external table

# Transformations with Pipelined Table Functions

- **Oracle9i supports pipelined and parallelizable table functions:**
  - Output is a set of rows
  - Input can be a set of rows
  - Output can be pipelined
  - Evaluation of the table function can be parallelized
- **Table functions are used in the FROM clause of a SELECT statement**
- **Table functions can be defined in PL/SQL using a native PL/SQL interface, or in Java or C using the Oracle Data Cartridge Interface (ODCI)**

ORACLE

9-22

Copyright © Oracle Corporation, 2001. All rights reserved.

## Overview of Table Functions

In the ETL process, the data extracted from a source system passes through a sequence of transformations before it is loaded into a data warehouse. A large class of user-defined transformations are implemented in a procedural manner, either outside the database or inside the database in PL/SQL. The table functions in Oracle9i provide support for pipelined and parallel execution of such transformations implemented in PL/SQL, C, or Java.

Using Table Functions avoid intermediate staging tables, which interrupt the data flow through the various transformation steps.

# Creating Table Functions

```
CREATE OR REPLACE FUNCTION transform(p r.ref_cur_type)
RETURN table_order_items_type
PIPELINED
PARALLEL_ENABLE( PARTITION p BY ANY) IS
BEGIN
    FOR rec IN p LOOP
        ... -- Transform this record
        PIPE ROW (rec);
    END LOOP;
    RETURN;
END;
```

ORACLE

9-23

Copyright © Oracle Corporation, 2001. All rights reserved.

## Example of Creating Table Functions

In the above example, REF\_CUR\_TYPE and TABLE\_ORDER\_ITEMS\_TYPE are user-defined object types. REF\_CUR\_TYPE is defined as a ref cursor. TABLE\_ORDER\_ITEMS\_TYPE is an object table. Pipelined functions must have return statements that do not return any values.

Here, the idea is to transform each record corresponding to the cursor passed as the parameter for the function, and return the corresponding row to the caller. Note that one record is returned as soon as one row is processed.

The new PIPELINED instruction in PL/SQL:

- Returns a single result row
- Suspends the execution of the function
- Restarts the function when the caller requests the next row

PARALLEL\_ENABLE is an optimization hint indicating that the function can be executed in parallel. The function should not use session states, such as package variables, because those variables may not be shared among the parallel execution servers.

The optional PARTITION *argument* BY clause is used only with functions that have a REF CURSOR argument type. It enables you to define the partitioning of the inputs to the function from the REF CURSOR argument. This affects the way the query is parallelized when the function is used as a table function (that is, in the FROM clause of the query). ANY indicates that the data can be partitioned randomly among the parallel execution servers. Alternatively, you can specify RANGE or HASH partitioning on a specified column list.

# Transformations Using Table Functions

```
SELECT * FROM TABLE  
(transform(cursor(select * from order_items_ext)))
```

```
INSERT /*+ APPEND, PARALLEL */  
INTO   order_items  
SELECT *  
FROM   TABLE(transform  
              (cursor  
                (select * from order_items_ext)  
              )  
            )
```

ORACLE

9-24

Copyright © Oracle Corporation, 2001. All rights reserved.

## Examples of Using Table Functions

Pipelined functions can be used in the FROM clause of SELECT statements. The result rows are retrieved iteratively from the table function implementation. Multiple invocations of a table function, either within the same query or in separate queries, result in multiple executions of the underlying implementation. In other words, there is no buffering and reuse of rows.



## **Advantages of PL/SQL Table Functions**

- **Table functions can reduce response time by pipelining the results to the consuming process as soon as they are produced.**
- **Table functions can return multiple rows during each invocation (pipelining of data).**
- **There are fewer invocations, and thus improved performance.**
- **Pipelining eliminates the need for buffering the produced rows.**

ORACLE

## Transformation Using Multitable INSERT Statements

- The **INSERT ... SELECT** statement can insert rows into multiple tables as part of a single DML statement.
- Multitable **INSERT** statements can be used in data warehousing systems to transfer data from one or more operational sources to a set of target tables.
- They can also be used internally for refreshing materialized views.
- You can still benefit from:
  - Parallelization
  - The direct-load mechanism

ORACLE

9-26

Copyright © Oracle Corporation, 2001. All rights reserved.

### Overview of Multitable INSERT Statements

The **INSERT ... SELECT** statement with the new syntax can be parallel, and can be used with the direct-load mechanism. The multitable **INSERT** statement inserts computed rows derived from the rows returned from the evaluation of a subquery. There are two forms of the multitable **INSERT** statement: Unconditional and conditional. In the unconditional form, an **INTO** clause list is executed once for each row returned by the subquery. In the conditional form, **INTO** clause lists are guarded by **WHEN** clauses that determine whether the corresponding **INTO** clause list is executed.

- An **INTO** clause list consists of one or more **INTO** clauses. Executing an **INTO** clause list inserts one row for each **INTO** clause in the list.
- An **INTO** clause specifies the target into which a computed row is inserted. The target specified can be any table expression that is legal for an **INSERT ... SELECT** statement. However, aliases cannot be used. The same table can be specified as the target for more than one **INTO** clause.
- An **INTO** clause also provides the value of the row to be inserted using a **VALUES** clause. An expression used in the **VALUES** clause can be any legal expression, but must refer only to columns returned by the **SELECT** list of the subquery. If the **VALUES** clause is omitted, the **SELECT** list of the subquery provides the values to be inserted. If a column list is given, each column in the list is assigned a corresponding value from the **VALUES** clause or the subquery. If no column list is given, the computed row must provide values for all columns in the target table.

## Overview of Multitable `INSERT` Statements (continued)

**Note:** The multitable `INSERT` statement can be performed with or without direct loading, and with or without parallelization for faster performance. In general, the rules are the same as for the single-table `INSERT` statement. The idea is that all corresponding tables are direct-loaded (or parallel), or none of them. That is why you need to specify only the `APPEND/PARALLEL` hint, without specifying tables names.

## **Advantages of Multitable INSERT Statements**

### **Multitable INSERT statements:**

- **Eliminate the need for multiple INSERT ... SELECT statements to populate multiple tables**
- **Eliminate the need for a procedure to do multiple insertions using the IF ... THEN syntax**
- **Significantly improve performance over above two methods, because they eliminate the cost of materialization and repeated scans on the source data**

ORACLE

## Unconditional INSERT

```
SQL> INSERT ALL
  2 INTO product_activity
  3 VALUES(today, product_id, quantity)
  4 INTO product_sales
  5 VALUES(today, product_id, total)
  6 SELECT trunc(order_date) today,
  7        product_id,
  8        SUM(unit_price) total,
  9        SUM(quantity) quantity
 10 FROM   orders o, order_items i
 11 WHERE  o.order_id = i.order_id
 12 AND    order_date = TRUNC(SYSDATE)
 13 GROUP BY product_id;
```

ORACLE

### The ALL INTO Clause in an Unconditional INSERT

To perform an unconditional multitable insert, specify ALL followed by multiple INTO clauses. Each INTO clause is executed once for each row returned by the subquery.

# Pivoting INSERT

```
SQL> INSERT ALL
  2 INTO sales VALUES (product_id,week,sales_sun)
  3 INTO sales VALUES (product_id,week,sales_mon)
  4 INTO sales VALUES (product_id,week,sales_tue)
  5 INTO sales VALUES (product_id,week,sales_wed)
  6 INTO sales VALUES (product_id,week,sales_thu)
  7 INTO sales VALUES (product_id,week,sales_fri)
  8 INTO sales VALUES (product_id,week,sales_sat)
  9 SELECT product_id, TO_DATE(week_id,'WW') week,
10        sales_sun,sales_mon,sales_tue,sales_wed,
11        sales_thu,sales_fri,sales_sat
12 FROM    sales_source_data;
```

ORACLE

9-30

Copyright © Oracle Corporation, 2001. All rights reserved.

## Example of a Pivoting INSERT

The above slide shows an example of inserting into the same table several times, pivoting from a nonnormalized form to a normalized form.

## Conditional ALL INSERT

```
SQL> INSERT ALL
  2  WHEN    product_id IN
  3        (SELECT product_id
  4          FROM    promotional_items)
  5  THEN    INTO promotional_sales
  6          VALUES(product_id,list_price)
  7  WHEN    order_mode = 'online'
  8  THEN    INTO web_orders
  9          VALUES(product_id, order_total)
 10  SELECT  product_id, list_price,
 11          order_total, order_mode
 12  FROM    orders;
```

ORACLE

9-31

Copyright © Oracle Corporation, 2001. All rights reserved.

### Syntax for Conditional ALL INSERT

The above example inserts a row into the `promotional_sales` table for any sold products that are on the promotional list, and into the `web_orders` table for products ordered online. It is possible that two rows are inserted for some item lines, and none for others.

## Conditional FIRST INSERT

```
SQL> INSERT FIRST
  2  WHEN order_total > 10000
  3  THEN INTO priority_handling VALUES(id)
  4  WHEN order_total > 5000
  5  THEN INTO special_handling  VALUES(id)
  6  WHEN order_total > 3000
  7  THEN INTO privilege_handling VALUES(id)
  8  ELSE INTO regular_handling  VALUES(id)
  9  SELECT order_total, order_id id
 10 FROM   orders ;
```

ORACLE

9-32

Copyright © Oracle Corporation, 2001. All rights reserved.

### Syntax for Conditional FIRST INSERT

The above statement inserts into the appropriate table according to the total of an order.

If you specify **FIRST**, as in the above example, the database evaluates each **WHEN** clause in the order in which it appears in the statement. For the first **WHEN** clause that evaluates to **True**, the database executes the corresponding **INTO** clause and skips subsequent **WHEN** clauses for the given row.

For a given row, if no **WHEN** clause evaluates to **True**, the following occurs:

- If you specified an **ELSE** clause, the database executes the **INTO** clause list associated with the **ELSE** clause.
- If you did not specify an **ELSE** clause, the database takes no action for that row.



# MERGE Statements

## **MERGE statements:**

- **Enable you to conditionally update or insert into the database. An `UPDATE` operation is performed if the row exists, and an `INSERT` operation is performed if it is a new row.**
- **Are sometimes called “upsert” statements**
- **Avoid multiple updates**

ORACLE

## Applications of MERGE Statements

- **MERGE statements use a single SQL statement to complete an UPDATE, or INSERT, or both.**
- **The statement can be parallelized transparently.**
- **Bulk DML can be used.**
- **Performance is improved because fewer statements require fewer scans of the source tables.**

ORACLE

## Example of Using the MERGE Statement in Data Warehousing

```
SQL> MERGE INTO customer c
  2  USING cust_src s
  3  ON (c.customer_id = s.src_customer_id)
  4  WHEN MATCHED
  5  THEN UPDATE
  6       SET c.cust_address = s.cust_address
  7  WHEN NOT MATCHED
  8  THEN INSERT(customer_id, ...)
  9       VALUES(src_customer_id, ...);
```

ORACLE

### Example of the MERGE Statement

This is an example of using MERGE in data warehousing. `customer` is a large fact table and `cust_src` is a smaller delta table, with rows which need to be inserted into `customer` conditionally. This MERGE statement indicates that the `customer` table has to be merged with the rows returned from the evaluation of the ON clause of the MERGE statement. The USING clause in this case is the `cust_src` table, but it can be an arbitrary query. Each row from `cust_src` is checked for a match to any row in `customer` by satisfying the join condition specified by the ON clause. If so, each row in `customer` is updated using the UPDATE SET clause of the MERGE statement. If no such row exists in `customer`, then the rows are inserted into table `customer` using the ELSE INSERT clause.

## Enhancements to Materialized Views

- **The `DBMS_MVIEW.EXPLAIN_MVIEW` procedure advises what actions are possible with a materialized view**
- **Materialized views can contain grouping sets**
- **Enhanced Summary Advisor includes**
  - **Graphical display of recommendations and their benefits**
  - **Support for multiple workloads, filters and new scripts**
- **Dimensions are no longer required by Summary Advisor**

ORACLE

9-36

Copyright © Oracle Corporation, 2001. All rights reserved.

### Enhancements to Materialized Views

The `DBMS_MVIEW.EXPLAIN_MVIEW` procedure advises what is possible with a materialized view, or potential materialized view, in terms of fast refreshes and query rewrites. If any operation is not possible, it will indicate why. You can use this information to help you reconfigure the materialized view to support the missing capability.

Grouping sets, discussed in the previous lesson, may be used in a materialized view definition.

The Enhanced Summary Advisor provides a graphical representation of recommendations gives user the ability to select materialized views from a graph based on the estimated benefits. The Summary Advisor recommends what materialized views should be created. There is a new procedure, `GENERATE_MVIEW_SCRIPT`, that produces a SQL script to implement a set of recommendations. Other changes to the Summary Advisor include:

- The ability to store multiple workloads
- Extract the workload from a user defined table or the SQL cache
- Filter the workload using a variety of different criteria
- Enhancements to the Summary Advisor engine with an improved interface to make it easier to use
- Removal of the restriction that dimensions must be present to use the Summary Advisor

## Enhancements to Materialized Views

- **The `DBMS_MVIEW.EXPLAIN_REWRITE` procedure shows when a query can be rewritten**
- **Rewrite is possible**
  - When the materialized view contains filtered data, such as `region_id > 10 AND region_id < 100` or `job_id IN (10,20,30)`
  - With inline views and self-joins
- **Partition Change Tracking in refresh and rewrite**
- **Fast refresh after DML or partition maintenance operations**

ORACLE

9-37

Copyright © Oracle Corporation, 2001. All rights reserved.

### Enhancements to Materialized Views (continued)

The `DBMS_MVIEW.EXPLAIN_REWRITE` procedure advises you when a query can be rewritten and, if not, why not. Using the results, you can take the appropriate action needed to make a query rewrite if at all possible.

Rewrite using filtered data is a major change in Oracle9i because you can now create small, compact materialized views servicing only a subset of the data. This eliminates the need to create large materialized views.

In Oracle9i, changes to partitions are tracked for efficient refresh and rewrite management using a feature often called Partition Change Tracking. If the materialized view is stale but the partition is fresh and, additionally, if rewrite optimization determines that it only needs data from the fresh partition, then a rewrite occurs. Likewise, a refresh keeps track of which partitions are stale. With this information, only the stale partitions are refreshed rather than the entire materialized view.

Fast refresh performance is improved and more materialized views are fast refreshable. The most significant change is the ability to fast refresh a materialized view containing joins and aggregates after a DML statement.

# Summary

**In this lesson, you should have learned how to:**

- **Use full extraction**
- **Use incremental extraction**
- **Transport data using flat files**
- **Transport data using transportable tablespaces**
- **Load data using pipelined transformations**
- **Access external tables**
- **Transform data with SQL**
- **Use new features of materialized views**

ORACLE