
Oracle9i Database: Product Overview

Student Guide

D12862GC10
Production 1.0
October 2001
D33985

ORACLE®

Authors

Stefan Lindblad
Lex De Haan
Jean-Francois Verrier

Technical Contributors and Reviewers

Lilian Hobbs, Larry
Baumann, Tamas Kerepes,
Joel Goodman, Christine Jeal,
Alexander
Hunold

Publisher

Sheryl Domingue

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.

Oracle and all references to Oracle Products 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

- Lesson Overview - Unit 1 Fundamentals I-2
- Lesson Overview - Unit 2 Optional Lessons I-3
- Lesson Overview - Unit 3 Appendixes I-4
- DBMS Application Types I-5

1 Introduction to the Relational Theory

- Objectives 1-3
- Domains and Relations 1-4
- Tables 1-5
- Missing Information 1-6
- Integrity Constraints 1-7

2 Data Access Languages

- Objectives 2-2
- SQL Language 2-3
- SQL:1999 Overview 2-4
- Oracle XML Developer's Kit 2-5
- PL/SQL Language 2-6
- Embedding SQL Statements 2-7
- Calls to OCI 2-8
- ODBC: Open Database Connectivity 2-9
- Oracle Objects for OLE 2-10
- External Procedures 2-11
- Java Programming Models 2-13
- Oracle JDBC Drivers 2-15
- Using the SQLJ Translator 2-16

3 Oracle9i Database Objects

Objectives 3-2

Oracle9i Type System 3-4

Large Objects (LOBs) 3-5

Collections 3-6

B*-Tree Indexes 3-7

Data Integrity Constraints and Triggers 3-8

Constraint Features 3-9

Database Triggers 3-10

Clusters 3-11

Bitmap Indexes 3-12

What Is a Bitmap Join Index? 3-13

Other Index Types 3-14

Views 3-15

Materialized Views 3-16

Index-Organized Tables 3-17

External Tables 3-18

Transformations with Pipelined Table Functions 3-19

Sequence Number Generation 3-20

Packages 3-21

Queues 3-22

4 Basic Oracle9i Architecture

Objectives 4-2

Instance and Database 4-3

Instance Components 4-4

Database Components 4-6

Shared Server 4-7

Database Structure 4-9

Data Dictionary 4-11

5 Oracle9i Real Application Clusters

Objectives 5-2

Modern Business Requirements 5-3

Real Application Clusters Overview 5-4

Real Application Clusters Characteristics 5-5

Background Processes 5-6

Benefits of Real Application Clusters 5-7

Global Cache Service 5-8

Dynamic Resource Remastering 5-9

Block Transfers 5-10

Benefits of Cache Fusion 5-11

High Availability Features 5-12

Real Application Clusters Guard 5-13

Real Application Clusters Guard Architecture 5-14

Monitors and Failover 5-15

Shared Initialization Parameter File 5-16

Shared Server-Side Parameter File 5-17

Real Application Clusters and OEM 5-18

OEM Instance Management Provisions 5-19

6 User and Data Security Management

Objectives 6-2

Privileges and Role-Based Security 6-4

Secure Application Role 6-5

Database Password Management 6-6

Virtual Private Database 6-8

Resource Management 6-9

Database Resource Manager 6-10

N-Tier Authentication 6-11

Oracle9i Enterprise User Management 6-12

Authentication Architecture 6-13

Database Auditing 6-15

Fine-Grained Auditing 6-16

Obfuscation Toolkit 6-17

Oracle Label Security 6-18

7 Oracle9i Optimizer

Objectives 7-2

Access Paths 7-3

EXPLAIN PLAN 7-5

Cached Execution Plans 7-6

SQL Trace and Tkprof 7-7

Rule-Based Optimizer 7-8

Cost-Based Optimizer 7-9

Statistics Management 7-10
Monitoring Table and Index Usage 7-11
Optimizer Plan Stability 7-12
Cursor Sharing 7-13

8 Transactions

Objectives 8-2
Data Concurrency Model 8-3
Potential Concurrency Problems 8-4
Undo Information 8-5
Resumable Space Allocation 8-6
Read Consistency 8-7
Oracle Flashback 8-8
Serializable Transactions 8-9
Transaction Management 8-10
Workspace Management for Long Transactions 8-11
Special Transaction Types 8-12
Distributed Databases 8-13
XA Architecture 8-14
Unit 1 Summary 8-16

9 Oracle9i Object-Relational Features

Objectives 9-3
What Is an Object? 9-4
Object Terminology 9-5

Relational and Object Tables 9-7

Object Views: An Evolutionary Approach 9-8

Extensible Type System 9-9

Extensible Cartridge Development Enabling Multimedia Applications 9-10

Object Type Translator 9-12

Oracle9i Client Side Object Cache 9-13

Type Inheritance 9-15

SQLJ Object Type 9-16

10 Globalization Support

Objectives 10-2

Globalization Support Features 10-3

Encoding Scheme Types 10-5

Database Character Sets and National Character Sets 10-6

Specifying Language-Dependent Behavior 10-7

Character Set Scanner 10-8

Oracle Locale Builder 10-9

11 Oracle9i Business Intelligence

Objectives 11-2

Partitioning 11-3

Range Partitioning 11-4

Hash Partitioning 11-5

Composite Partitioning 11-7

Partitioned Tables and Indexes	11-8
Parallel Execution	11-9
Materialized Views Management	11-10
Making Full Use of Materialized Views	11-11
Management of Materialized Views	11-12
Explain Materialized View	11-13
Explain Query Rewrite	11-14
Monitor Long-Running Operations	11-15
Transportable Tablespaces	11-16
Fast and Direct Data Loading	11-17
Common Extraction, Transformation, and Loading (ETL) Process	11-18
Pipelined Data Transformation in Oracle9i	11-19
Oracle9i OLAP Services	11-20
OLAP Application Platform	11-21

12 Backup and Recovery

Objectives	12-2
Data Export	12-3
Data Import	12-4
NOARCHIVELOG Mode: Offline Backups Only	12-5
Loss of a Data File	12-6
Restore	12-7
ARCHIVELOG Mode: Offline and Online Backups	12-8
Loss of a Data File	12-9
Partial Restore	12-10

Datafile Recovery 12-11
Trial Recovery 12-12
Recovery Manager (RMAN) 12-13
Block Media Recovery (BMR) 12-14
Oracle9i Data Guard Architecture 12-15
Analyzing Redo Log Files 12-16

13 Tools to Administer Oracle9i

Objectives 13-2
iSQL*Plus Architecture 13-3
Enterprise Manager Architecture 13-4
Job and Event System Features 13-6
SQL*Loader 13-8
Exporting Data 13-10
Importing Into a Database 13-11
Online Table Redefinition 13-12
Oracle SNMP Support 13-13
SNMP Support and Oracle Products 13-14
Unit 2 Summary 13-15

A Oracle 9i Network Services

B Oracle 9i Replication

C Content Management and Oracle 9i



Oracle9i Database: Product Overview

Lesson Overview - Unit 1 Fundamentals

- **Introduction to relational theory**
- **Data access languages**
- **Oracle9i database objects**
- **Basic Oracle9i architecture**
- **Real Application Clusters architecture**
- **User and data security management**
- **Oracle9i optimizer**
- **Transactions**

ORACLE

Lesson Overview - Unit 2 Optional Lessons

- **Oracle9i object-relational features**
- **Globalization support**
- **Oracle9i business intelligence**
- **Backup and recovery**
- **Tools to administer Oracle9i**

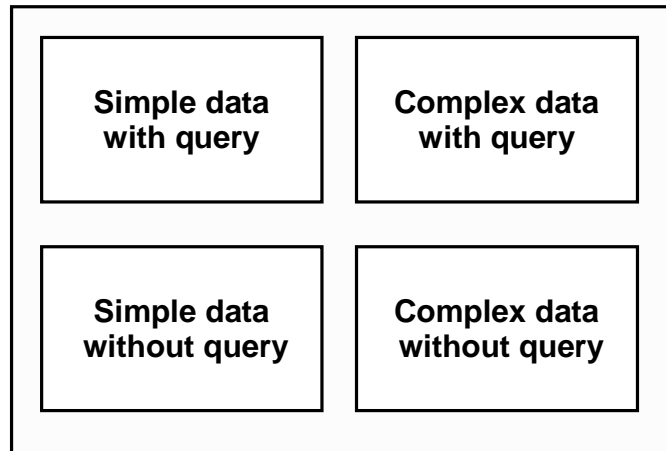
ORACLE

Lesson Overview - Unit 3 Appendixes

- **Networking**
- **Replication**
- **Content management for the Internet**

ORACLE

DBMS Application Types



ORACLE

I-5

Copyright © Oracle Corporation, 2001. All rights reserved.

DBMS Application Types

This slide shows an overview of database management systems (DBMSs) from both technical and marketplace perspectives. Each quadrant represents one of the four general types of DBMSs; there is no single DBMS that solves the requirements of all applications. Simple Data without Query: Text processing systems such as Word, NotePad. The perfect DBMS for these applications are File Systems.

Simple Data with Queries: Any SQL92 database supports this kind of application. Simple data because only built-in data types are supported. This market represents \$8 billion per year growing at a rate of 25% per year.

Complex Data without Queries: Object Oriented DBMSs are one example of such applications; they focus on providing tight integration with a programming language. This type of DBMSs is growing at nearly 50% per year but remains a market niche with about \$8 million per year.

Complex Data with Queries: Typically, a client requests a picture by content. DBMSs that support a dialect of SQL:1999, include non-traditional tools, and optimize for complex SQL:1999 queries are called object-relational DBMSs. They are relational because they support SQL; they are object-oriented because they support complex data. The object-relational market is expected to be 50% more than the relational market by year 2005 because of two market forces: new multimedia applications and business complex data applications.

Unit 1

Fundamentals

1

Introduction to the Relational Theory

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Domains**
- **Relations**
- **Tables**
- **Missing information**
- **Integrity**

ORACLE

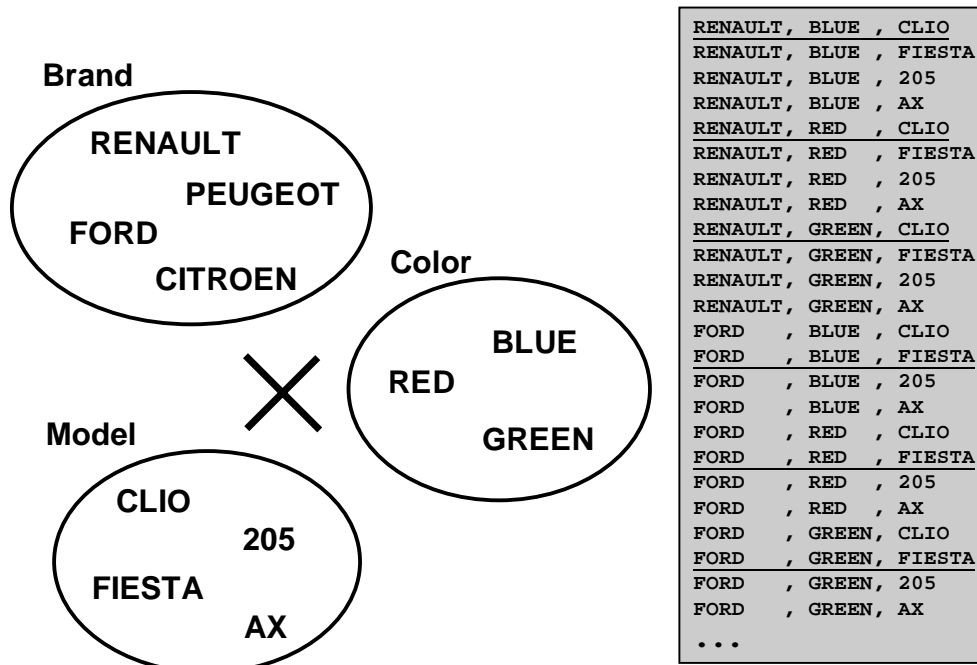
1-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

This first lesson introduces the basic concepts you need to know to understand the foundation of the relational theory.

Domains and Relations



ORACLE

1-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Domains and Relations

Initially, Oracle was primarily a database management system, with the main goal of storing and retrieving information.

The model used to store data is based on a mathematical theory called the relational model. Therefore, Oracle was a relational database management system (RDBMS).

This theory relies on the Set theory. Every set on the slide above is called a domain. Car brands can form a domain (for example, Renault, Peugeot, Ford). In the same way, you can have a domain of colors and a domain of models.

If you know that the different sets represent the real world of a company, you can relate them to each other. You start with the Cartesian product of the domains, as shown on the slide above, and you isolate the combinations that make sense. For example, there is no Peugeot Fiesta, because the Fiesta is a Ford model. Also, this model may not be available in all colors. This means you are defining a subset of the Cartesian product, called a relation.

Tables

CARS

RegNr	Brand	Color	Model	
5234FX13	RENAULT	BLUE	CLIO	...
6144RJ45	FORD	RED	FIESTA	...
...				

ORACLE

1-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Tables

A relation is a mathematical representation of a set of rows. This is called a *table* in an RDBMS. A table is the logical implementation of a relation. In a later lesson, you learn that inside Oracle, the physical implementation of a table is called a data segment.

A table has two dimensions, *rows* and *columns*. Each column in a table contains values that are a subset of a corresponding domain and each row in a table represents an element of the Cartesian product of all domains. The intersection of a column with a row is called a *field* or *column* value.

In reality, things are done in the opposite order in an Oracle9i database. You start by creating a table and then populate it with rows. Oracle does not implement the domain concept. When creating a table, you associate each column with one of the Oracle built-in data types, for example, NUMBER, VARCHAR, DATE, or with a user-defined data type.

Tables are the basis and central object of any RDBMS.

Using the ANSI/ISO standard SQL language, you are able to perform basic tasks, such as creating a table, selecting a subset of the rows, a subset of the columns, intersection, union, and join. The next lesson discusses SQL and other data access languages and interfaces.

Missing Information

- The relational model uses NULL values to represent missing information.
- NULL values imply a three-valued logic: conditions can be true, false, or unknown.
- You cannot represent the reason why information is missing.

ORACLE

1-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Missing Information

Each database management system needs a consistent way to represent missing information. The relational model uses NULL values for that purpose. A database language needs a consistent way to treat those NULL values, which implies a three-valued logic. For example, if a certain EMPLOYEE row does not have a SALARY value associated, you do not know whether it is higher or lower than a certain amount: logical statements or conditions can be true, false, or unknown.

Because the relational model only supports one way to represent missing information, you cannot distinguish between possible reasons why a certain value is missing. For example, a fax number can be missing because somebody does not have a fax at all, or just because you do not know the number, or even because you do not know whether somebody has a fax. In fact, some years ago Ted Codd, the father of the relational model, proposed a four-valued logic with two different NULL values: *applicable* and *inapplicable*. Because this would add complexity without resolving the problem, it will probably never be implemented.

The best approach is to try to avoid as much missing information as possible, in particular missing information of the *inapplicable* type, by modeling your database correctly.

Integrity Constraints

- **Application or database side implementation**
- **Column, row, table, database level**
- **Static and dynamic constraints**
- **Declarative versus procedural**

ORACLE

1-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Integrity Constraints

Because databases are supposed to model the real world, you need the ability to define rules. These rules are called integrity constraints. Of course, you can implement such rules in your application code, but you also need to be able to define them as an integral part of the logical database model.

There are three constraint levels:

- **Attribute or column level:** for example, salaries must be positive numbers, the hire-date is a mandatory attribute (you do not allow missing information)
- **Table level:** for example, each row needs a unique column or column combination that uniquely identifies that row (also called the primary key)
- **Database level:** for example, employees must be related with an existing department (also called referential integrity), or orders should have at least one associated order item

You also can distinguish between static and dynamic integrity constraints. Static integrity constraints describe valid database states, as in all of the examples above, and dynamic integrity constraints describe valid database state transitions. For example, only salary raises are allowed, or a new status of *divorced* is only allowed when the current status is *married*.

Integrity Constraints (continued)

You can implement constraints at the database level in two ways:

- Declarative, using features of the SQL language
- Procedural, writing database triggers using PL/SQL

Both ways of implementing constraints are discussed in the following lessons.

2

Data Access Languages

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

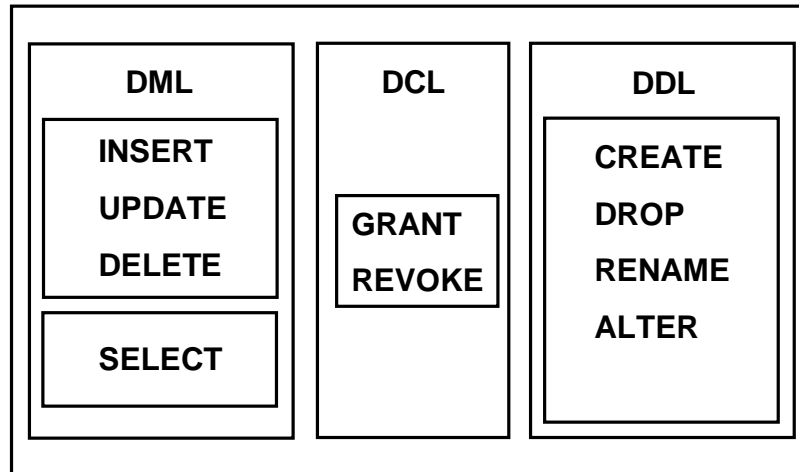
Objectives

After completing this lesson, you should be able to explain how to access Oracle9i using:

- SQL
- PL/SQL
- Embedded SQL
- OCI
- ODBC
- External procedures
- JDBC
- SQLJ
- Java stored procedures
- Enterprise JavaBeans
- CORBA

ORACLE

SQL Language



ORACLE

2-3

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL Language

Structured Query Language (SQL) is the language with which all programs and users access data in an Oracle database. Application programs and Oracle tools often allow users to access the database without using SQL directly, but these applications in turn must use SQL when executing user requests.

SQL lets you work with data at the logical level. You need not be concerned with the implementation details only when you want to manipulate the data. For example, to retrieve a set of rows from a table, you define a condition used to filter the rows. All rows satisfying the condition are retrieved in a single step and can be passed as a unit to the user, to another SQL statement, or to an application. You need not deal with the rows one by one, nor do you have to worry about how they are physically stored or retrieved. All SQL statements use the optimizer, a part that determines the most efficient means of accessing the specified data. Oracle9i also provides techniques you can use to enhance optimizer performance. SQL provides statements for a variety of tasks, including: querying data, inserting, updating, deleting rows in a table, creating, replacing, altering, and dropping objects, and controlling access to the database and its objects.

With Oracle9i, you can also perform multitable inserts and merge rows from one table into another. This means that you can either update an existing row or insert and merge rows.

SQL:1999 Overview

Oracle9i supports the new ISO/ANSI standards in SQL:1999, such as:

- **Full SQL:1999 join compliance**
- **Introduction of CASE expressions**
- **Introduction of scalar subqueries**
- **Support for explicit DEFAULT values**
- **The MERGE statement**
- **Multitable inserts**
- **Additional analytical functions and grouping sets**
- **Naming query blocks with the WITH clause**

ORACLE

2-4

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL:1999 Enhancements Overview

Oracle SQL complies with the latest industry-accepted standards such as the American National Standards Institute (ANSI) and the International Standards Organization (ISO). The latest SQL standard published is called SQL:1999. In this lesson the phrase “SQL:1999” refers to the SQL:1999 notations of both ANSI and ISO/IEC standards, officially known as “ISO/IEC 9075-1:1999.”

SQL:1999 syntax compliance is important for the following reasons:

- Provides easier migration of third party applications without the need to modify existing SQL code
- Provides ANSI/ISO standard functionality within the Oracle9i database
- Provides an easier learning curve when moving from other database products to Oracle9i

Note: For more details about Oracle and the ANSI/ISO SQL standard, please refer to one of the appendixes of the Oracle9i SQL Reference.

Oracle XML Developer's Kit

- **Standards-based libraries and utilities to generate, manipulate, render, and store XML-formatted data**
- **Includes:**
 - XML Parser
 - XSL Processor
 - XML Class Generator
 - XML Transviewer JavaBeans
 - XSQL Servlet
- **Available for Java, JavaBeans, C, C++, and PL/SQL**

ORACLE

2-5

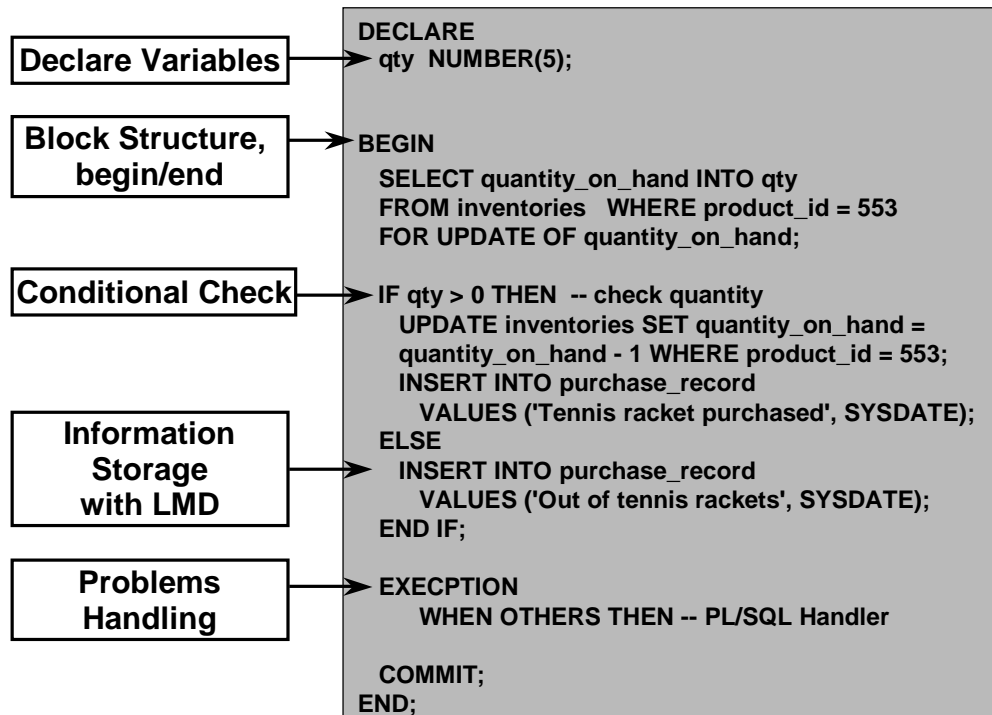
Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle and XML

To work with XML data types, you will use the XML Developers Kit. This kit contains several different components:

- XML Parser enables programmatic access to XML documents.
- XSL Processor transforms one XML document to another XML or text document, such as HTML.
- XML Class Generator generates class files so you can add XML document creation to your application.
- XML Transviewer JavaBeans can help with viewing the content of a XML document.
- XSQL Servlet produces dynamic XML documents based upon one or more SQL queries.

PL/SQL Language



ORACLE

2-6

Copyright © Oracle Corporation, 2001. All rights reserved.

PL/SQL Language

A good way to get acquainted with PL/SQL is to look at a sample program. This example program processes an order for tennis rackets. First, it declares a variable of type `NUMBER` to store the quantity of tennis rackets on hand. Then, it retrieves the quantity on hand from a database table named `inventory`. If the quantity is greater than zero, the program updates the table and inserts a purchase record into another table named `purchase_record`. Otherwise, the program inserts an out-of-stock record into the `purchase_record` table.

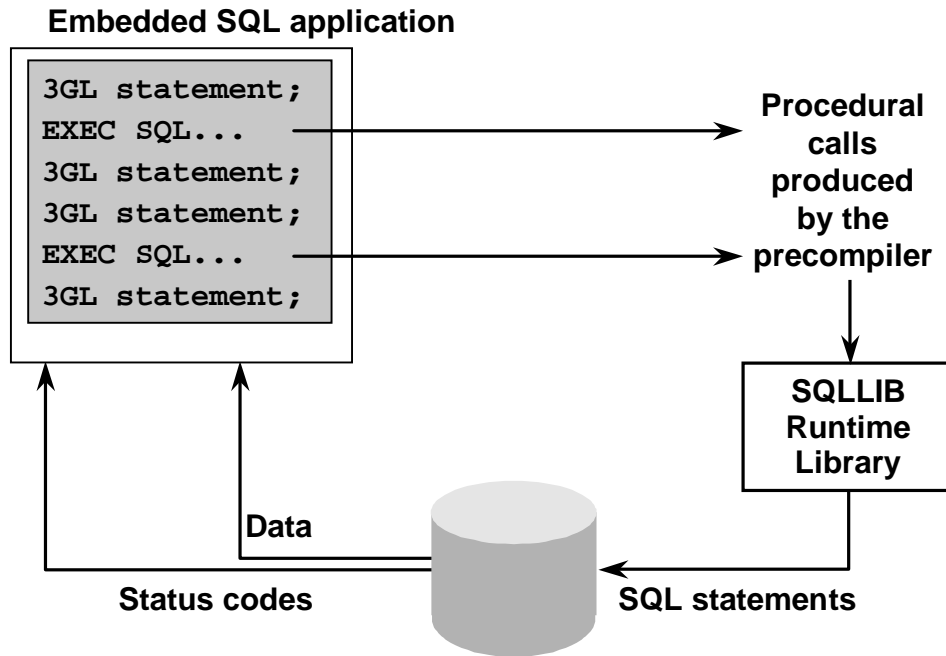
With PL/SQL, you can use SQL statements to manipulate Oracle data and flow-of-control statements to process the data. Moreover, you can declare constants and variables, define procedures and functions, and trap runtime errors. Thus, PL/SQL combines the data manipulating power of SQL with the data processing power of procedural languages.

PL/SQL is a block-structured language. That is, the basic units (procedures, functions, and anonymous blocks) that make up a PL/SQL program are logical blocks, which can contain any number of nested sub-blocks. Typically, each logical block corresponds to a problem or subproblem to be solved. Thus, PL/SQL supports the divide-and-conquer approach to problem solving called *stepwise refinement*.

A block (or sub-block) lets you group logically-related declarations and statements. That way, you can place declarations close to where they are used. The declarations are local to the block and cease to exist when the block completes.

In Oracle9i, PL/SQL can be native compiled, giving better performance for large applications.

Embedding SQL Statements



ORACLE

2-7

Copyright © Oracle Corporation, 2001. All rights reserved.

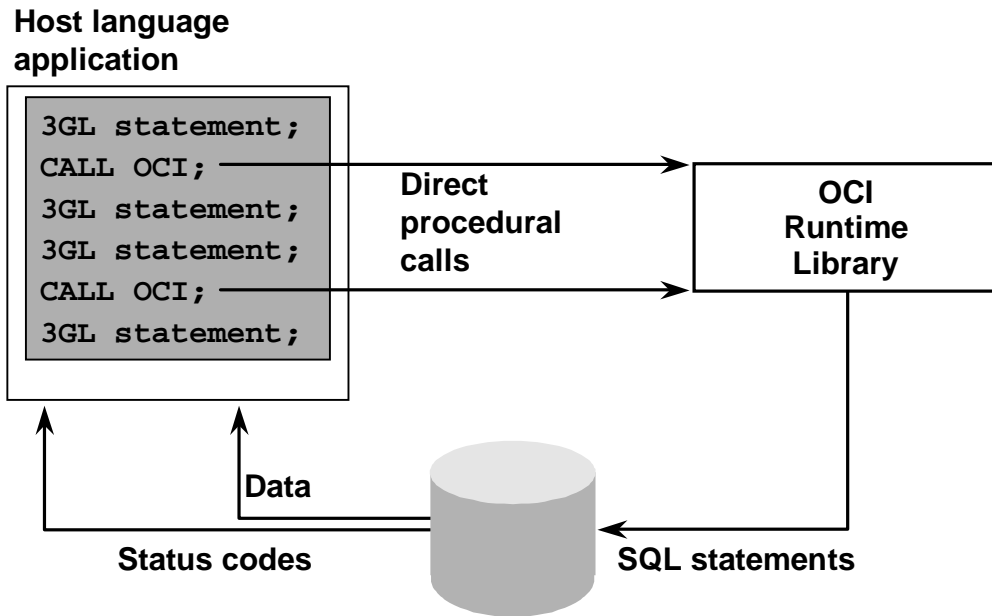
Embedding SQL Statements

Using the Oracle precompiler you can embed SQL statements in a high-level host program. The precompiler accepts the host program, translates the embedded SQL statements into various host language statements and into standard Oracle runtime library (SQLLIB) calls. It generates a source program that you can compile, link, and execute in the usual way. Oracle Corporation provides precompilers for C/C++ and COBOL.

Developing an embedded SQL application requires the following steps:

1. Editor creates the host program. The host program contains the embedded SQL statements. Write and save the host program using a system editor or word processor.
2. Oracle precompiler creates the source program. The source program contains the library calls which replace the SQL statements. Send the host program through the precompiler to produce source code. Debug any precompile-time errors. Error messages have the prefix PCC-.
3. Compiler creates the object program. Send the source program through the compiler to produce object code. Debug any compile-time errors.
4. Linker creates the executable program. The Linker calls the Oracle SQL Runtime Library to resolve the library calls. Link the object code with SQLLIB, the Oracle library of routines that process SQL statements, to produce an executable program. Run the program. Debug any runtime errors.

Calls to OCI



2-8

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

Calls to OCI

The Oracle Call Interface (OCI) is an application programming interface (API) that allows you to create applications that use the native procedures or function calls of a third-generation language to access an Oracle database server and control all phases of SQL statement execution. OCI supports the data types, calling conventions, syntax, and semantics of a number of third-generation languages, including C, C++, COBOL and FORTRAN.

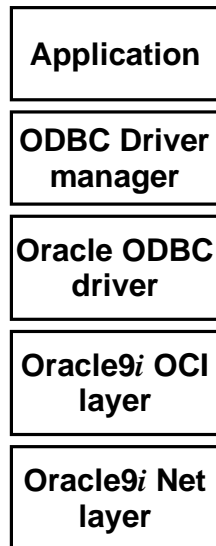
Using Oracle Call Interface (OCI), you can make procedural calls to OCI library functions directly from the host language source code. You compile and link the program in the same way that you compile and link a nondatabase application. There is no need for a separate precompilation step.

OCI provides significant advantages over other methods of accessing an Oracle database:

- More fine-grained control over all aspects of the application design
- Use of familiar 3GL programming techniques and application development tools such as browsers and debuggers
- Ability to associate a commit request with an execute to reduce roundtrips
- Optimization for queries using transparent pre-fetch buffers to reduce roundtrips
- Thread safety so you do not have to use mutually exclusive locks (mutex) on OCI handles
- Scrollable cursors so you can go anywhere in an open cursor

ODBC: Open Database Connectivity

- **Common neutral database API**
- **Write to one API set and access almost any SQL database**



ORACLE

2-9

Copyright © Oracle Corporation, 2001. All rights reserved.

ODBC: Open Database Connectivity

ODBC defines a vendor-independent application programming interface as defined by Microsoft, for accessing data stored in relational and non-relational databases using SQL as a standard for accessing data. The ODBC interface permits maximum interoperability where a single application can access different database management systems. This allows an application developer to develop, compile, and ship an application without targeting a specific DBMS. Users can then add modules called database drivers that link the application to their choice of database management systems.

The application layer is the ODBC enabled application. Some of the common applications are: Access, Excel, Visual Basic, Crystal Reports, Microsoft Foundation Classes.

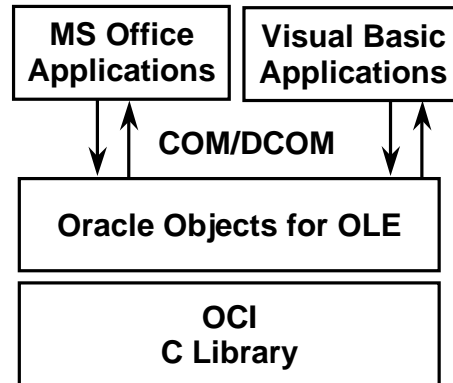
The ODBC Driver Manager Layer consists of the ODBC Driver Manager and ODBC Administrator that is provided by Microsoft. The ODBC Driver Manager is an application that loads and unloads the driver, and translates all the ODBC driver information and calls. The ODBC Administrator program is used to configure the data source name (DSN) and the connection string or Net alias into the ODBC Administrator for the ODBC connection. The ODBC driver is a set of DLLs that is used to translate from an ODBC enabled application to OCI.

The Oracle Call Interface (OCI) layer provides another set of DLLs that translate the calls from the ODBC driver into calls the database will understand.

The Oracle9i Net application layers provide the transportation of the data from the client machine to the Oracle database.

Oracle Objects for OLE

- **Collection of ActiveX objects and controls**
 - Data control
 - C++ classes
 - OLE automation objects
- **Full support for COM/DCOM**
- **Objects model similar to DAO.**
Examples: database, recordset, tabledef



ORACLE

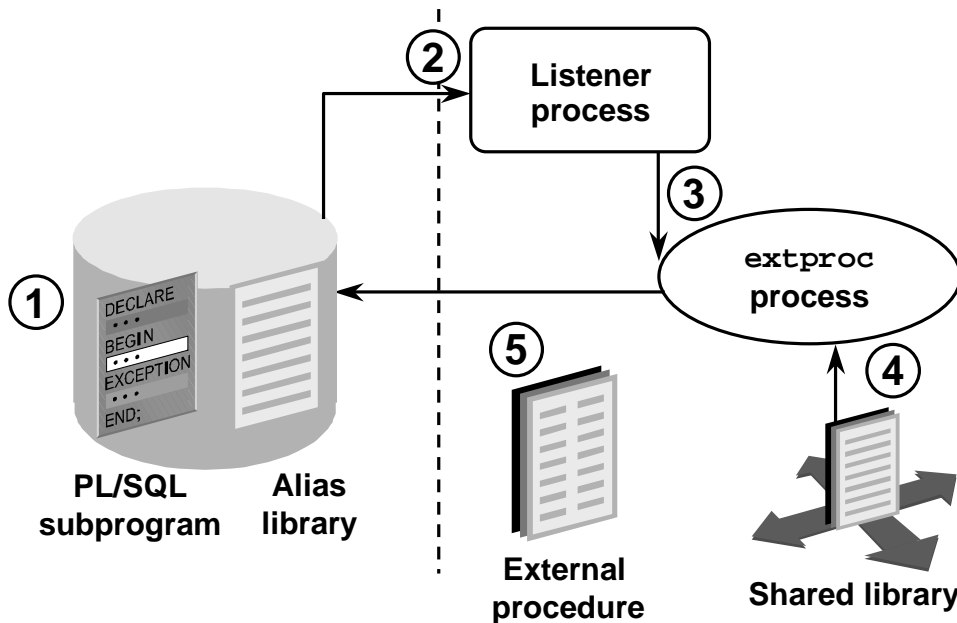
2-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Objects for OLE

Oracle Objects for OLE (OO4O) is an easy-to-use, Microsoft OLE/COM-based application programming interface (API) which integrates the Oracle server's data and functionality with Microsoft Windows clients and applications. Oracle Objects for OLE provides a simple and powerful way for Visual Basic, C++, and OLE 2.0 scripting-enabled applications to access Oracle9i data. For Visual Basic or other 4GL development, Oracle Objects for OLE provides a single custom control (VBX) combined with an OLE in-process server to allow users to plug native Oracle9i functionality into Windows applications. At the same time, C++ developers can use Class Libraries to access Oracle9i. Users wanting to access Oracle9i directly from Excel spreadsheets can do so using Objects for OLE's support for the Visual Basic for Applications Macro language. Oracle Objects for OLE is the best programmatic interface for desktop application developers familiar with the OLE/COM object model and who are seeking an easy-to-use API to integrate an Oracle database with their Windows applications. Oracle Objects for OLE offers these developers such benefits as ease of use through drag-and-drop access to an Oracle database from Visual Basic applications; excellent performance on low-end hardware; tight integration with the Oracle server, including access to PL/SQL stored procedures; and the ability to simultaneously access multiple databases from a single client session.

External Procedures



ORACLE

2-11

Copyright © Oracle Corporation, 2001. All rights reserved.

External Procedures

An external procedure is a 3GL routine that can perform special purpose processing. You call the external procedure from within PL/SQL or SQL. This is accomplished through a library schema object, which is called from PL/SQL, that contains the compiled library filename that is stored on the operating system.

Callouts and callbacks can also be referred to as an external procedure.

The interactions between the operating system and the database are designed to provide safe callouts. Although it is possible to create callouts that could damage the instance or database, the external procedure feature is not intended to be used in that manner.

The external procedure call enables you to invoke an external procedure using a PL/SQL program unit. Additionally, you can integrate the powerful programming features of 3GLs with the ease of data access of SQL and PL/SQL commands.

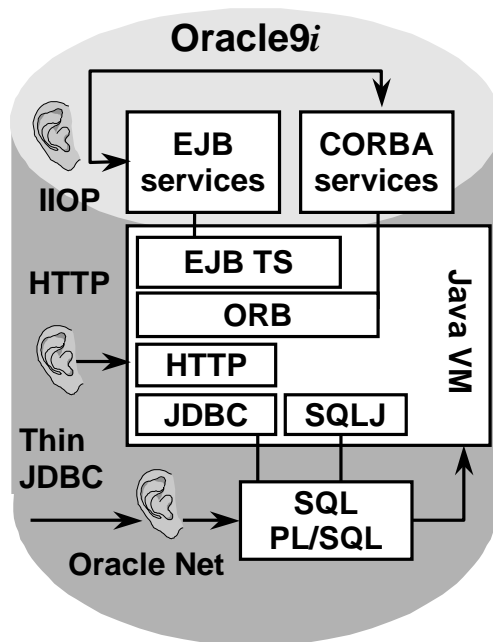
The external procedure can be set up to use a specific extproc process. This extproc process can be on a different machine from the database.

Extensibility allows you to extend the database and to provide backward compatibility. For example, you can invoke different index or sorting mechanisms as an external procedure, or call data cartridges.

Executing External Procedures

1. The server process executes a PL/SQL subprogram, which looks up an alias library.
2. The PL/SQL subprogram passes the request to the listener.
3. The listener process spawns the `extproc` process.
The `extproc` process remains active throughout your Oracle session until you log off.
4. The `extproc` process loads the shared library.
5. The `extproc` process executes the external procedure and returns results to the PL/SQL subprogram.

Java Programming Models



- **2 ways to access SQL:**
 - JDBC
 - SQLJ
- **3 protocols:**
 - Oracle Net, HTTP, IIOP
- **4 programming models:**
 - Java stored procedures
 - Enterprise JavaBeans
 - CORBA
 - Java Server Pages

ORACLE

2-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Java Programming Models

Oracle offers a comprehensive Java application with both products that you can use to build database applications in Java, as well as a server platform on which to run these applications.

Oracle offers three products to build database applications in Java:

- Standards compliant JDBC Drivers to access Oracle from Java applications
- SQLJ: a precompiler for embedded SQL in Java applications. Oracle's JDeveloper integrates both the JDBC Drivers and the SQLJ translator and provides a complete development environment with a number of features supporting the server
- Oracle has also built a scaleable Java Virtual Machine in the Oracle database which provides a very efficient platform on which these applications can be run.

This will allow a user to write Java stored procedures, functions, and triggers in Java. It will also serve as an Enterprise JavaBean Transaction Server and support a variety of CORBA clients.

To increase performance, all Java created in the database can be native compiled, meaning that java is compiled for that specific platform.

Oracle also provides an Application Server platform which also executes Java application programs.

The applications written in Java with any of these products can be deployed either in a two-tier client server environment or in multitier configurations.

In addition to supporting the EJB programming model, Oracle9i also serves as a CORBA client and server. It integrates a CORBA 2.0 compliant Object Request Broker or ORB. This allows application programs to call into the server and access Java stored programs and Enterprise JavaBeans using the IIOP protocol. In addition, Java applications executing on the Java VM can call out from the database using IIOP. In the first instance, the database behaves as a CORBA server; in the second, as a CORBA client. The ORB implementation in Oracle9i is highly scaleable, leveraging the Oracle Shared Server architecture for scalability, that is, in the way sessions are created and serviced in the database.

Further, all the advances that Oracle9i provided to handle 10,000+ Net connections, that is, internal connection pooling and Oracle9i Net Connection Manager, now work with IIOP.

Enterprise JavaBeans and Java stored procedures in the database are accessed through CORBA/IIOP in addition to Net. A user writes a Java stored program and can *publish* it to two different mechanisms. Firstly, to be accessed as a stored procedure through Net, it needs to be published or hooked up to SQL. Secondly, to be accessed as a CORBA server, it can be published to the ORB and accessed through the CORBA IIOP.

In addition to supporting CORBA clients, the database supports D/COM clients, such as Internet Explorer or Microsoft's Transaction Server through a D/COM to CORBA half-bridge.

With Oracle9i, you will have a number of different choices for using Web servers with the database:

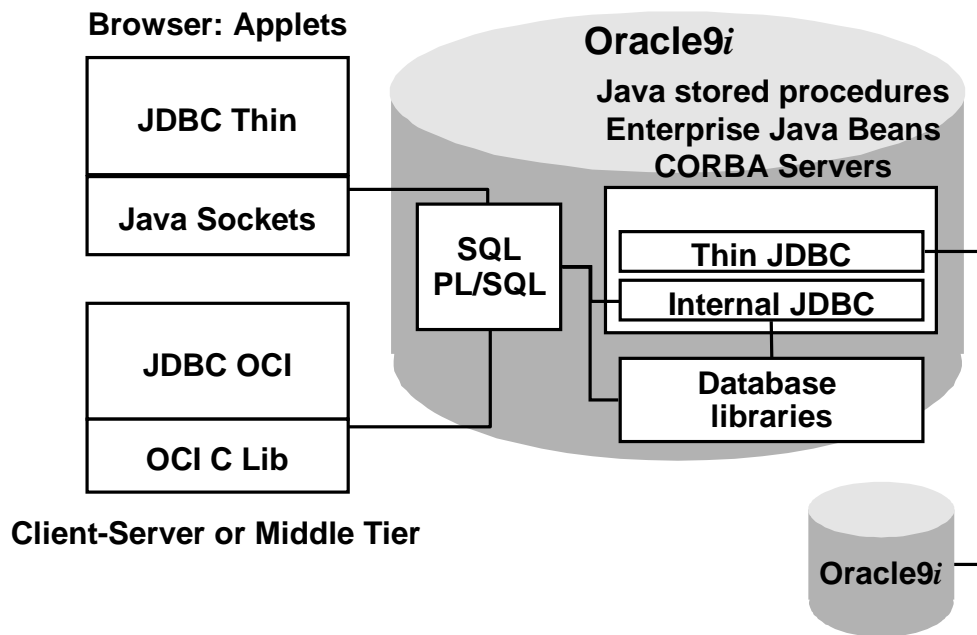
- The Oracle9i Net listener, dispatcher framework has been extended with Oracle9i to provide native support in the server for the HTTP protocol. In this case, there is no separate Web server process running or separate HTTP listener. The database server simply listens on a port to HTTP requests.
- Alternatively, you can also run a Web server and the database server as separate processes on the same machine. This is required if you are using the thin JDBC driver with a JDK 1.0.2 Java VM. It is not recommended however, since the Web server will typically make your database server I/O bound if it is run on the same box.
- Third, you can run the Web server and database server on different machines. This will provide you the best scalability and is suited when you use thin JDBC with JDK 1.2 signed applets.

A Java Server Page is simply a mechanism to generate dynamic Web pages by embedding Java tags within an HTML page.

Oracle supports Stored Java objects in the following cases:

- Java classes as database library units. These could be optionally exposed as EJB or CORBA objects.
- Java Stored Procedures callable from SQL and JDBC/SQLJ
- Methods of user defined Oracle9i objects

Oracle JDBC Drivers



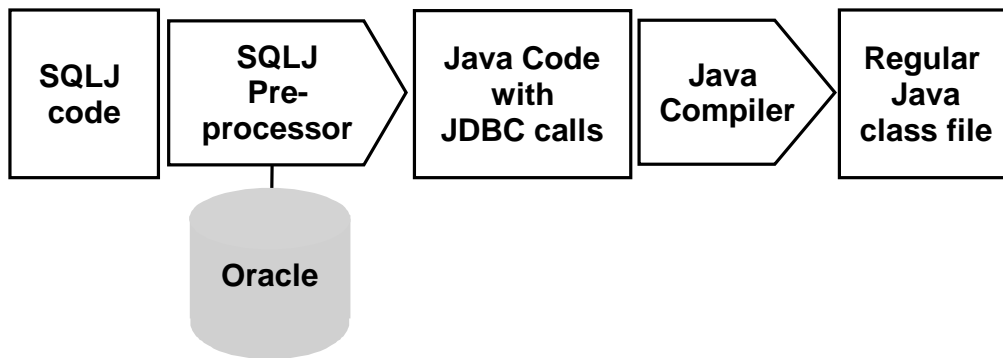
Oracle JDBC Drivers

The above diagram shows the four JDBC drivers developed at Oracle, each for a different deployment architecture.

- The JDBC Thin driver (Type 4) can be used in applets that execute inside a Web browser, such as Netscape Communicator or MS Internet Explorer; that is, in a thin client. This driver is built on Java sockets that directly communicate to the RDBMS engine, and it requires no client installation.
- The JDBC OCI, also called the Type 2 Driver, is built on top of Oracle's call interface, namely, OCI. This driver communicates through Oracle Net to the RDBMS engine, and requires that the OCI library be installed on the client.
- The server-side Internal JDBC (KPRB) driver is the version used in Java Stored procedures. It is built on top of Oracle's KPRB library, the equivalent of OCI on the server side. This driver is also significant, because this is the first time that the KPRB library has been exposed. It was not exposed earlier, because C was an *unsafe* language, which could cause the server to crash. But that danger is eliminated with Java, which is a safe language, just like PL/SQL.
- The Oracle JDBC server-side Thin driver offers the same functionality as the client-side Thin driver, but runs inside an Oracle database and accesses a remote database.

These three drivers together allow the Java application programmer true portability for the first time ever.

Using the SQLJ Translator



- **Pre-compiler generates standard Java source code with JDBC calls**
- **Checks SQL statements against the database**
- **Generated code compiles and runs like other Java programs**

ORACLE

2-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Using the SQLJ Translator

SQLJ allows applications programmers to embed static SQL operations in Java code in a way that is compatible with the Java design philosophy. A SQLJ program is a Java program containing embedded static SQL statements that comply with the ANSI-standard SQLJ Language Reference syntax; static SQL operations are predefined. The operations themselves do not change in real-time as a user runs the application, although the data values transmitted can change dynamically. Typical applications contain much more static SQL than dynamic SQL; dynamic SQL operations are not predefined. The operations themselves can change in real time and require direct use of JDBC statements. Note, however, that you can use SQLJ statements and JDBC statements in the same program.

SQLJ consists of both a translator and a run-time component and is smoothly integrated into your development environment. The translator is run by the developer, with translation, compilation, and customization taking place transparently. Translation replaces embedded SQL with calls to the SQLJ runtime, which implements the SQL operations. In standard SQLJ, this is typically, but not necessarily, done through calls to a JDBC driver. In the case of an Oracle database, you would typically use an Oracle JDBC driver. When the end user runs the SQLJ application, the run time is invoked to handle the SQL operations in real time. The Oracle SQLJ translator is conceptually similar to other Oracle precompilers and allows the developer to check SQL syntax, compare SQL operations to what is available in the schema, and check the compatibility of Java types with corresponding database types. In this way, errors can be caught by the developer instead of by a user at run time.

3

Oracle9i Database Objects

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

After completing this lesson, you should be able to describe the following Oracle9i database concepts:

- **Data types**
- **B*-tree indexes**
- **Data integrity and constraints**
- **Database triggers**
- **Clusters**
- **Additional index types**
- **Views and Materialized Views**
- **Index-organized tables**

ORACLE

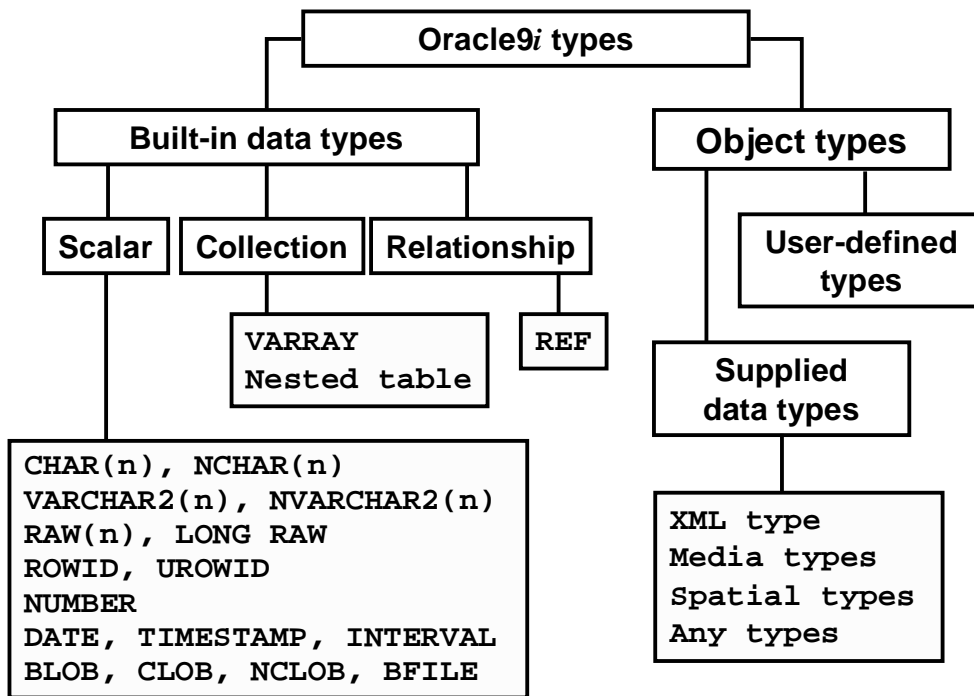
Objectives

After completing this lesson, you should be able to describe the following Oracle9i database objects:

- **External Tables**
- **Table functions**
- **Sequences**
- **Packages**
- **Queues**

ORACLE

Oracle9i Type System



ORACLE

3-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i Type System

The slide shows the complete list of all data types provided by Oracle9i.

These data types are SQL-based interfaces, called Oracle-supplied data types, for defining new types when the built-in or ANSI-supported types are not sufficient.

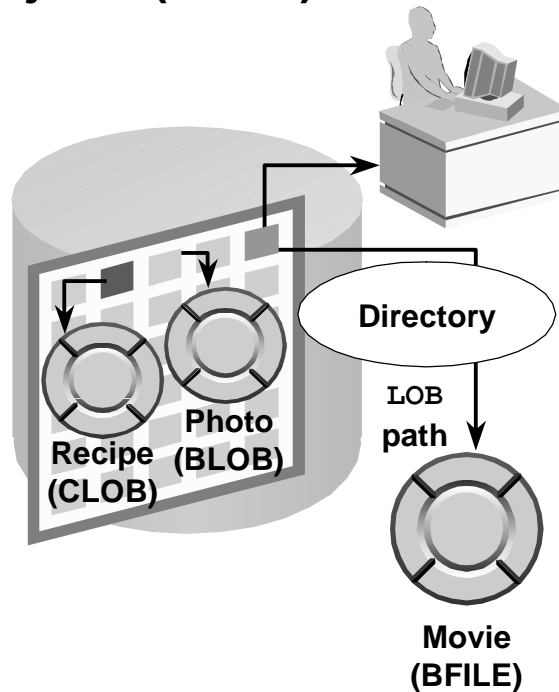
The Oracle server automatically provides the low-level infrastructure services needed for input-output, heterogeneous client-side access for new data types, and optimizations for data transfers between the application and the database.

For example the *Any* types provide highly flexible modeling of procedure parameters and table columns where the actual type is not known. These data types enable you to dynamically encapsulate and access type descriptions, data instances, and sets of data instances of any other SQL type. These types have OCI and PL/SQL interfaces for construction and access.

Another Oracle-supplied type is XML type, which provides a single CLOB storage option. This type can be used to store and query XML data in the database. It has member functions you can use to access, extract, and query the XML data using XPath expressions.

Large Objects (LOBs)

- **LOB storage:**
 - Unstructured data
 - Binary or character
 - Size to 4 GB
 - Special concurrency
- **Storage method:**
 - Internal: database
 - External: OS files



ORACLE

3-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Large Objects (LOBs)

There are four LOB data types in Oracle9i.

- BLOB represents a binary large object.
- CLOB represents a character large object.
- NCLOB represents unicode characters. Both fixed-width and variable-width character sets are supported, both using the NCHAR database character set.
- BFILE represents a binary file stored outside the database.

LOBs are characterized in two ways: their interpretation by the Oracle9i Server (binary or character), and their storage aspects. LOBs can be stored inside the database (Internal LOBs) or in host files (External LOBs). There are two categories of LOB:

- Internal LOBs (CLOB, NCLOB, BLOB)—Stored in the database
- External files (BFILE)—Stored outside the database

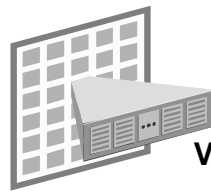
A DIRECTORY item specifies an alias for a directory on the server's file system. By granting suitable privileges on these items to users, secured access can be provided to files in the corresponding directories on a user-by-user basis (certain directories can be made read-only, inaccessible, and so on).

Further, these directory aliases can be used while referring to database files (open, close, read, and so on) in PL/SQL and OCI. This provides applications abstraction from hard-coded pathnames, providing flexibility in portably managing file locations.

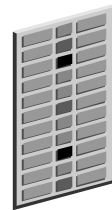
The DIRECTORY item is owned by the system and created by the DBA.

Collections

- **Objects that contain objects**
- **VARRAYs are ordered sets of elements containing a count and a limit**
- **Nested tables are tables with a column or variable of the TABLE data type**



VARRAY



Nested table

ORACLE

3-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Collections

A collection is an object that contains other objects, where each contained object is of the same type. Collection types are parameterized data types.

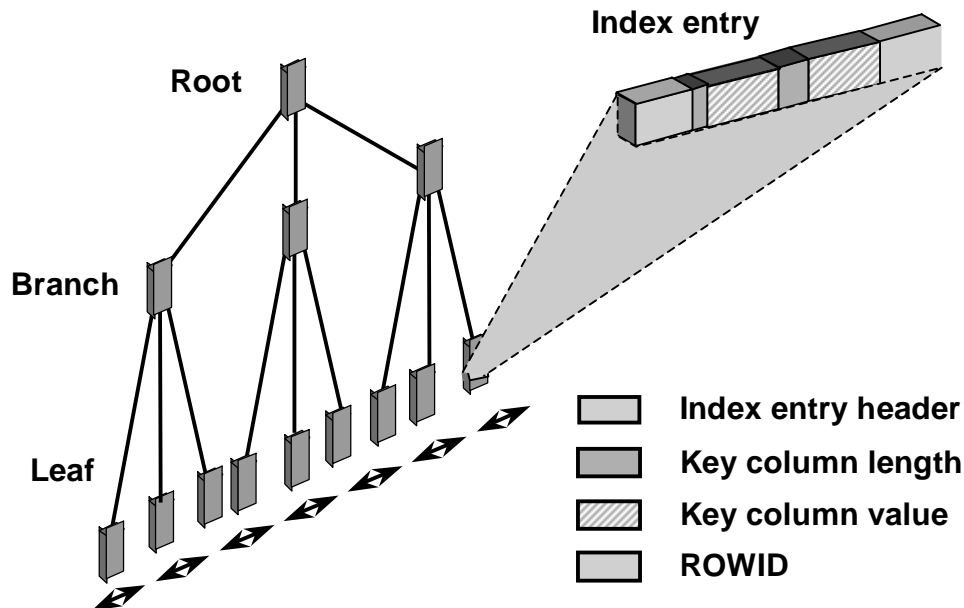
Collection types are arrays (VARRAY data type) and nested tables.

VARRAYs can be used to hold lists of objects, REFs, or scalar data types. They are primarily useful in dealing with small sets, where individual elements need not be manipulated through SQL.

Nested tables are used where large sets of data need to be placed within a parent table and are typically useful where there is a master-detail relationship, where for a given master row, there could be many detailed rows.

With Oracle9i Database, it is possible to use multilevel collections. Multilevel collection types are collection types where the collection element itself is directly or indirectly another collection type. This means that one collection can be contained in another and this can occur in several levels. This is useful when working with the object-oriented approach in application development.

B*-Tree Indexes



B*-Tree Indexes

Indexes are optional structures associated with tables and clusters. You can create indexes on one or more columns of a table to speed SQL statement execution on that table. Just as an index in any manual helps you locate information faster than if there were no index, an Oracle index provides a faster access path to table data. Indexes are the primary means of reducing disk I/O when properly used.

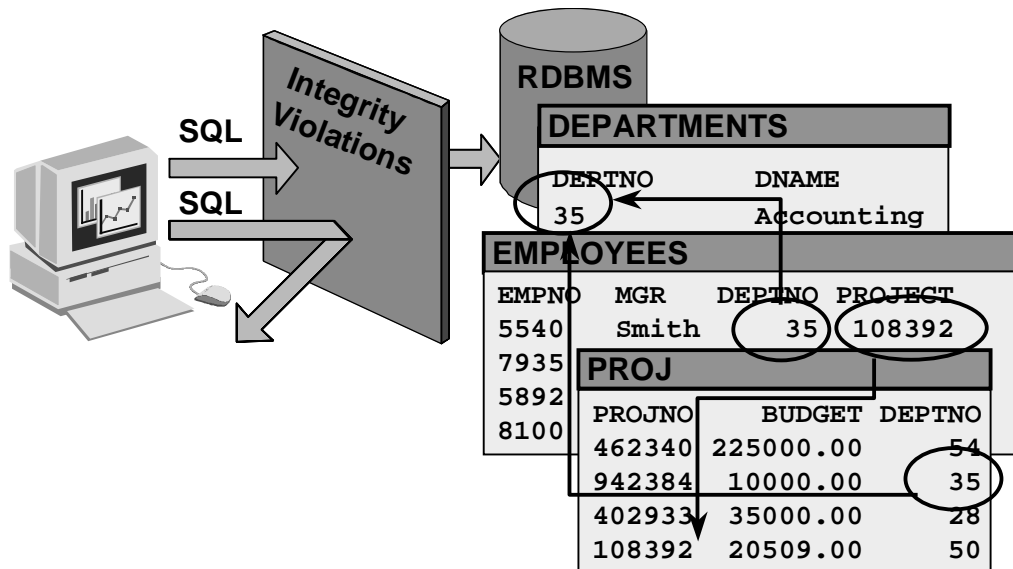
Oracle provides several indexing schemes, which provide complementary performance functionality: B*-tree indexes (currently the most common), B*-tree cluster indexes, hash cluster indexes, reverse key indexes, and bitmap indexes. Oracle also provides support for function-based indexes and domain indexes specific to an application or cartridge.

The absence or presence of an index does not require a change in the wording of any SQL statement. Given a data value that has been indexed, the index points directly to the location of the rows containing that value.

Oracle automatically maintains and uses indexes after they are created. Oracle automatically reflects changes to data, such as adding new rows, updating rows, or deleting rows, in all relevant indexes with no additional action by users.

Oracle uses B*-tree indexes that are balanced to equalize access times to any row. The upper blocks (branch blocks) of a B*-tree index contain index data that points to lower level index blocks. The lowest level index blocks (leaf blocks) contain every indexed data value and a corresponding ROWID used to locate the actual row. The leaf blocks are doubly linked.

Data Integrity Constraints and Triggers



ORACLE

3-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Data Integrity

It is important that data adhere to a predefined set of rules, as determined by the database administrator or application developer. This section describes the rules that can be applied to table columns to enforce different types of data integrity.

- **Nulls:** A null is a rule defined on a single column that allows or disallows inserts or updates of rows containing a null (the absence of a value) in that column.
- **Unique Column Values:** A unique value defined on a column (or set of columns) allows the insert or update of a row only if it contains a unique value in that column (or set of columns).
- **Primary Key Values:** A primary key value defined on a key (a column or set of columns) specifies that each row in the table can be uniquely identified by the values in the key.
- **Referential Integrity:** A rule defined on a key (a column or set of columns) in one table that guarantees that the values in that key match the values in a key in a related table (the referenced value). Referential integrity also includes the rules that dictate what types of data manipulation are allowed on referenced values and how these actions affect dependent values.
- **Check:** Constraints enable you to enforce very specific or sophisticated integrity rules by specifying a check condition. The condition of a CHECK constraint needs to be static.
- **Trigger:** Oracle also allows you to enforce integrity rules with a nondeclarative approach using database triggers.

You can disable integrity constraints temporarily so that large amounts of data can be loaded without the overhead of constraint checking.

Constraint Features

- **Deferred constraint checking (until COMMIT)**
- **Enable constraints without checking table data:**
`ENABLE NOVALIDATE`
- **Validate table data without enabling a constraint:**
`DISABLE VALIDATE`
- **Unique and primary key constraint checking with nonunique indexes (or no indexes at all)**
- **Indicate validity of data without enabling/validating a constraint:** `DISABLE NOVALIDATE RELY`

ORACLE

3-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Constraint Features

By default, constraints are checked immediately. If a constraint is violated, the statement is rolled back. In Oracle9i, you can also defer constraint checking until the end of your transaction.

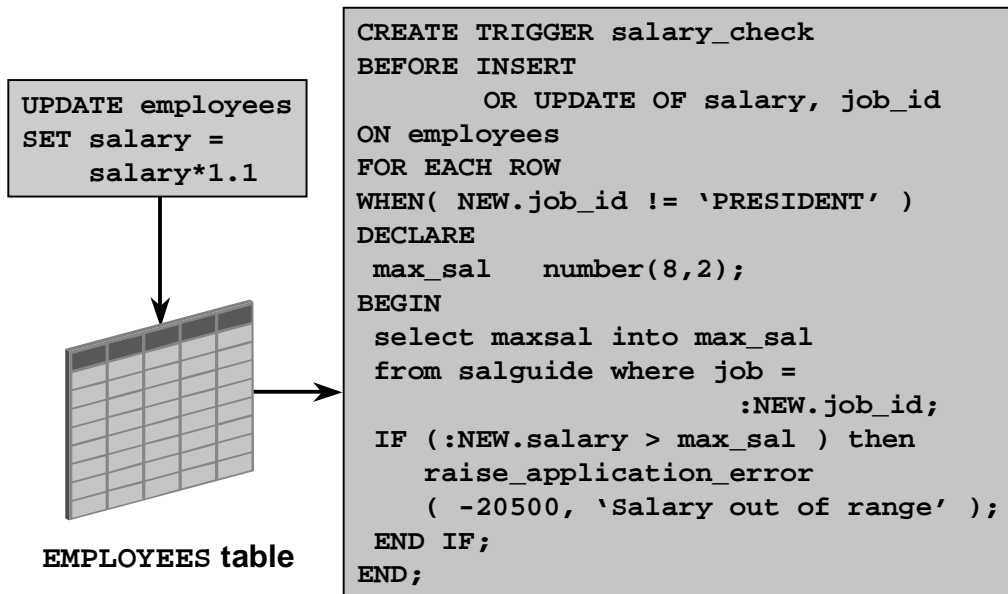
By default, the table data is checked when you enable a constraint. You can choose to enable a constraint (checking new DML against the table) without validating the current data. This avoids locking, and allows concurrent DML while enabling the constraint.

When you disable a constraint, the data is not checked by default. However, you can `DISABLE VALIDATE` a constraint, which means that the constraint is disabled and associated indexes can be dropped while the constraint is kept valid. So you can have index-less constraints; further DML is not allowed. This is a very useful feature in large (read-only) data warehouses.

For unique or primary key indexes, Oracle9i needs to create or make use of an existing index in order to check for uniqueness. If violations occur during the building or reuse of an index, the constraint cannot be validated.

The `RELY` flag provides a mechanism to indicate that you *believe* that a given constraint is true, without actually having to require that the database verifies this. This feature is used internally by the summary management facility and query rewrites.

Database Triggers



ORACLE

3-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Database Triggers

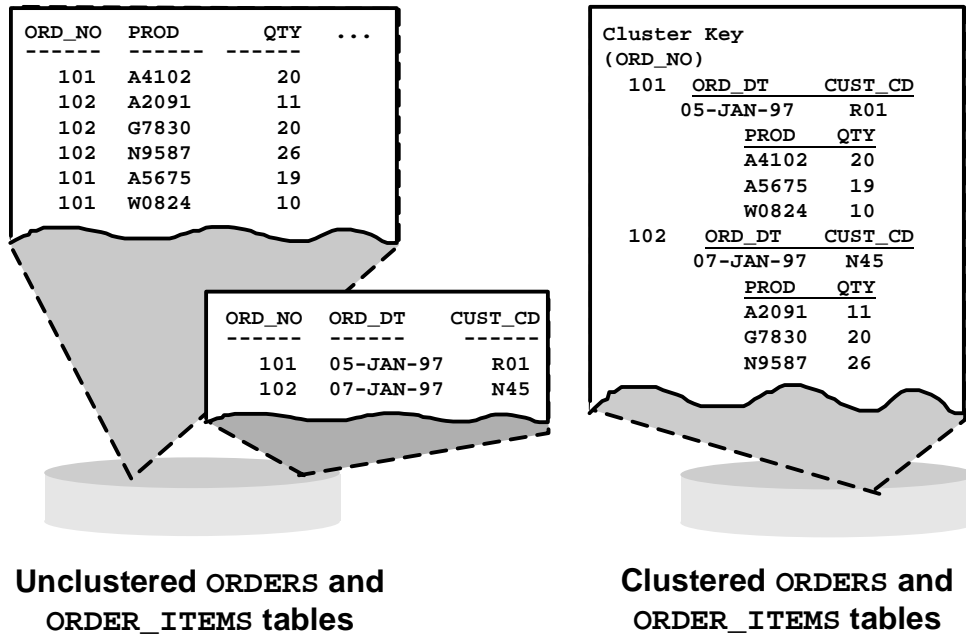
Oracle allows you to define procedures called triggers that execute implicitly when an INSERT, UPDATE, or DELETE statement is issued against the associated table (or, in some cases, against a view) or when database system actions occur. These procedures can be written in PL/SQL or Java and stored in the database, or they can be written as C callouts.

A trigger stored in the database can include PL/SQL or Java statements to execute as a unit and can invoke stored procedures. However, procedures and triggers differ in the way that they are invoked. A procedure is explicitly executed by a user, application, or trigger. Triggers (one or more) are implicitly fired (executed) by Oracle when a triggering event occurs, no matter which user is connected or which application is being used. The example above shows a database application with a SQL statement that implicitly fires a trigger stored in the database.

A triggering event or statement is the SQL statement, database event, or user event that causes a trigger to be fired. A triggering event can be one or more of the following:

- An INSERT, UPDATE, or DELETE statement on a specific table (or view, in some cases)
- A CREATE, ALTER, or DROP statement on any schema object
- A database startup or instance shutdown
- A specific error message or any error message
- A user logon or logoff

Clusters



ORACLE

3-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Clusters

Clusters are an optional method of storing table data. A cluster is a group of tables that share the same data blocks because they share common columns and are often used together. For example, the ORDERS and ORDER_ITEMS table share the ORDER_ID column. When you cluster the ORDERS and ORDER_ITEMS tables, Oracle physically stores all rows for each order from both the ORDERS and ORDER_ITEMS tables in the same data blocks.

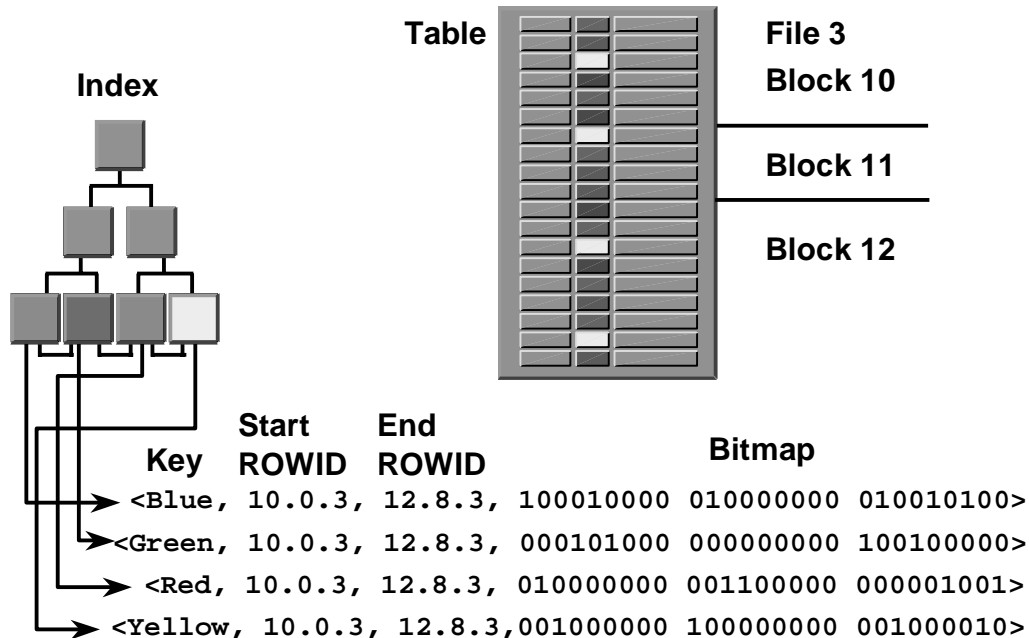
The Cluster Index

A cluster index is an index defined specifically for a cluster. Such an index contains an entry for each cluster key value. To locate a row in a cluster, the cluster index is used to find the cluster key value, which points to the data block associated with that cluster key value. Therefore, Oracle accesses a given row with a minimum of two I/Os.

Hash Clusters

Hashing is an optional way of storing table data to improve the performance of data retrieval. To use hashing, you create a hash cluster and load tables into the cluster. Oracle physically stores the rows of a table in a hash cluster and retrieves them according to the results of a hash function. The key of a hash cluster (like the key of an index cluster) can be a single column or composite key. To find or store a row in a hash cluster, Oracle applies the hash function to the row's cluster key value; the resulting hash value corresponds to a data block in the cluster, which Oracle then reads or writes on behalf of the issued statement.

Bitmap Indexes



ORACLE

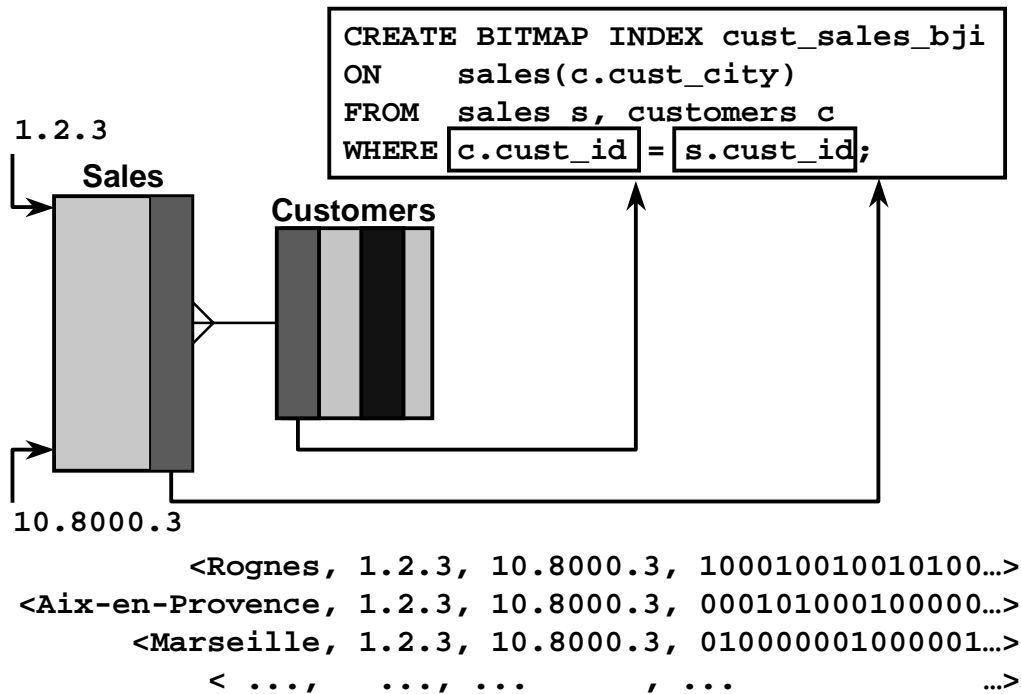
3-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Bitmap Indexes

The purpose of an index is to provide pointers to the rows in a table that contain a given key value. In a regular index, this is achieved by storing a list of ROWIDs for each key corresponding to the rows with that key value. In a bitmap index, a bitmap for each key value is used instead of a list of ROWIDs. Each bit in the bitmap corresponds to a possible ROWID, and if the bit is set, it means that the row with the corresponding ROWID contains the key value. A mapping function converts the bit position to an actual ROWID, so the bitmap index provides the same functionality as a regular index even though it uses a different representation internally. If the number of different key values is small, bitmap indexes are very space efficient. Bitmap indexing efficiently merges indexes that correspond to several conditions in a WHERE clause. Rows that satisfy some, but not all conditions are filtered out before the table itself is accessed. This improves response time, often dramatically.

What Is a Bitmap Join Index?



ORACLE

3-13

Copyright © Oracle Corporation, 2001. All rights reserved.

What Is a Bitmap Join Index?

In addition to a bitmap index on a single table, you can create a bitmap join index in Oracle9i. A bitmap join index 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.

Here, we create a new bitmap join index named `cust_sales_bji` on the `SALES` table. The key of this index is the column `CUST_CITY` of the `CUSTOMERS` table. This example assumes that there is an enforced primary key constraint on `CUSTOMERS` in order to ensure that what is stored in the bitmap reflect the reality of the data in the tables. The column `CUST_ID` is the primary key of `CUSTOMERS` but is also a foreign key inside `SALES`, although not required.

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

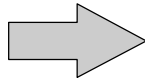
The middle part of the above graphic shows you a theoretical implementation of this bitmap join index. Each entry or key in the index represents a possible city found in the `CUSTOMERS` table. A bitmap is then associated to one particular key. Each bit in a bitmap corresponds to one row in the `SALES` table. In the first key above (Rognes), you can see that the first row in the `SALES` table corresponds to a product sold to a Rognes customer, while the second bit is not a product sold to a Rognes customer.

Other Index Types

- **Function-based indexes:**
Based on expressions or functions

```
SQL> create index sales_profit_idx  
2 on sales(revenue-cost);
```

Index can
be used



```
SQL> select ordid from sales  
2 where (revenue-cost)>=1000;
```

- **Descending indexes:**
Avoid sorting

ORACLE

3-14

Copyright © Oracle Corporation, 2001. All rights reserved.

Function-Based Indexes

Function-based indexes can be useful in several cases:

- When the SELECT list uses expressions that can be materialized as a function-based index value
- When the WHERE clause contains expressions, as shown in the example, to avoid full table scans
- While using different NLS collating sequences

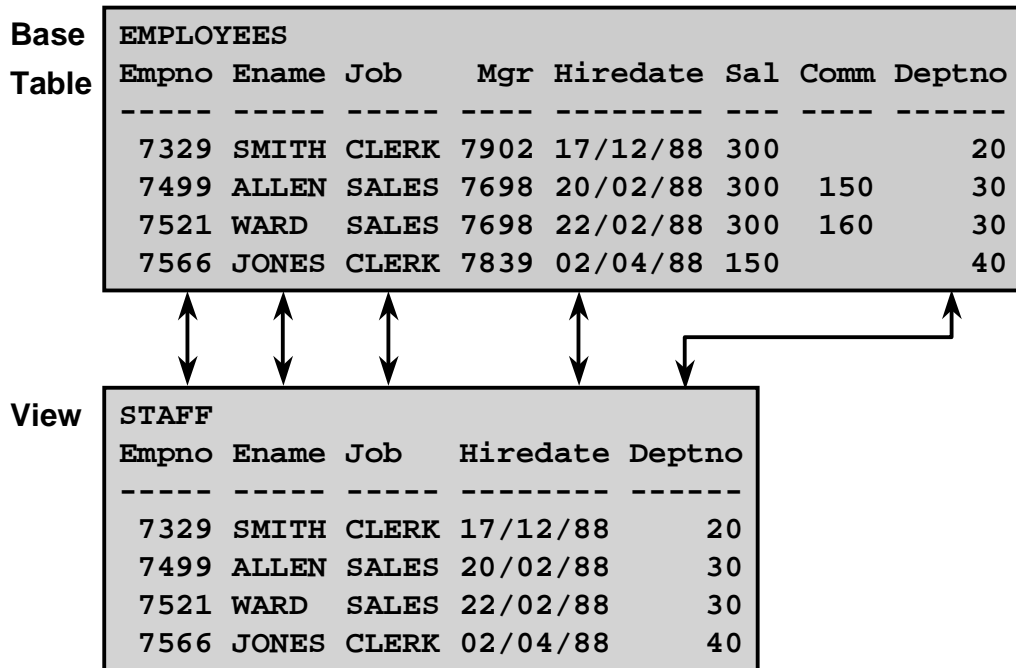
Function-based indexes can use any function or object method that is declared as *repeatable*. That is, the returned value cannot change for a given set of input values.

Descending Indexes

In Oracle9i, indexes can also be stored in descending order of key values. This is, for example, useful for concatenated indexes where you frequently sort ascending on the first column and descending on the second column.

Note that the DESCENDING keyword has always been supported by Oracle for compatibility reasons; however, it was ignored until Oracle8i.

Views



ORACLE

3-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Views

A view is a tailored presentation of the data contained in one or more tables. A view takes the output of a query and treats it as a table. Therefore, a view can be thought of as a *stored query* or a *virtual table*. You can use views in most places where a table can be used.

For example, the EMP table has several columns and numerous rows of information. If you only want users to see five of these columns, or only specific rows, you can create a view of that table for other users to access.

Since views are derived from tables, they have many similarities. You can query views, and with some restrictions you can update, insert into, and delete from views. All operations performed on a view actually affect data in some base table of the view and are subject to the integrity constraints and triggers of the base tables.

Unlike a table, a view is not allocated any storage space, nor does a view actually contain data; rather, a view is defined by a query that extracts or derives data from the tables the view references. These tables are called base tables. Base tables can in turn be actual tables or can be views themselves. Because a view is based on other objects, a view requires no storage other than storage for the definition of the view (the stored query) in the data dictionary.

Views can be used as restricted windows on your data and so it can secure access to them and also simplify their access.

An inline view is not a schema object, but rather it is a subquery with an alias (correlation name) that you can use like a view within a SQL statement.

Materialized Views

A materialized view:

- Is an instantiation of a SQL statement
- Has its data stored in tables, offering:
 - Space management options
 - Use of indexes and partitions
- Can be compared to an index in many ways
- Used for:
 - Data warehouses
 - Distributed computing
 - ...

ORACLE

3-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Materialized Views

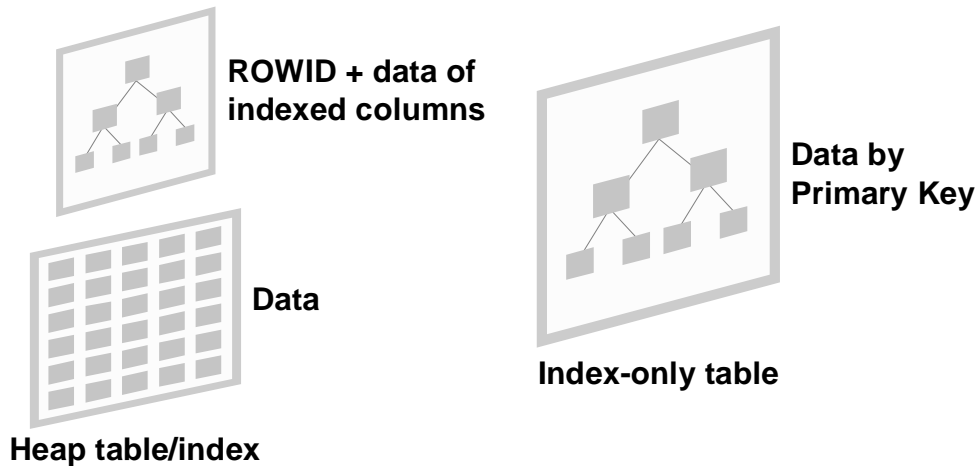
One technique commonly used to improve performance is the creation of materialized views. Materialized views are objects that improve query execution times by precalculating expensive joins and aggregation operations prior to execution, and storing the results in a database table. For example, a table T could be created to contain the sum of sales by region and by product. If a database application needs to retrieve the sum of sales by region and by product, it is more efficient to directly query T than to access the SALES table. However, this manual technique has at least two disadvantages:

- The database application needs to be aware of table T.
- It is possible that table T does not reflect reality. (What if one modifies SALES?)

Oracle9i automates these processes by using materialized views. When the end-user application executes queries, the Oracle optimizer may transparently rewrite the SQL query to use the materialized view (analog to indexes). This mechanism significantly improves the response time and eliminates the need for database applications to be aware of precreated materialized views. Materialized views can be refreshed automatically, ensuring that both materialized views and database tables are synchronized.

For more information on materialized views, see Oracle9i Business Intelligence and Oracle9i Replication lessons.

Index-Organized Tables



Index-Organized Tables

An index-organized table differs from an ordinary table (called a heap table) in that the data for the table is held in its associated index. Changes to the table data, such as adding new rows, updating rows, or deleting rows, result only in updating the index.

The index-organized table is like an ordinary table with an index on one or more of its columns, but instead of maintaining two separate storages for the table and the B*-tree index, the database system only maintains a single B*-tree index which contains both the encoded key value and the associated column values for the corresponding row. Rather than having a row's ROWID as the second element of the index entry, the actual data row is stored in the B*-tree index. The data rows are built on the primary key for the table, and each B*-tree index entry contains

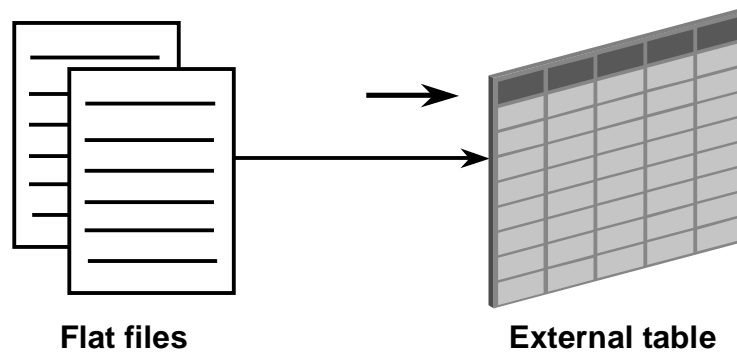
`<primary_key_value, non_primary_column_values>`

Index-organized tables are suitable for accessing data by the primary key or any key that is a valid prefix of the primary key. There is no duplication of key values because only non-key column values are stored with the key. You can build secondary indexes to provide efficient access by other columns. Applications manipulate the index-organized table just like an ordinary table, using SQL statements. However, the database system performs all operations by manipulating the corresponding B*-tree index.

In order to keep the B*-tree structures small and efficient, index-organized tables normally have an overflow segment, especially if the rows are quite long. You can define which part of the row should be stored into the B*-tree, and which part should go to the overflow segment.

External Tables

- Read data from text files in the operating system
- Presents data as a normal table inside the database
- Reduce time and complexity when loading data



External Tables

Oracle9i's external table feature allows you to use external data as a virtual table that can be queried and joined directly and in parallel without requiring the external data to be first loaded in the database.

External tables are like regular SQL tables with the exception that the data is read-only and does not reside in the database, thus the organization is external. As a result, the external table acts as 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 them.

The `CREATE TABLE ... ORGANIZATION EXTERNAL` operation involves only the creation of metadata in the Oracle Dictionary since the external data already exists outside the database in text files. Once the metadata is created, the external table feature enables the user to easily perform parallel extraction of data from the specified external sources.

External Tables utilizes SQL*Loader functionality. The arguments given when creating an external table are similar to those found in SQL*Loader control files.

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

3-19

Copyright © Oracle Corporation, 2001. All rights reserved.

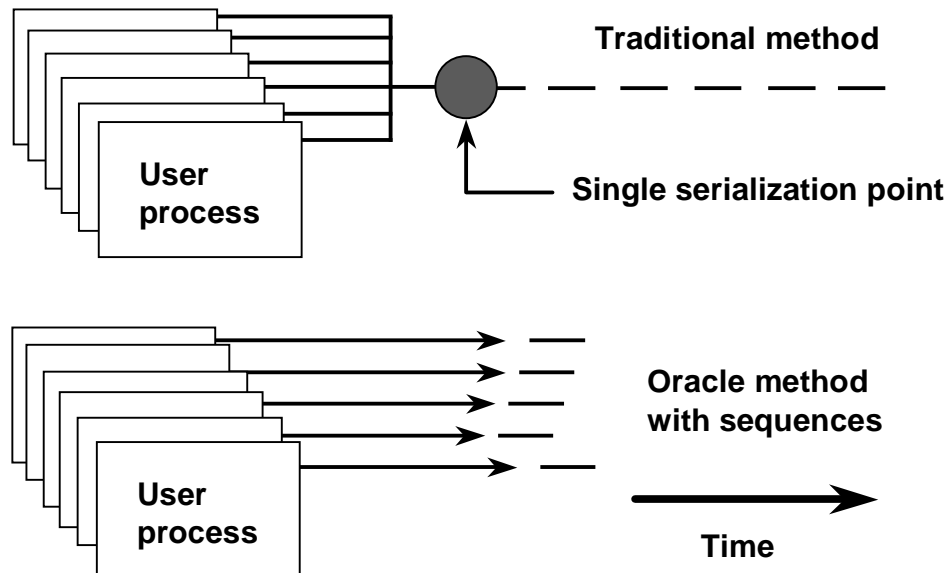
Table Functions

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 avoids intermediate staging tables, which interrupt the data flow through the various transformation steps.

External tables enable the pipelining of the loading phase with the transformation phase. The transformation process can be merged 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.

Sequence Number Generation



ORACLE

3-20

Copyright © Oracle Corporation, 2001. All rights reserved.

Sequence Number Generation

A system often needs to generate a unique number, for example, a number for a sales order.

One way to achieve this is to have a row within a control table that contains the last used number. A transaction creating an order locks the row, reads the number, and then writes the number +1 back to the control table. When the transaction commits, the lock on the control table is released and another transaction can access the next number. In an OLTP system, the control table becomes a bottleneck, as all transactions are serialized on the control table.

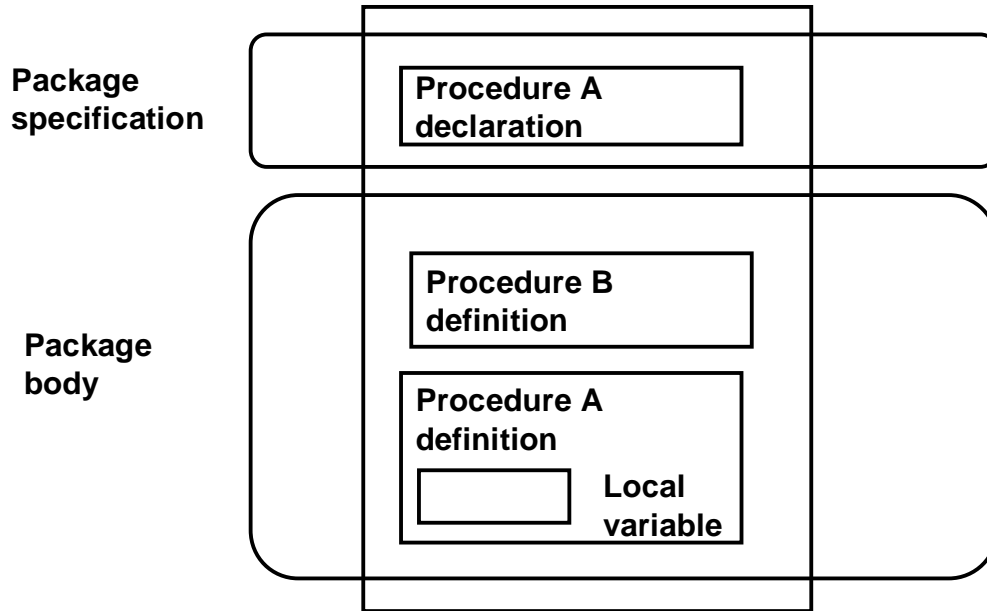
1. `select LASTVALUE+1 into X from SEQ_TAB for update of LASTVALUE`
2. `insert into ORDERS values (:X, ...)`
3. `update SEQ_TAB set LASTVALUE = :X`
4. `commit`

Oracle provides a sequence number generator to solve this problem. It can be thought of as a set of unique numbers, cached in memory. Transactions do not require locks on the sequence number and Oracle guarantees they are unique.

1. `insert into ORDERS values (ORD_SEQ.NEXTVAL, ...)`
2. `commit`

The Oracle method also requires fewer SQL statements, which is extremely beneficial in client-server environments.

Packages



ORACLE

3-21

Copyright © Oracle Corporation, 2001. All rights reserved.

Packages

Packages bundle related PL/SQL types, items, and subprograms structures into one container. For example, a hr package might contain hiring and firing procedures, commission and bonus functions, and tax exemption variables.

Usually a package has a specification and a body, stored separately in the database.

The specification is the interface to your applications. It declares the types, variables, constants, exceptions, cursors, and subprograms available for use.

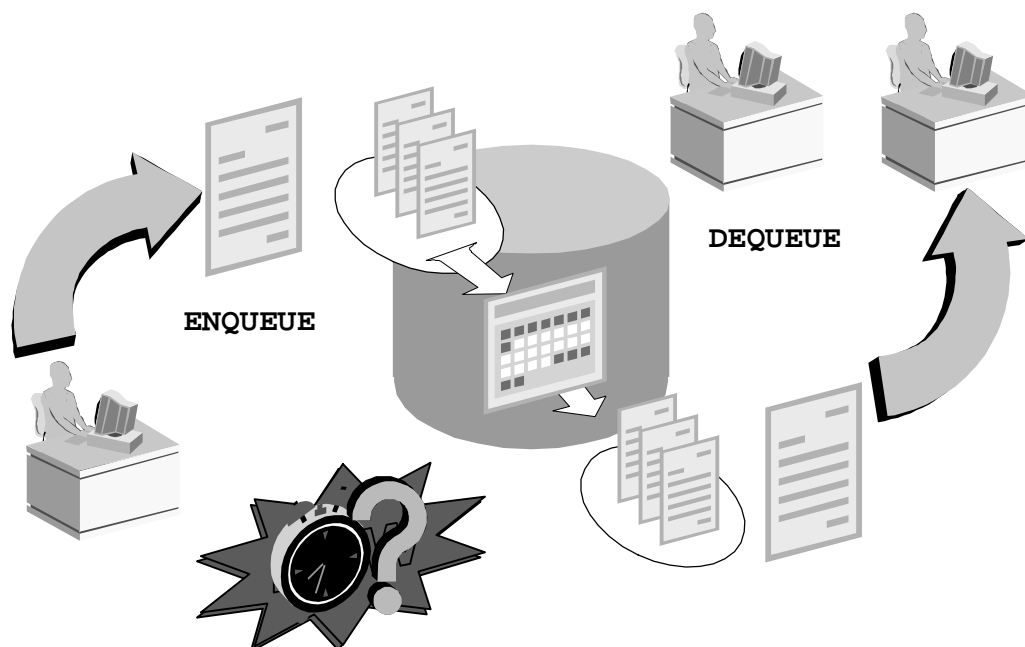
The body fully defines cursors and subprograms, and so implements the specification.

The package itself cannot be called, parameterized, or nested. Still, the format of a package is similar to that of a subprogram. Once written and compiled, the contents can be shared by many applications.

When calling a packaged PL/SQL construct for the first time, the whole package is loaded into memory. Therefore, later calls to related constructs require no disk I/O.

Because all packages in Oracle9i can be native-compiled, performance is increased as compared to the traditional method. This is done by translating the package to C code, and then compiling that C code and mapping it into the database again. All of this is done from within the database.

Queues



ORACLE

3-22

Copyright © Oracle Corporation, 2001. All rights reserved.

Queues

In client/server environments, requests are executed immediately. This is often called online or connected execution of work. However, there are many environments in which a deferred or disconnected execution is preferred or even required.

Queuing implements deferred execution of work. When a request for work is entered, queuing defers processing of that request until the requester has completed the task, process, or transaction that created the request.

It is often important that each request is processed exactly once, even in the presence of application and system failures. By integrating transaction processing with queuing technology, persistent queuing supports this requirement. The importance of queuing has been proven by transaction processing (TP) monitors which typically include such a facility.

Business process management and workflow are more and more recognized as fundamental technologies for emerging applications. Queuing is one of the key technologies for this class of applications.

Oracle provides an efficient queuing infrastructure that does not depend on the use of TP monitors or other evolving message-oriented middleware (MOM) infrastructure.

Queues (continued)

Oracle Advanced Queuing (AQ) offers the following functionality:

- Structured payload for messages
- Priority and ordering of messages in queues
- Ability to specify a window of execution for each message
- Ability to query queues using standard SQL
- Integrated transactions to simplify application development and management
- Ability to dequeue multiple messages as a bundle
- Ability to specify multiple recipients
- Ability to propagate messages to queues in local or remote Oracle databases
- Rule-based publish/subscribe with content based filtering
- Ability to wait for messages on multiple queues
- Persistent and non-persistent queuing
- Statistics on messages stored in queues and propagated to other queues
- Retention of messages and message history for analysis purposes
- Queue level access control
- Exception handling support
- Support for Oracle Parallel Server environment to achieve higher performance asynchronous notification using callback functions

Since Oracle AQ queues are implemented in database tables, all the operational benefits of high availability, scalability, and reliability are applicable to queue data. In addition, you can use database development and management tools with queues.

4

Basic Oracle9i Architecture

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

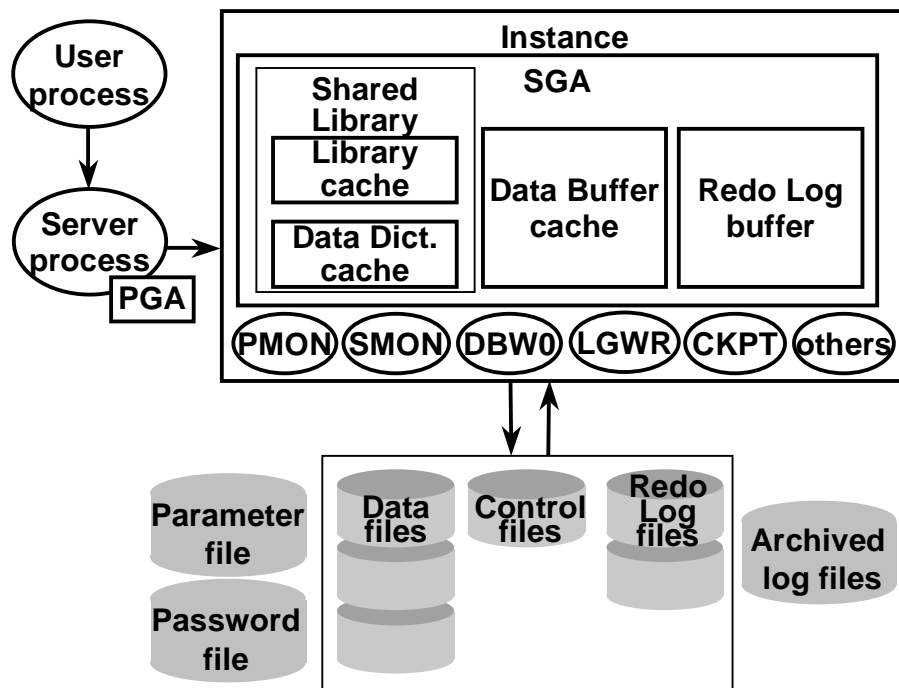
Objectives

After completing this lesson, you should be able to describe the basic Oracle9i architecture:

- **Logical and physical data structures**
- **The data dictionary**
- **Instances, memory structures, and processes**
- **Server architecture, including Shared Servers**

ORACLE

Instance and Database



ORACLE

4-3

Copyright © Oracle Corporation, 2001. All rights reserved.

Instance and Database

Every running Oracle database is associated with an Oracle instance. When a database is started on a database server, Oracle allocates a shared memory area called the System Global Area (SGA) and starts several Oracle processes. This combination of the SGA and the Oracle processes is called an Oracle instance.

After starting an instance, Oracle associates the instance with a specified database. This is called mounting the database. The database is then ready to be opened, which makes it accessible to authorized users. Multiple instances can execute concurrently on the same computer, each accessing its own physical database.

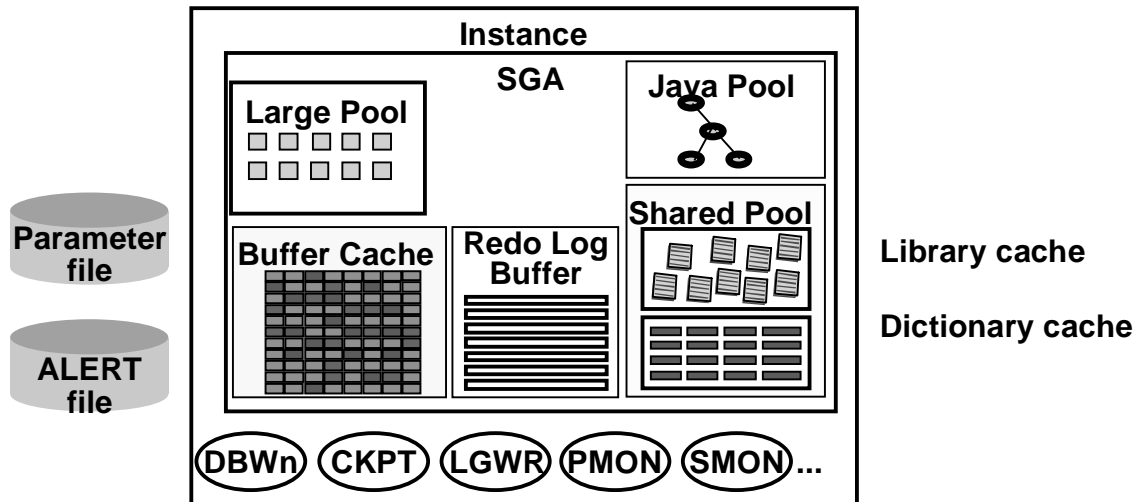
Server Processes

Oracle creates server processes to handle requests from connected user processes. A server process is in charge of communicating with the user process and interacting with Oracle to carry out requests of the associated user process. Oracle can be configured to vary the number of user processes per server process. On some systems, the user and server processes are separate, while on others they are combined into a single process. Client/server systems separate the user and server processes and execute them on different machines.

Program Global Area (PGA)

The PGA is a private (non-shared) memory buffer that contains data and control information for a server process. A PGA is created by Oracle when a server process is started.

Instance Components



4-4

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

Parameter File

To start an instance, Oracle must read a parameter file, which is a text file containing a list of initialization parameter settings for that instance and database, or a binary file called an SPFILE maintained by Oracle. You will see more information on SPFILE in the next lesson.

Background Processes

Oracle creates a set of background processes for each instance. The background processes asynchronously perform I/O and monitor other Oracle processes to provide increased parallelism for better performance and reliability. Each Oracle instance may use several background processes. The names of these processes are DBWn, LGWR, CKPT, SMON, PMON, ARCn, RECO, and so on.

Database Buffer Cache

The database buffer cache is the portion of the SGA that holds copies of data blocks read from the data files. All user processes share access to the database buffer cache.

Redo Log Buffer

The redo log buffer is a circular buffer that holds information about changes made to the database. This information is stored in redo entries. Redo entries contain the information necessary to reconstruct, or redo, changes made to the database. Redo entries are used for database recovery, if necessary. The background process LGWR writes the redo log buffer to the active online redo log file, or group of files, on disk.

Shared Pool

The shared pool contains three major areas: library cache, dictionary cache, and some internal control structures.

The **library cache** includes the shared SQL areas, private SQL areas, PL/SQL procedures and packages, and control structures such as locks and library cache handles.

The data dictionary is a collection of database tables and views containing reference information about the database, its structures, and its users. Oracle accesses the data dictionary frequently during the parsing of SQL statements, and has a **dictionary cache** to keep data dictionary information in memory to avoid time-consuming data dictionary lookups.

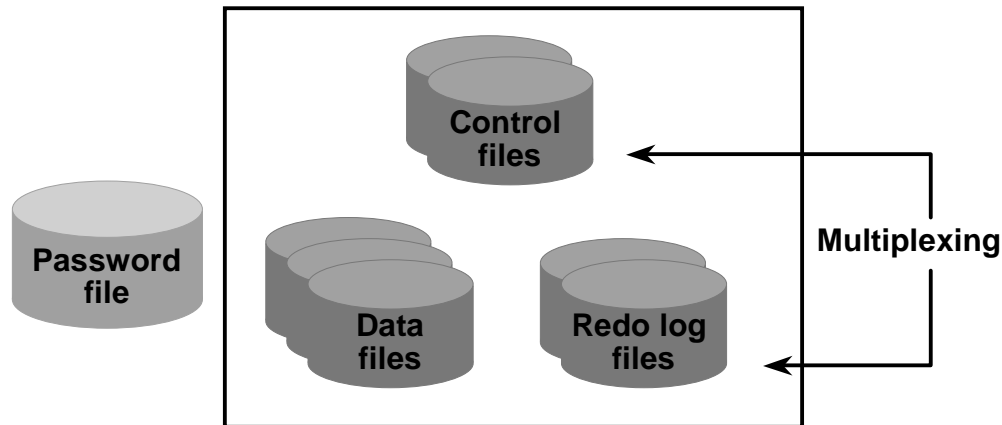
Large Pool

The database administrator can configure an optional memory area called the large pool to provide large memory allocations for session memory for the Shared Server and the Oracle XA interface and I/O server processes during Recovery Manager's backup and restore operations. The large pool is also used for parallel execution, to pass messages between the processes involved.

Java Pool

The primary initialization parameters that affect Java usage and performance are `shared_pool_size` and `java_pool_size`. The database initialization process requires `shared_pool_size` to be set to 50MB as it loads the Java binaries for more than 8,000 classes and resolves them. The `shared_pool_size` resource is also consumed when you create call specifications and while the system tracks dynamically loaded Java classes at runtime. The server side Java VM memory manager allocates all other Java states during runtime execution from the amount of memory allocated using `java_pool_size`. This memory includes the shared in-memory representation of Java method and class definitions, as well as the Java objects that are migrated to session space at end-of-call.

Database Components



ORACLE

4-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Data Files

Every Oracle database has one or more physical data files. A database's data files contain all the database data. The data of logical database structures such as tables and indexes is physically stored in the data files allocated for a database.

Redo Log Files

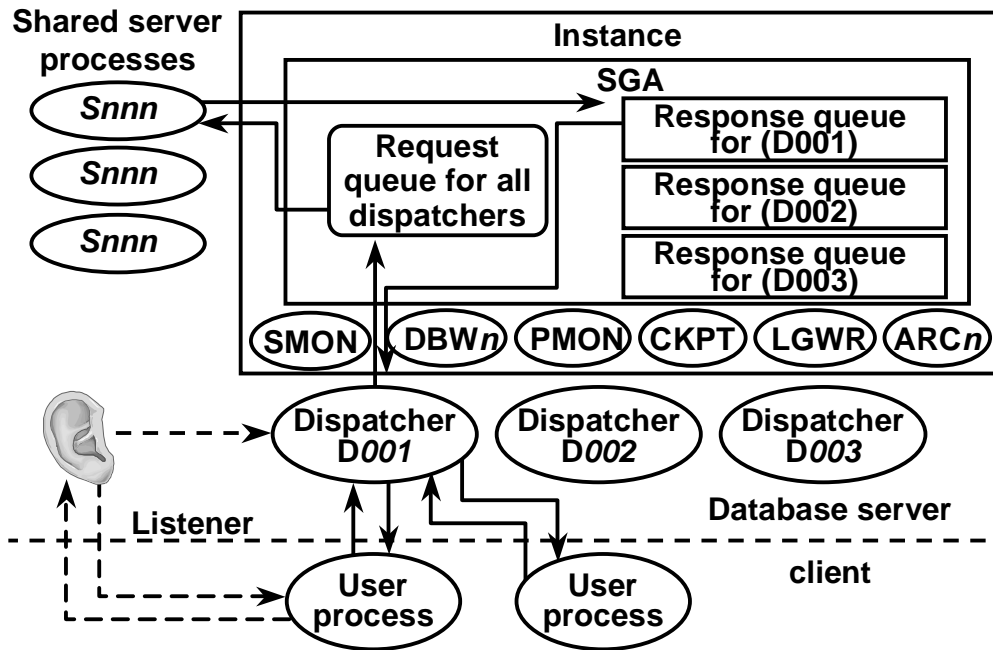
Every Oracle database has a set of two or more redo log files. The set of redo log files for a database is collectively known as the database's redo log. A redo log is made up of redo entries, each of which is a group of change vectors describing a single atomic change to the database. The primary function of the redo log is to record all changes made to the database. Should a failure prevent modified data from being permanently written to the data files, the changes can be obtained from the redo log and work is never lost. Redo log files should be multiplexed to protect them against failures.

In case of instance failures, Oracle will automatically recover the database using information from the current online redo log files.

Control Files

Every Oracle database has at least one control file. A control file contains entries that specify the physical structure of the database. Control files should be multiplexed to protect them against failures.

Shared Server



ORACLE

4-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Shared Server

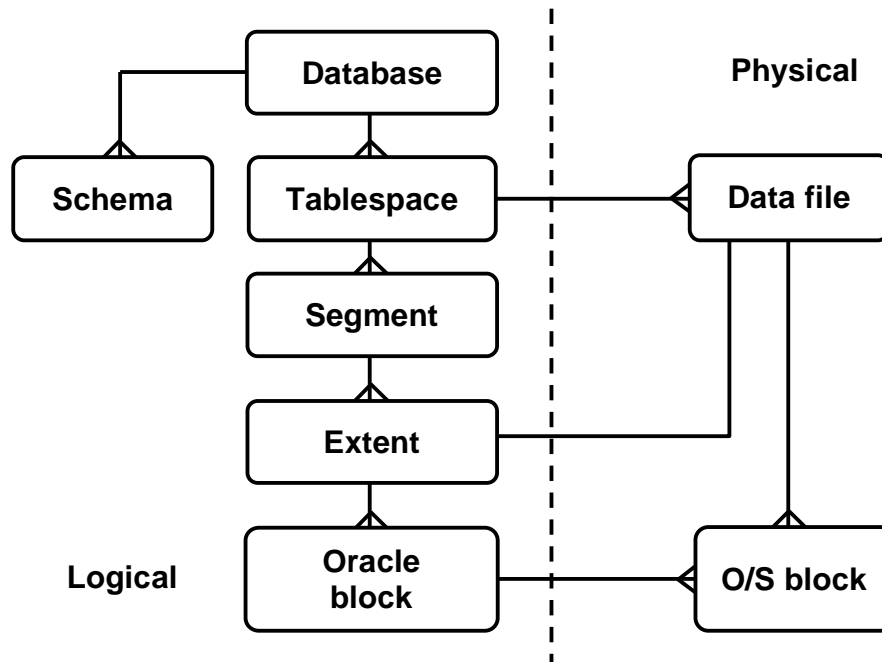
The shared server configuration allows many user processes to share very few server processes. The user processes connect to a dispatcher background process, which routes client requests to the next available shared server process. The advantage of the shared server configuration is that system overhead is reduced, increasing the number of users that can be supported. A small number of shared server processes can perform the same amount of processing as many dedicated server processes, and the amount of memory required for each user is relatively small. The shared server requires Net. When an instance starts, the network listener process opens and establishes a communication pathway through which users connect to Oracle. Then, each dispatcher process gives the listener process an address at which the dispatcher listens for connection requests. At least one dispatcher process must be configured and started for each network protocol that the database clients will use. When a user process makes a connection request, the listener examines the request and determines whether the user process can use a shared server process. If so, the listener returns the address of the dispatcher process that has the lightest load and the user process connects to the dispatcher directly.

Note: More shared server features, such as connection pooling and multiplexing, are discussed in a later lesson.

Dispatcher Request and Response Queues

A request from a user is a single program interface call that is part of the user's SQL statement. When a user makes a call, its dispatcher places the request on the request queue, where it is picked up by the next available shared server process. The request queue is in the SGA and is common to all dispatcher processes of an instance. The shared server processes check the common request queue for new requests, picking up new requests on a first-in-first-out basis. One shared server process picks up one request in the queue and makes all necessary calls to the database to complete that request. When the server completes the request, it places the response on the calling dispatcher's response queue. Each dispatcher has its own response queue in the SGA. The dispatcher then returns the completed request to the appropriate user process.

Database Structure



ORACLE

4-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Database Structure

An Oracle database is a collection of data that is treated as a unit. The general purpose of a database is to store and retrieve related information. The database has logical structures and physical structures.

Tablespaces

A database is divided into logical storage units called tablespaces, that group related logical structures together. For example, tablespaces commonly group all of an application's objects to simplify some administrative operations. In Oracle9i, you also have different tablespace types such as UNDO tablespace for undo information generation, that will be used if someone writes rollback instead of commit.

Databases, Tablespaces, and Data files

The relationship among databases, tablespaces, and data files is illustrated in the above slide. Each database is logically divided into one or more tablespaces. One or more data files are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace. If it is a TEMPORARY tablespace, instead of a data file, the tablespace has a temporary file.

Schemas

A schema is a collection of database objects, owned by a database user. Schema objects are the logical structures that directly refer to the database's data. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. In general, everything your application creates in the database.

Data Blocks

At the finest level of granularity, an Oracle database's data is stored in data blocks. One data block corresponds to a specific number of bytes of physical database space on disk. A data block size is specified for each tablespace when it is created. A database uses and allocates free database space in Oracle data blocks.

Extents

The next level of logical database space is called an extent. An extent is a specific number of contiguous data blocks, obtained in a single allocation, used to store a specific type of information.

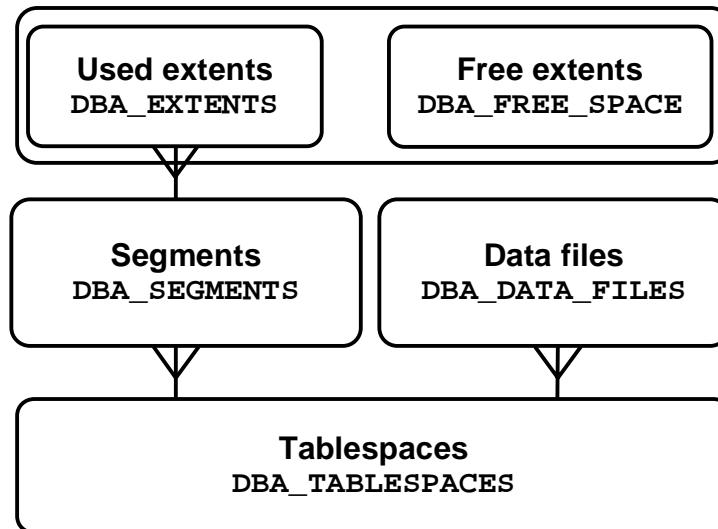
Segments

The level of logical database storage above an extent is called a segment. A segment is a set of extents allocated for a certain logical structure. For example, the different types of segments include:

- *Data segments*: Each non-clustered table has a data segment. All of the table's data is stored in the extents of its data segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
- *Index segments*: Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.
- *Undo segments*: One UNDO tablespace is created by the database administrator to temporarily store *undo* information. The information in an undo segment is used to generate read-consistent database information during database recovery to roll back uncommitted transactions for users.
- *Temporary segments*: Temporary segments are created by Oracle when a SQL statement needs a temporary work area to complete execution. When the statement finishes execution, the temporary segment's extents are returned to the system for future use. If you specify a default temporary tablespace for every user, or global in the database as a default temporary tablespace, you can make sure that everything that spills to disk is in the proper location.

Oracle dynamically allocates space when the existing extents of a segment become full. Therefore, when the existing extents of a segment are full, Oracle allocates another extent for that segment as needed. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

Data Dictionary



- DBA_ .. Database level information
- V\$.. Memory and process information

ORACLE

4-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Data Dictionary

One of the most important parts of an Oracle database is the data dictionary, which is a read-only set of tables that provides information about its associated database. A data dictionary contains the definitions of all schema objects in the database, such as tables, views, indexes, synonyms, sequences, procedures, packages, triggers, and so on. It also shows how much space has been allocated for, and is currently being used, by the database objects, default values for columns, integrity constraint information, and so on.

All the data dictionary tables and views for a given database are stored in the database's SYSTEM tablespace.

Not only is the data dictionary central to every Oracle database, it is an important tool for all users, from end users to application designers and database administrators. To access the data dictionary, you use SQL statements. Because the data dictionary is read-only, you can only issue queries against the tables and views of the data dictionary.

Metadata Unload

In Oracle9i, you can also use metadata unload. Metadata unload gives you the possibility to extract information from the data dictionary regarding one specific table. The result from metadata unload will be an XML document that you can use in another database to create a copy of the table.

5

Oracle9i Real Application Clusters

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

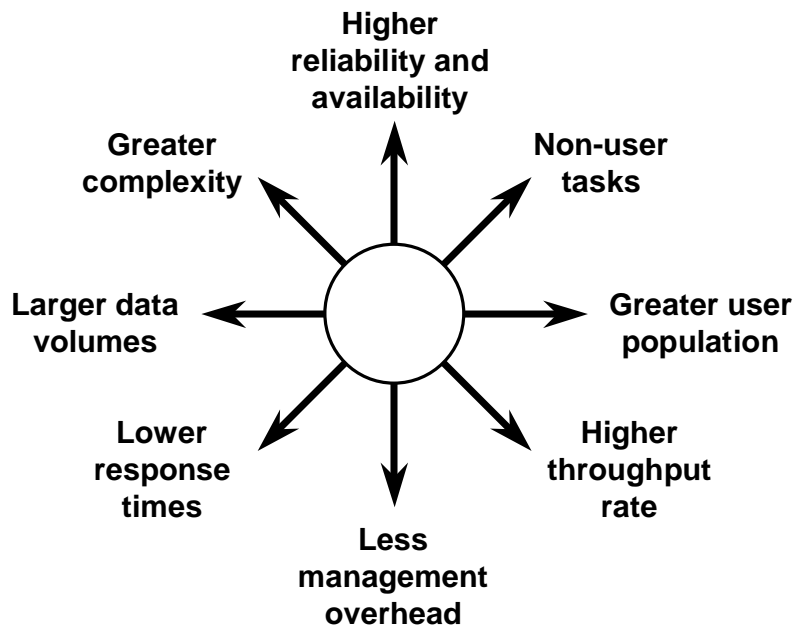
Objectives

After completing this lesson, you should be able to do the following:

- **Describe Real Application Clusters architecture**
- **Explain the functions of the Global Cache Service, the Global Enqueue Service, and the Global Resource Directory**
- **Define Cache Fusion block transfer methods**
- **Describe the shared server-side initialization parameter file**
- **Describe management tool enhancements**

ORACLE[®]

Modern Business Requirements



ORACLE

5-3

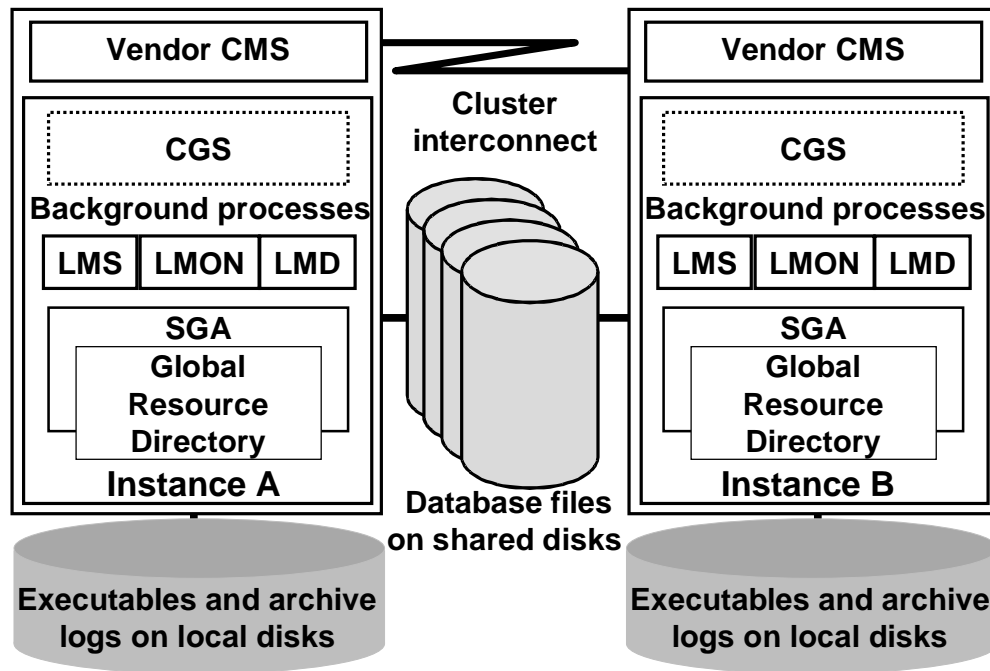
Copyright © Oracle Corporation, 2001. All rights reserved.

Modern Business Requirements

Modern business requirements have a great impact on the information technology solutions that companies need. Reliability refers to the likelihood that a component will either work or fail. In computers, it is typically measured as mean time to failure (MTTF). Availability is the measure of the amount of time a resource is available. Availability is typically measured as a percentage of time, with availability of between 90% and 99% being typical goals.

The above slide shows some of the modern businesses characteristics.

Real Application Clusters Overview



Real Application Clusters Overview

Oracle9i Real Application Clusters allow multiple instances to execute against the same database. The typical installation involves a cluster of nodes with access to a set of shared disks. A node is defined as the collection of processors, shared memory, and disks that run an instance. A node may have more than one CPU, either in an SMP or a NUMA configuration. The cluster nodes are connected by an interconnect that allows them to share disks and to communicate with each other through cluster management software (CMS).

The shared disks store the database files: data, online redo, and control files. The Oracle9i executables can also be saved on the shared disks although it is advisable to store these on each node's local disks to help ensure availability. Archived redo log files generated by each instance are stored, ideally, on each node's local disks and on a shared device used by each instance. To ensure the highest level of transfer rates for messages and data, the cluster interconnect should be based on fast interconnect technologies with low latency. This includes high-speed Ethernet and technologies based on the Virtual Interface Architecture (VIA) messaging API, and hardware, such as GigaNet and ServerNet.

Vendor-provided CMS monitors the health of processes running in the cluster, and is used by Real Application Clusters for internode messaging and to control the membership of the instances. A layer of Oracle9i software, the Cluster Group Services (CGS), provides an interface to the vendor's CMS and also performs its own instance validation checks.

Note: Other components shown in the graphic are discussed on the next few pages.

Real Application Clusters Characteristics

- Each instance is a separate System Global Area (SGA)
- Each instance writes to its own set of redo log files
- An Oracle instance can be started on each node
- All instances can read the redo log files of other instances
- All instances share the same data files and control files
- Row-level locking is preserved
- All instances can concurrently execute transactions against the same database

ORACLE

5-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Real Application Clusters Characteristics

- Each instance is a separate System Global Area (SGA) and a set of background processes.
- Each instance writes to its own set of redo log files.
- An Oracle instance can be started on each node.
- All instances can read the redo log files of other instances.
- All instances share the same data files and control files.
- Row-level locking is preserved.
- All instances can concurrently execute transactions against the same database and each instance can have multiple users executing transactions.
- Applications that access the same database can run on the same nodes as Real Application Clusters instances, or on separate nodes using a three-tier or client-server architecture.

Background Processes

Real Application Clusters background processes replace those used by Oracle Parallel Server

- **The Parallel Server background process BSP is no longer used.**
- **The following background processes have different functions:**
 - **LMS: Global Cache Service process**
 - **LMON: Global Enqueue Service monitor**
 - **LMD: Global Enqueue Service process**

ORACLE

5-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Background Processes

Although some of the background processes used by Real Application Clusters have the same names as those in Oracle Parallel Server in Oracle8i, their functions and activities are different from those in previous Oracle cluster-enabled software releases. These differences are discussed below:

- **BSP:** The Block Server Process was introduced in Oracle8i Parallel Server to support Read Consistent Cache Fusion. It does not exist in a Real Application Clusters database where these activities are performed by the LMS process.
- **LMS:** This process was known as the Lock Manager Server Process in Parallel Server. The new LMS process in Real Application Clusters executes as the Global Cache Service Process.
- **LMON:** This process was known as the Lock Manager Monitor in Parallel Server. While Real Application Clusters databases use the same process name, the process is defined as the Global Enqueue Service Monitor.
- **LMD:** There was a Lock Manager process, called LMD0, in Oracle Parallel Server, with a zero at the end of the name implying that multiple copies of the process may be allowed. This process is not used by Real Application Cluster, but a new process called LMD executes as the Global Enqueue Service.

Benefits of Real Application Clusters

- **Increased number of users**
- **Scalability**
- **Application speedup**
- **Availability**

ORACLE[®]

5-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Benefits of Real Application Clusters

By using multiple instances, each on its own node, to process against the same database, Real Application Clusters provide the following advantages over single instance databases:

- Increasing user populations can be supported by adding additional nodes when the available nodes reach their session capacity.
- Work can scale (more work can be completed in the same amount of time) when each instance can be optimized to support a maximum workload.
- Some processing, particularly operations that can be executed with parallel components, can complete faster when the work is spread across multiple nodes.
- Applications have higher availability because they can be accessed from any instance. An instance failure on one node does not prevent work from continuing on one or more surviving instances, and transparent application failover enables fast reconnections and resumption of work following an instance failure.

Global Cache Service

Instance A		Instance B		Instance C	
Global Cache Service		Global Cache Service		Global Cache Service	
Re-source	Granted to Instance	Re-source	Granted to Instance	Re-source	Granted to Instance
...
20	A,B,C	21	B	22	A,C
23	A,B	24	A,C	25	C
26	C	27	A,B,C	28	A,B,C
29	B,C	30	B,C	31	B
...

Note: For simplicity, resource status values are not shown.

ORACLE

5-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Global Cache Service

A key function of Real Application Clusters is the coordination of resources shared across instances, including block images in their buffer caches. In previous Oracle cluster-enabled software releases, this function was provided through a Distributed Lock Manager which managed block locks, called Parallel Cache Management locks, and other shared resources such as global enqueues. In Oracle9i, database block resources are managed by the Global Cache Service, using the LMS background process, and non-database block resources by the Global Enqueue Service and its LMD process.

The information tracked by the Global Cache and the Global Enqueue Services is stored in the Global Resource Directory using memory from each of the active instances. Each resource is mastered by just one instance at a time, as shown in the graphic for resources 20 through 31. When an instance needs a shared resource, it sends a message to mastering instance across the cluster interconnect using the CGS and the vendor's CMS. The Global Cache and the Global Enqueue Services also send messages to instances, for example, to request them to release resources required by other instances.

For database blocks, the Global Cache Service maintains a list of resource statuses. These resources are acquired and released based on demands of all the instances, independently of transaction and row resources. The resources tracked by the Global Cache Service can be acquired, dropped, or have their statuses converted regardless of the state of any transactions on the related block.

Dynamic Resource Remastering

- **Dynamic remastering allows resource masters to be changed without complete reconfiguration.**
- **During processing, lazy dynamic remastering moves resources to the most active instance.**
- **Should an instance leave the group, the background processes only remaster the resources from the departing instance.**
- **Similarly, when a new instance joins the group, the resources are gradually remastered, adapting to cluster workload.**

ORACLE

5-9

Copyright © Oracle Corporation, 2001. All rights reserved.

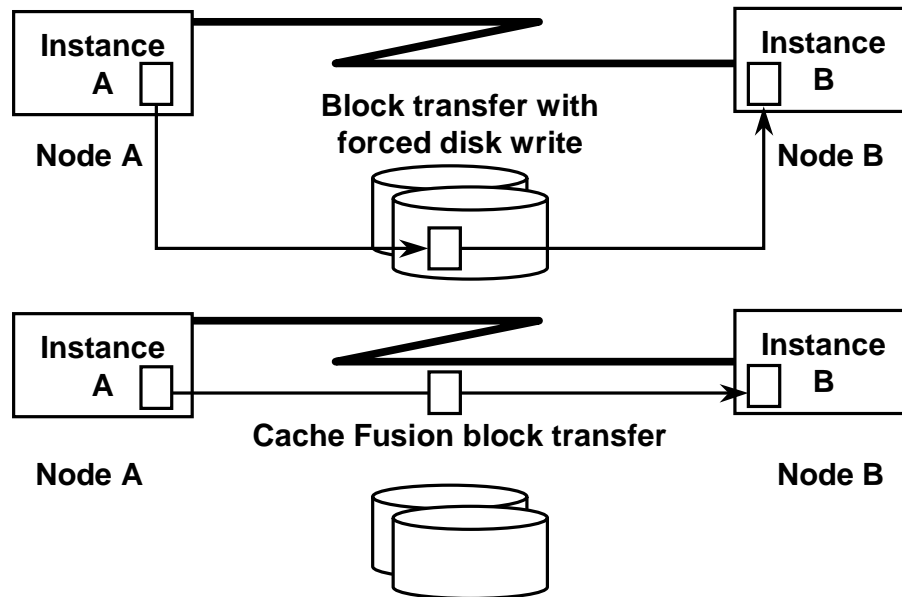
Dynamic Resource Remastering

In previous releases of Oracle cluster-enabled software, the Distributed Lock Manager lock database was distributed among the available instances. Once assigned to an instance, a lock would not move from its assigned instance until another instance started up or shut down, known as a node transition. A node transition required the lock database to be entirely rebuilt across the new set of active instances. While this rebuilding was in progress, access to the Distributed Lock Manager was frozen, disabling new request processing and reducing database availability. Further, when an instance shut down, all information from its portion of the lock database was lost.

In Real Application Clusters, the resources in the Global Resource Directory are remastered (moved to different instances) dynamically. During processing, when particular instances use blocks almost exclusively, the block resources are remastered to those instances. This remastering is done *lazily*, that is, when the resource information can be included with other messages or when the background processes have no pending requests to process, causing no obvious performance overhead.

Dynamic resource remastering also increases database availability during node transitions in Real Application Clusters. Only the resources associated with new or departing instances have to be processed. Both LMS and LMD will continue to process requests for unaffected resources as they perform this work, minimizing the performance impact of node transitions.

Block Transfers



ORACLE

5-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Block Transfers

If a block is only being queried (read) by all the instances, the same image is valid for all the instances and this image is consistent with the on-disk copy. Thus the block image can be read from disk or from another instance.

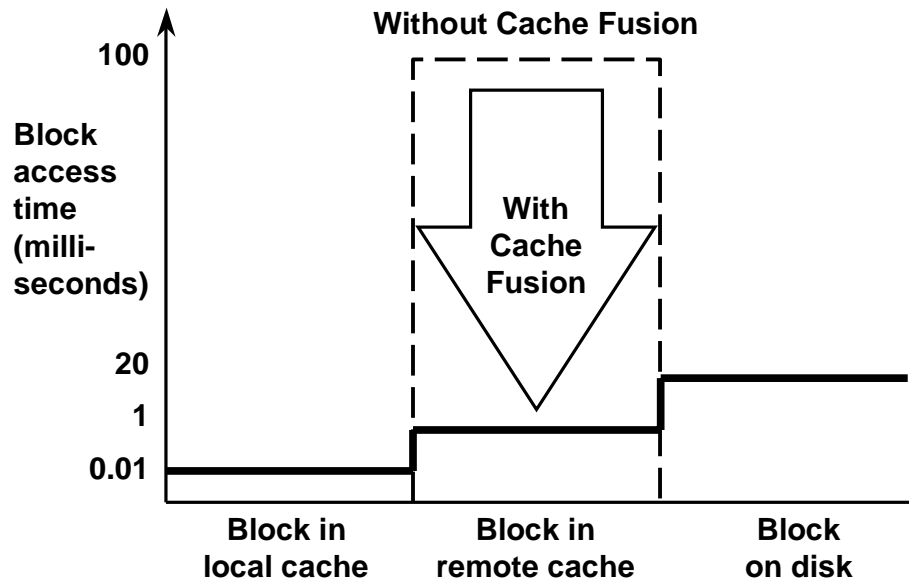
However, when a block has been changed (written) by an instance, another instance must obtain the current block image before it can make its own changes. Similarly, an instance performing a read may need to see changes committed by the writing instance. In these cases, the Global Cache Service instructs the instance holding the current block image to transfer the image to the requesting instance using one of three basic mechanisms:

Block Transfer with Forced Disk Write: This method, often referred to as a block ping, was the only block transfer method in Oracle cluster-enabled software releases prior to Oracle8, release 8.1, and was also used in Oracle8i to transfer blocks between writing instances. The block is written to disk by the holding instance, and the requesting instance reads the freshly written copy.

Consistent Read Cache Fusion: This algorithm enables the writing instance to prepare a read-consistent image in its own buffer cache and to send this image across the interconnect to the buffer cache of the instance performing the query. A similar algorithm was used in Oracle8i Parallel Server databases.

Cache Fusion Block Transfer: Exclusive to Real Application Clusters, this algorithm passes either clean or dirty block images between instances across the cluster interconnect.

Benefits of Cache Fusion



ORACLE

5-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Benefits of Cache Fusion

By allowing any block being held by an instance to be copied to another instance across the cluster interconnect, Cache Fusion allows data to be shared between instances without the overhead of forced disk write block transfers. It even allows an instance to obtain a block already cached in another instance faster than it could read that same block from disk.

Studies on typical clusters with high-speed interconnects regularly demonstrate the levels of speedup shown in the graph. This, in turn, allows applications to run on Real Application Clusters that might not have scaled successfully in previous Oracle cluster-enabled software releases.

High Availability Features

- **Dynamic resource remastering by the Global Cache Service**
- **Concurrent Global Cache Service remastering and instance recovery**
- **CGS contains its own instance validity checking**
- **Use of past images and two-pass redo log recovery**
- **Enhancements to Real Application Clusters Guard (the replacement for Oracle Parallel Fail Safe)**

ORACLE

5-12

Copyright © Oracle Corporation, 2001. All rights reserved.

High Availability Features

Real Application Clusters is Oracle Corporation's premier high availability solution. You can use it to configure alternate instances on a database to be used if the active instances fail due to software or hardware errors. High availability features are incorporated into many of components of the architecture.

- Global Cache Service reconfiguration, discussed earlier, minimizes down time because only resources for one instance need to be remastered after a node transition.
- Due to the relatively small number of resources to be remastered following instance failure, instance recovery by a surviving instance on behalf of the failed instance progresses concurrently with Global Cache Service remastering. In Oracle Parallel Server, instance recovery was deferred during lock database reconfiguration.
- The CGS software contains its own instance status detection algorithms. This enables active instances to monitor each other without relying on the vendor's CMS. Therefore problems are detected quickly, allowing Global Resource Directory reconfiguration and instance recovery to proceed with minimal interruption to service.
- The use of Cache Fusion past images to refresh data files prior to recovery and Oracle9i two-pass redo log processing enable faster recovery.
- Real Application Clusters Guard, the Oracle9i replacement for Oracle Parallel Fail Safe, works with more vendor clusters than its predecessor and has improved functionality.

Real Application Clusters Guard

- **High availability database solution for mission critical systems**
- **An integrated feature of Oracle9i Real Application Clusters**
- **Incorporates technology from Oracle9i and the vendor's CMS:**
 - **Primary and secondary instance configuration**
 - **Self-contained software packs**
 - **Configuration and management tools**

```
ACTIVE_INSTANCE_COUNT = 1
```

ORACLE

5-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Real Application Clusters Guard

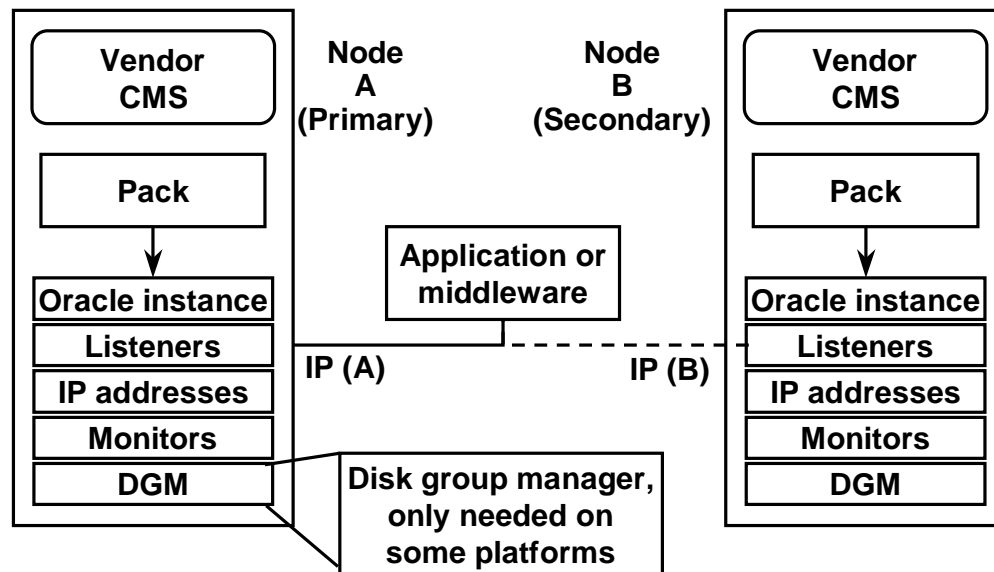
Real Application Clusters Guard provides a high-availability option that minimizes down time in the event of instance or node failure for critical applications. Real Application Clusters Guard combines features of Real Application Clusters and the vendor's CMS. A similar product, Oracle Parallel Fail Safe, was available in earlier releases but only ran on a limited set of clusters (depending on the specific release level) and required a separate installation. In Oracle9i, Real Application Clusters Guard supports a larger variety of clusters and is automatically installed with the Real Application Clusters option.

The typical configuration for Real Application Clusters Guard is a two instance Real Application Clusters database with `ACTIVE_INSTANCE_COUNT = 1` defined in their parameter files. This enables one instance as the primary instance and one as the secondary instance. The primary instance masters the entire Global Resource Directory and is the only instance to allow user connections through Oracle Net Services. The secondary instance takes over the primary role when the primary instance fails.

In addition, the pieces of software necessary to manage an instance are configured into packs. Each pack is a self-contained set of software that can enable and monitor all the components of a Real Application Clusters Guard instance on a node.

The packs and other components of Real Application Clusters Guard are set up and configured with a platform-specific tool. The tool also simplifies deployment of changes in your Real Application Clusters Guard environment.

Real Application Clusters Guard Architecture



ORACLE

5-14

Copyright © Oracle Corporation, 2001. All rights reserved.

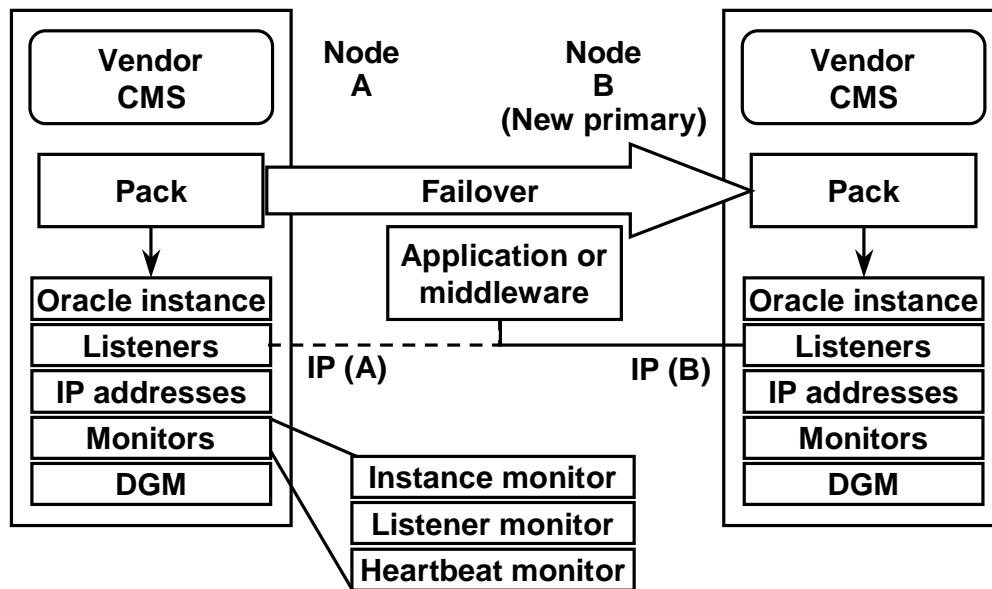
Real Application Clusters Guard Architecture

The graphic shows a two-node Real Application Clusters Guard configuration. Each node executes the vendor's CMS, which in addition to its normal functions (covered earlier in the lesson), is responsible for running and halting scripts automatically upon failover or when you issue the appropriate command.

Each node also contains a pack of software. A pack supports a single instance, with access provided through listeners. A pack controls startups, shutdowns, and restarts of the processes under its control. Packs contain the following components:

- A Real Application Clusters instance: In the example shown, Node A is running with the primary instance role and Node B with the secondary instance role.
- One or more listeners to accept Oracle Net connection requests: Public listeners support clients and private listeners support tools such as Oracle Enterprise Manager and Recovery Manager and also provide access for database administration tasks. In the example, the public listener points to the IP address of Node A, the primary node.
- One or more IP addresses: Public IP addresses are relocatable and can be moved between nodes to maintain availability to an active instance. Private IP addresses are static and support connections to private listeners.
- Three monitors: Discussed on the next slide.
- A disk group manager (DGM): Required only on some platforms, it enables public access to the database disks by the current primary node.

Monitors and Failover



ORACLE

5-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Instance Monitor

The instance monitor detects termination of the local instance and initiates failover to the secondary node or restarts the instance.

Listener Monitor

The listener monitor checks and restarts the listeners on its own node. When the public listener fails to restart, the listener monitor exits, initiating a halt script. Oracle Real Application Clusters Guard either begins failover or restarts the primary instance, depending on the state of the secondary node.

Heartbeat Monitor

The heartbeat monitor checks the availability of the Oracle instance. During normal operation, the heartbeat monitor on each instance updates its own local heartbeat and checks the heartbeat of the other instance. The heartbeat monitor on the primary instance also executes a customer-defined query to test whether the primary instance is capable of work. The local Oracle instance is considered unavailable if the heartbeat monitor fails to complete three consecutive attempts and there are no unusual circumstances, such as instance recovery or unusually large numbers of sessions logging on.

The heartbeat monitor also initiates one kind of failover action: If the primary instance is unavailable and the primary instance role has not resumed normal function on its new node, then the heartbeat monitor initiates takeover. A takeover occurs when the secondary node executes failover of the primary instance role to itself.

Shared Initialization Parameter File

- **Parameters for different instances can be mixed in a single initialization parameter file.**
 - Only one file must be maintained and propagated.
 - 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

5-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Shared Initialization Parameter File

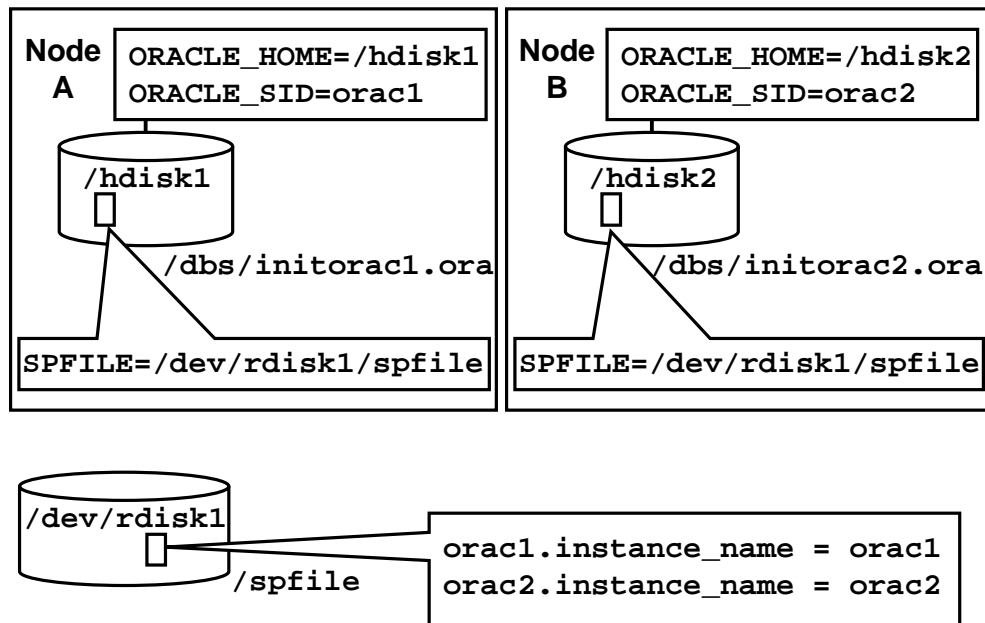
In previous Oracle cluster-enabled software releases, you needed a separate initialization file to assign different parameter values to each instance of your database. At the same time, you had to provide identical values for some parameters across all instances. This required you to maintain multiple parameter files, typically using the IFILE parameter to link your 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 only have to maintain one file. It is also easier to avoid making mistakes, such as changing a value in one file but not another, with all the parameters stored in one place.

To allow different instances to share the same initialization file, parameter entries 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 could include the following entries in your common parameter file:

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

Shared Server-Side Parameter File



ORACLE

5-17

Copyright © Oracle Corporation, 2001. All rights reserved.

Shared Server-Side Parameter File

When you put the parameters for all your instances in a single initialization file, this file must be available to the process that starts up each instance. If your instances are started automatically as part of the system startup routines, you would need a copy of the file on each node in the cluster. However, you can take advantage of a server side parameter files (SPFILE) in your Real Application Clusters database. 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 shared disk, typically in a raw partition.

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

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

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

That 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 in these parameters follow the recommended naming convention for instances, that is, they are the same as the SID:

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

Real Application Clusters and OEM

Oracle Enterprise Manager enhancements include:

- **Redo log assignments**
- **Targeted reporting for databases and instances**
- **Extended wizards and tools for cluster databases**
- **Monitoring and event enhancements**
- **Managing Export, Import, load, backup, and analyzing tasks**

ORACLE

5-18

Copyright © Oracle Corporation, 2001. All rights reserved.

Real Application Clusters and Oracle Enterprise Manager (OEM)

The following enhancements are included in OEM to support Real Application Clusters:

- **Redo Log Assignment:** Assigns redo log groups to specific threads
- **Report Generation:** Provides targeted reporting for cluster databases and instances
- **Wizards and Tools:** Extends tools, such as Database wizards, Application Management, Change Management, Database Applications, Diagnostic Pack, and Tuning Pack, for cluster databases
- **Events:** Enhances monitoring and events available with OEM and performance packs
- **Jobs:** Provides data management (Export, Import, Load) backup management, and analyzes tasks for cluster database and instance targets

OEM Instance Management Provisions

OEM in Oracle9i provides new instance level management:

- **SPFILE handling**
- **Stored configuration**
- **Session handling**
- **Lock details**
- **Resource monitoring**

ORACLE

5-19

Copyright © Oracle Corporation, 2001. All rights reserved.

OEM Instance Management Provisions

These single instance management capabilities were not supported by OEM in clustered databases until Oracle9i:

- **SPFILE Handling:** View and update a server side initialization parameter file (SPFILE)
- **Stored Configuration:** Create, edit, and store multiple startup configurations for each cluster database instance. This eliminates the need to track initialization parameter files for each instance.
- **Session Handling:** For each cluster database instance, list the status of connected users, view the latest SQL executed by a session, and kill an unwanted session.
- **Lock Details:** For each cluster database instance, list details for currently held user locks (SQL DML enqueue, transaction enqueue, and row level locks) and system locks
- **Resource Monitoring:** For the cluster database, create and modify resource consumer groups and define, modify, and activate resource plans. For each cluster database instance, provide performance statistics of active resource plans

6

User and Data Security Management

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

After completing this lesson, you should be able to explain how to control access to an Oracle9i database using:

- **Database users and security domains**
- **User authentication**
- **Tablespace quotas and user resource limits**
- **Password management and profiles**
- **Privileges and role-based security**

ORACLE®

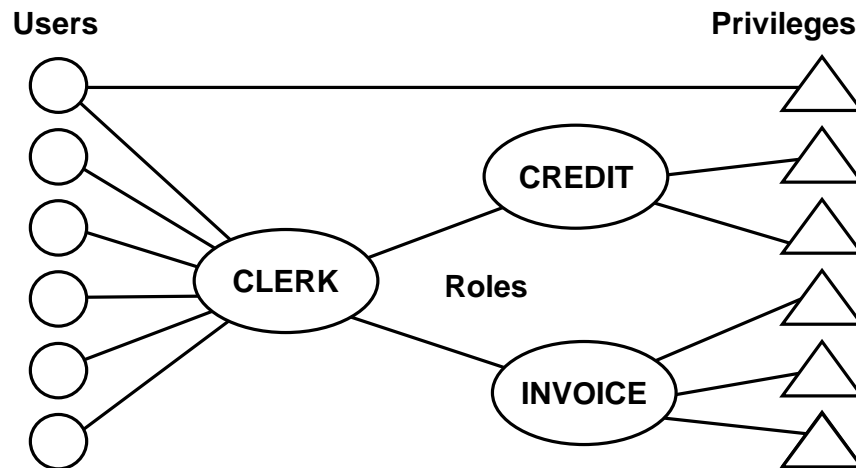
Objectives

After completing this lesson, you should be able to explain how to control access to an Oracle9i database using:

- **LDAP integration**
- **Virtual Private Database**
- **Database resource management**
- **Advanced Security Option (ASO)**
- **Auditing**

ORACLE

Privileges and Role-Based Security



ORACLE

6-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Privileges and Role-Based Security

Each Oracle database has a list of usernames. To access a database, a user must use a database application and attempt a connection with a valid username of the database. Each username has an associated password to prevent unauthorized use.

Each user has a security domain, which is a set of properties that determine the actions (such as privileges and roles) available to the user, tablespace quotas (available disk space) for the user, and system resource limits (for example, CPU processing time) for the user.

A privilege is a right to execute a particular type of SQL statement. Some examples of privileges include the right to connect to the database (create a session), the right to create a table in your schema, the right to select rows from someone else's table, and the right to execute someone else's stored procedure.

A role is a database object that can be granted to a user and to which you can grant privileges and other roles. For increased security, you can make sure that only authorized programs can switch on a role, by using a password or what is called an application role.

Privileges are granted to users or PUBLIC (all users) so that users can access and modify data in the database. You can grant privileges to users explicitly or to roles (a named group of privileges), and then the role can be granted to one or more users. Because roles allow for easier and better management of privileges, privileges are normally granted to roles and not to specific users.

Secure Application Role

- **Solves the problem of preventing unauthorized access to data through other client programs**
- **Enabling a role is checked through a package, not a password**
- **Uses the notion of application context**

ORACLE®

6-5

Copyright © Oracle Corporation, 2001. All rights reserved.

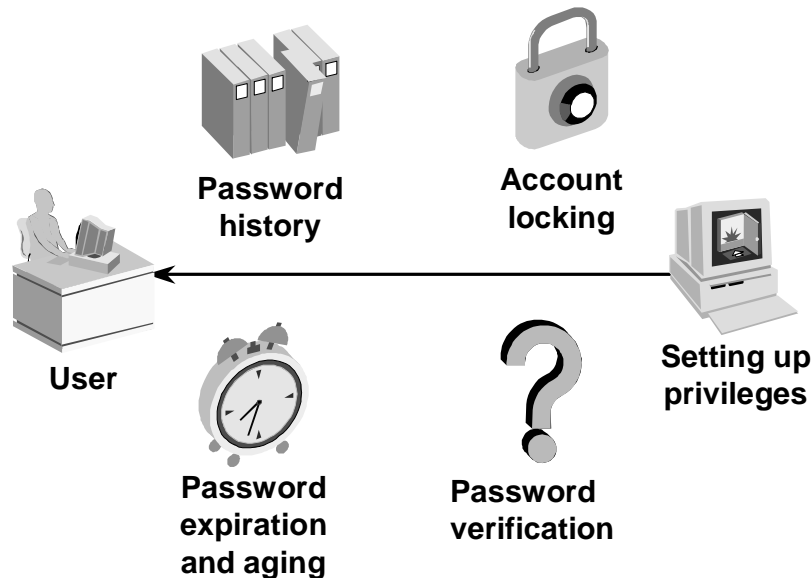
Secure Application Role

With the secure application role, you can control a user's ability to enable a role. A procedure is run every time someone tries to enable the role. Developers can add logic in this procedure to ensure that the role can be enabled by only one specific program.

The method used prior to Oracle9i consisted of using password authentication of the role, and embedding and hiding the password in the application. The weakness in that method was that if the password was known, any application could access the data.

With the secure application role, your application accesses a context containing specific attributes. These attributes are checked by the procedure, which ultimately decides to accept or reject your request to enable the role.

Database Password Management



ORACLE

6-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Authentication by the Oracle Database

The Oracle Database can authenticate users attempting to connect to a database by using information stored in that database. When the Oracle Database uses database authentication, you create each user with an associated password. A user provides the correct password when establishing a connection to prevent unauthorized use of the database. The Oracle Database stores a user's password in the data dictionary in an encrypted format. A user can change his or her password at any time.

To protect password confidentiality, Oracle allows you to encrypt passwords during network (client/server and server/server) connections. If you enable this functionality on the client and server machines, Oracle Net encrypts passwords using a modified data encryption standard (DES) algorithm before sending them across the network.

The Oracle Database can lock a user's account if the user fails to log in to the system within a specified number of attempts. Depending on how the account is configured, it can be unlocked automatically after a specified time interval or it must be unlocked by the database administrator.

Authentication by the Oracle Database (Continued)

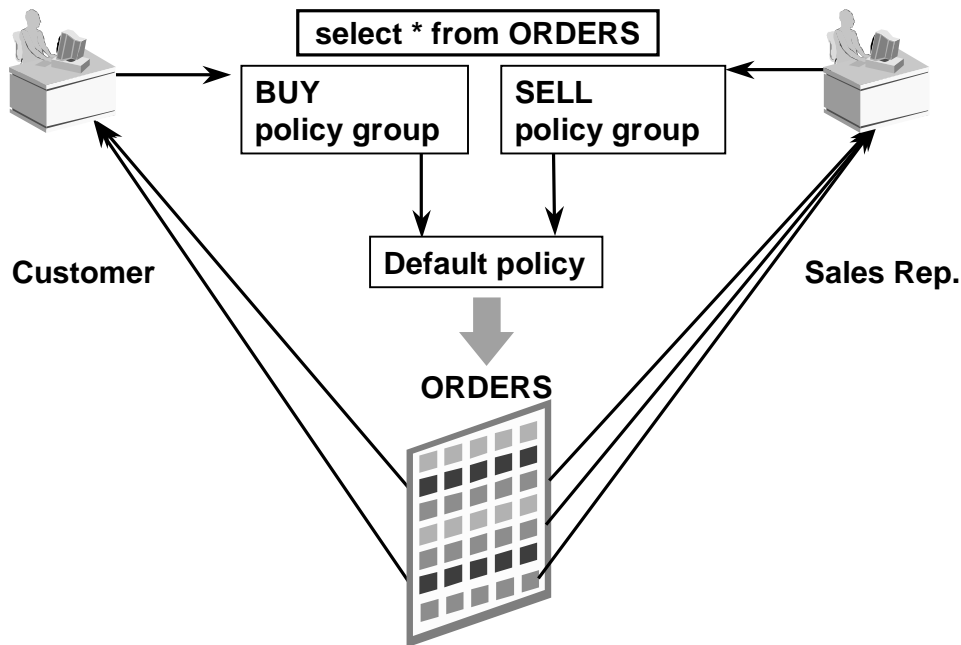
Password lifetime and expiration options allow the database administrator to specify a lifetime for passwords, after which time they expire and must be changed before a login to the account can be completed. On first attempt to log in to the database account after the password expires, the user's account enters the grace period, and a warning message is issued to the user every time the user tries to log in, until the grace period is over.

The password history option checks each newly specified password to ensure that a password is not reused for the specified amount of time or for the specified number of password changes.

Complexity verification checks that each password is complex enough to provide reasonable protection against intruders who try to break into the system by guessing passwords. The Oracle server default password complexity verification routine requires that each password:

- be a minimum of four characters in length
- not equal the `userid`
- include at least one alphabet character, one numeric character, and one punctuation mark
- not match any word on an internal list of simple words, such as *welcome*, *account*, *database*, *user*, and so on
- differ from the previous password by at least three characters

Virtual Private Database



6-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Virtual Private Database

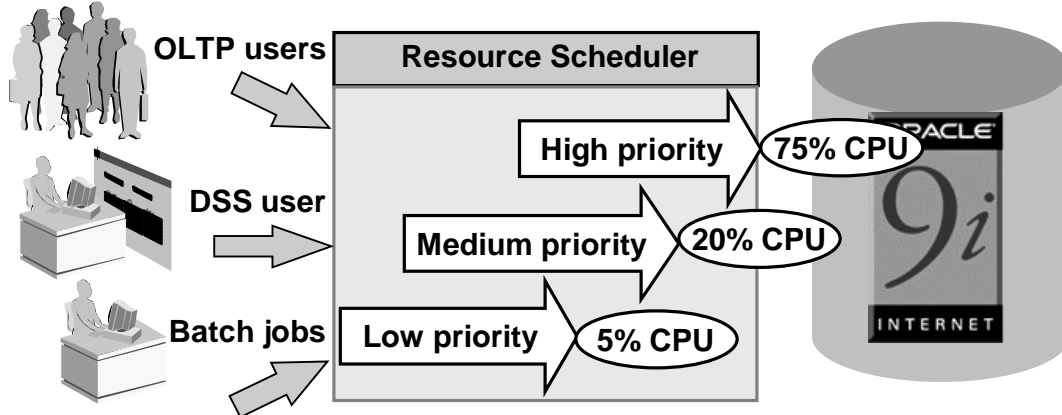
The Virtual Private Database is a combination of two features: fine-grained access control and application context. Within a single database, it enables per-user or per-customer data access with the assurance of physical data separation. In the above example, both the sales representative and the customer are executing the same query against the same **ORDERS** table, and yet the results returned to each of them is completely different. The result set for each is limited to *only his own data*. For the sales representative, the result set includes only *his* customers' orders, but no other representatives' customers. The customer, of course, only sees his own orders. This is all done by the data server itself, and is transparent to the user.

The implementation of fine-grained access control is through dynamic query modification. When a user accesses an object (directly, or through a subquery) that has a security policy attached to it, the Oracle Database automatically consults the package that implements the policy for that view or table. The policy returns a predicate (access condition) that is appended to the query. The statement is then parsed, optimized, and executed. Here, *query* can mean query for update, insert, or delete, as well as **SELECT** statements.

If there are multiple policies attached to a table, the data server the predicates returned by each policy. This is not desirable if the table is to be used by users from separate groups with different FGACs. Policy groups are used to distinguish policies among different applications.

Resource Management

- **Allocate system resources to (groups of) users**
- **Manage mixed workloads and control system performance**



6-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Resource Management

Database resource management provides a DBA with the ability to control and limit the total amount of processing resources available to a given user or set of users.

Resources are allocated to users based on a plan that is specified by the DBA. Users, and groups of users, are assigned to resource consumer groups. A resource plan specifies how resources are distributed among the different resource consumer groups. Plans can be dynamically altered on a production database without quiescing the system. This allows alternate plans for day time, night time, weekends, quarter end, or other times that require a different set of priorities.

The resource manager is fully integrated into the database security system. User sessions can switch resource consumer groups to increase execution priority, if the user has been granted the privilege to switch consumer groups. Users can also be moved from group to group by the administrator on a production system, thereby dynamically changing how CPU resources are used. The DBA can also set up the resource management system to switch the group for a user automatically, depending on the type of query the user is doing. These capabilities allow very simple, yet powerful allocation policies to be implemented for database resources.

Database Resource Manager

The following resources can be handled:

- **CPU time**
- **Parallel degree**
- **Undo quota**
- **Maximum execution time for a query**
- **Maximum number of active sessions**

ORACLE

6-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Database Resource Manager

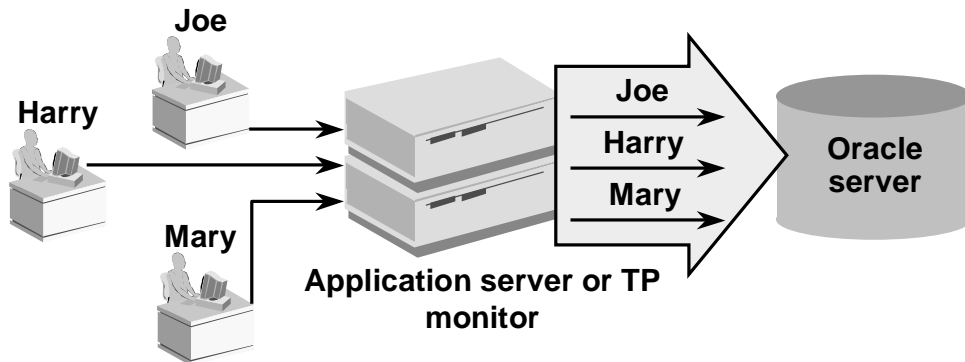
The following resources can be handled:

- CPU time
- Parallel degree
- Undo quota
- Maximum execution time for a query
- Maximum number of active sessions

Resource manager can also be used to quiesce the database. This halts all activity in the database without disconnecting the users, while the DBA performs maintenance operations.

N-Tier Authentication

- **Maintain the identity of the client through all tiers of the connection**
- **Limit privilege of a middle tier**



ORACLE

6-11

Copyright © Oracle Corporation, 2001. All rights reserved.

N-Tier Authentication

To provide scalability, an application server or TP monitor can be used to perform transactions on behalf of clients that are accessing the application server through the network.

The application server will perform the transactions, but there must be a mechanism that ensures that the real connected client is still the identity used in the database.

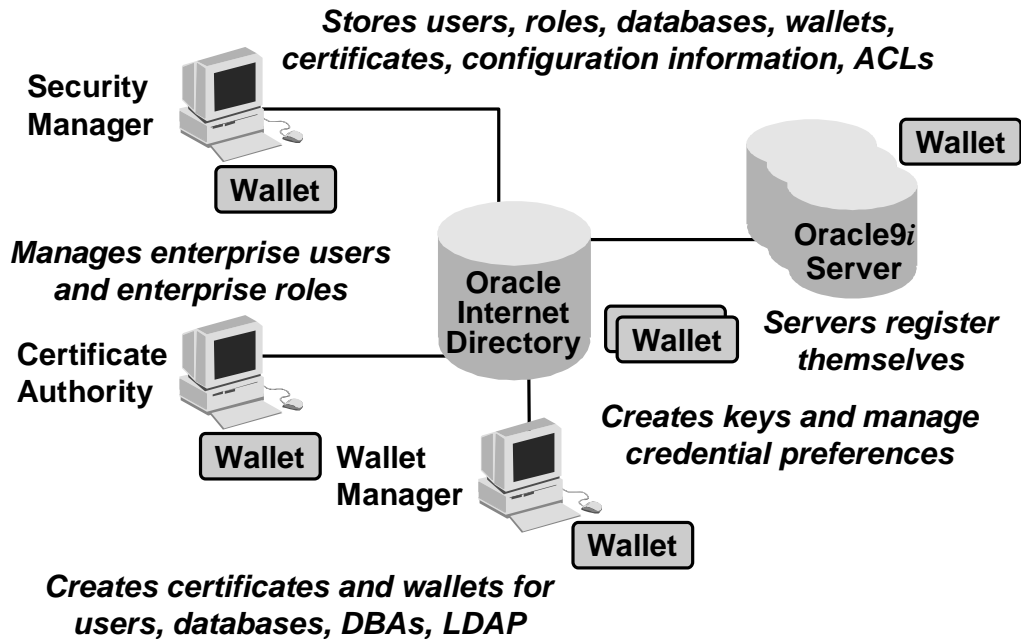
The client may authenticate itself to the middle tier using any authentication method the middle tier accepts. The middle tier may authenticate itself to the database using any method the database accepts.

In the Oracle9i server, the middle-tier does not have to reauthenticate the client to the back-end server. This requires less overhead, and is also useful for cases in which you cannot reauthenticate the client. For example, a client using an X.509 certificate cannot be authenticated through a middle tier to the server, because that would require the client to give up his private key to the middle tier, which is completely insecure.

The database can trust the middle tier because the following is verified:

- that the middle tier IS the middle tier (through authentication)
- that the client for whom a connection is established exists in the database
- that the middle tier is privileged to connect on behalf of that user, with the appropriate roles for that user

Oracle9i Enterprise User Management



ORACLE

6-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i Enterprise User Management

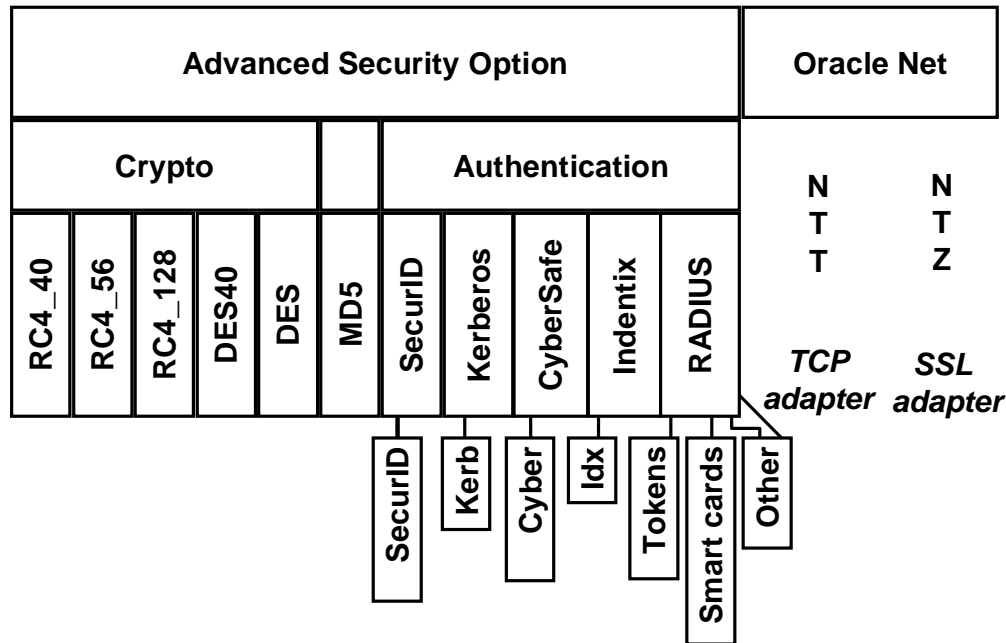
An Oracle wallet contains the user's certificate, private key, and a set of trust points (certificate authorities). A wallet can be stored on disk, or downloaded from an LDAP (Lightweight Directory Access Protocol) directory. Wallets are stored encrypted with a user password.

Initial integration will support Oracle Internet Directory, an LDAPv3-compliant directory serve; however, its intent is to be interoperable with other LDAP-compliant directories. LDAP integration in the Oracle9i server is required so that the database can get authentication information for the user attempting to connect, as well as authorizations, and potentially to set up the proper security context for that user. This integration gives a full solution regarding enterprise management and security/directory integration.

Single sign-on is a very useful feature, which allows a user to log on to a client machine and then access all the databases in the enterprise without further explicit authentication. The Oracle server supports single sign-on in multiple ways:

- Through network authentication services supported by the Advanced Security Option through Oracle Net
- Through X.509 certificates loaded into an Oracle Wallet and stored in an LDAP directory, over SSL-enabled Oracle Net
- Through native naming authentication adapters (for example, Windows NT)

Authentication Architecture



ORACLE

6-13

Copyright © Oracle Corporation, 2001. All rights reserved.

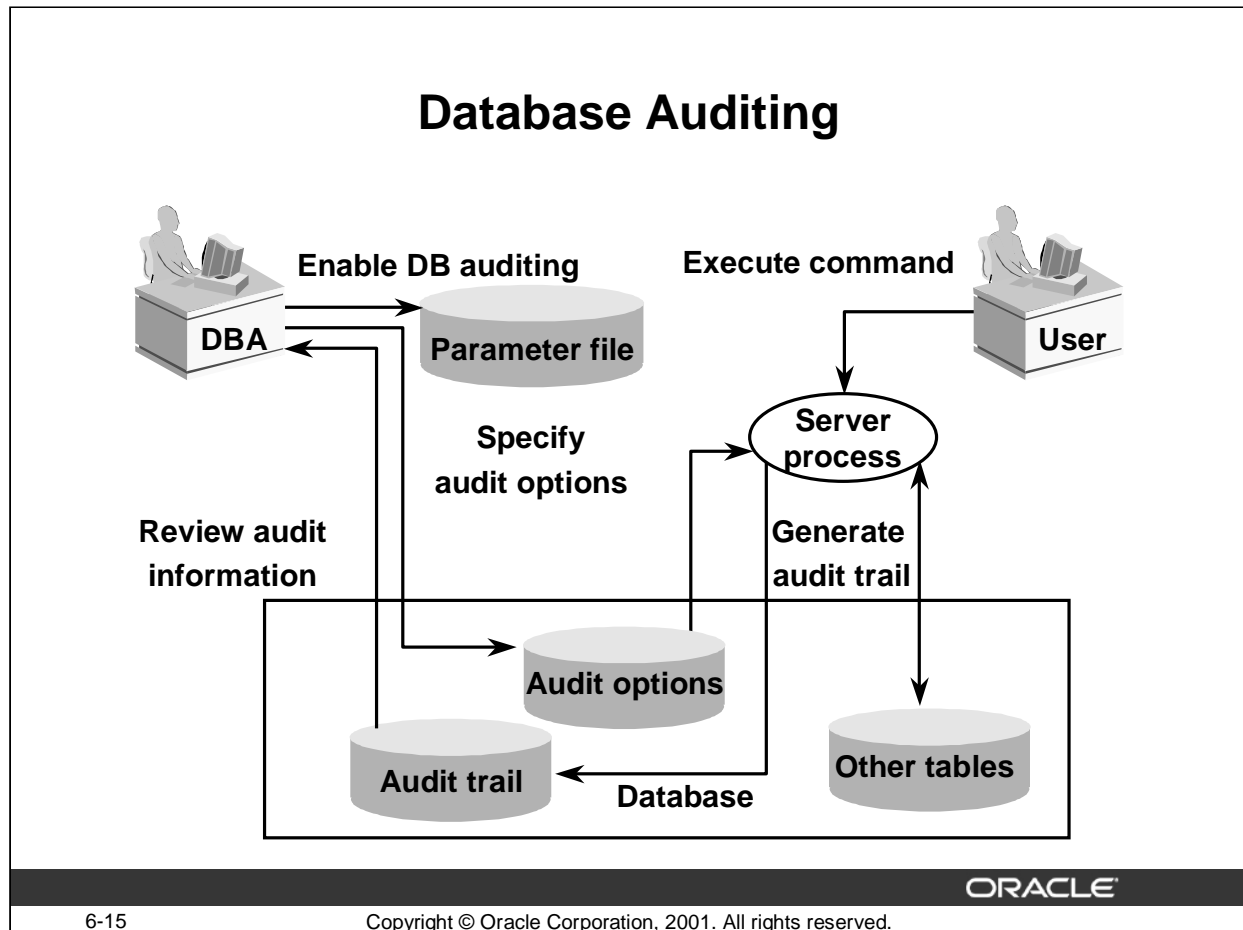
Authentication Architecture

Oracle supports several methods of authentication by the network:

- **Third Party-Based Authentication Technologies:** If network authentication services, such as DCE, Kerberos, or SESAME are available to you, the Oracle9i server can accept authentication from the network service. To use a network authentication service with the Oracle server, you need Oracle9i Enterprise Edition with the advanced security option (ASO).
- **Public Key Infrastructure-Based Authentication:** Authentication systems based on public key cryptography systems issue digital certificates to user clients, which use them to authenticate directly to servers in the enterprise without direct involvement of an authentication server. Oracle provides a public key infrastructure (PKI) for using public keys and certificates. It consists of the following components:
 - Authentication and secure session key management using Secure Sockets Layer (SSL)
 - Oracle Call Interface (OCI) and PL/SQL functions to sign user-specified data using a private key and certificate, and to verify the signature on data using a certificate and trust point
 - Oracle wallets, which are data structures that contain a user private key, a user certificate, and a set of trust points (the list of root certificates the user trusts)

Authentication Architecture (continued)

- Oracle Wallet Manager, which protects user keys and manages X.509v3 certificates on Oracle clients and servers
- X.509v3 certificates, which you obtain from a certificate authority (outside of Oracle). The certificates are loaded into Oracle wallets to enable authentication.
- Directory-enabled Oracle Security Manager, which provides centralized privilege management to make administration easier and increase your level of security. Directory-enabled Oracle Security Manager allows you to store and retrieve roles from Oracle Internet Directory or any directory compliant with the Lightweight Directory Access Protocol (LDAP).
- Oracle Internet Directory, which is an LDAP v3-compliant directory. It allows you to manage the user and system configuration environment, including security attributes and privileges, for users authenticated using X.509 certificates. Oracle Internet Directory enforces attribute-level access control, allowing the directory to restrict read, write, or update privileges on specific attributes to specific named users (for example, an enterprise security administrator).
- Remote Authentication: Oracle supports remote authentication of users through Remote Dial-In User Service (RADIUS), a standard lightweight protocol used for user authentication, authorization, and accounting. To use remote authentication of users through RADIUS with Oracle, you need Oracle9i Enterprise Edition with the Advanced Security option.



Database Auditing

Auditing is the monitoring and recording of selected user database actions. You can use it to:

- Investigate suspicious activity
- Monitor and gather data about specific database activities

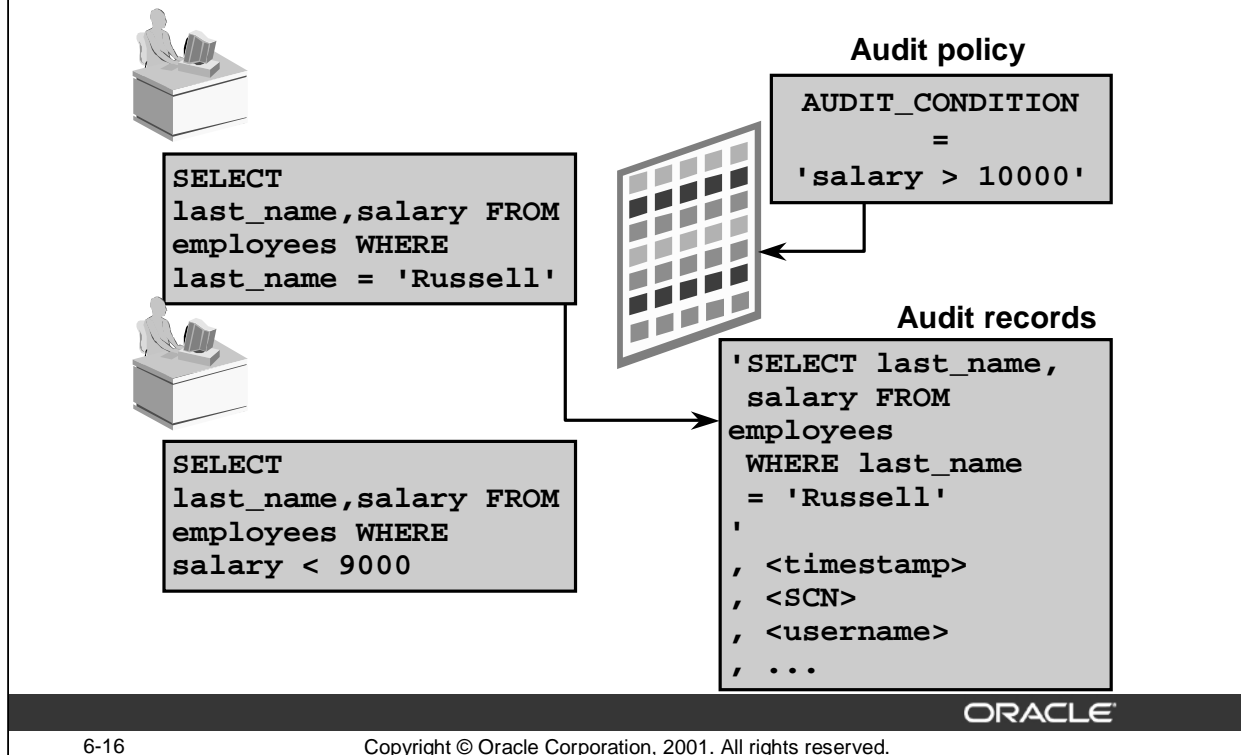
Oracle supports several general types of auditing:

- **Statement auditing:** The selective auditing of SQL statements with respect to only the type of statement, not the specific schema objects on which it operates. Statement auditing options are typically broad, auditing the use of several types of related actions per option. For example, `AUDIT TABLE` tracks several DDL statements regardless of the table on which they are issued.

You can set privilege auditing to audit a selected user or every user in the database.

- **Privilege auditing:** The selective auditing of the use of powerful system privileges to perform corresponding actions, such as `AUDIT CREATE TABLE`. Privilege auditing is more focused than statement auditing because it audits only the use of the target privilege. You can set privilege auditing to audit a selected user or every user in the database.
- **Schema object auditing:** The selective auditing of specific statements on a particular schema object, such as `AUDIT SELECT ON EMP`. Schema object auditing is very focused; it audits only a specific statement on a specific schema object. Schema object auditing always applies to all users of the database.

Fine-Grained Auditing



Fine-Grained Auditing

The Fine-Grained Audit (FGA) mechanism addresses a more granular level of auditing. The new auditing policy is based on simple user-defined SQL predicates on table objects as conditions for selective auditing. The predicate can specify auditing when a specified value is retrieved.

In addition to value based, there are also cases where the administrators are only interested in whether a certain column is being referenced or accessed. Because auditing is supported whenever a column is selected in any part of a DML statement, Oracle will audit the query.

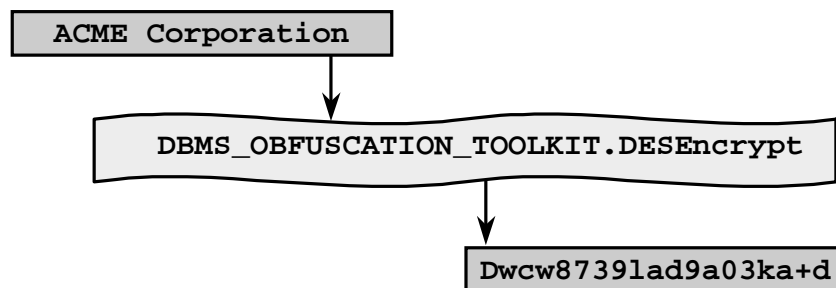
Fine-grained auditing is available only on SELECT statements with a WHERE clause, and for one audit column only.

The triggering event above was not that the user accessed employee records for which salaries are greater than 10,000, but that the salary was actually returned to the user. A record is written to the audit trail that includes the complete SQL statement the user submitted, a timestamp, username, and other information.

Fine-grained auditing does not automatically capture the values returned to the user. However, you may be able to use fine-grained auditing in conjunction with Flashback Query to re-create the records returned to the user.

Obfuscation Toolkit

- Package to encrypt and decrypt data
- Supports DES, 3DES, MD5



ORACLE

6-17

Copyright © Oracle Corporation, 2001. All rights reserved.

Obfuscation Toolkit

Among other security technologies, Oracle protects data in eBusiness systems through strong, standards-based encryption. Oracle has supported encryption of network data through Oracle Advanced Security (formerly known as "Secure Network Services", and then "Advanced Networking Option") since Oracle7.

The Oracle9i server also supports protection of selected data by means of encryption within the database.

To address the need for selective data encryption, the Oracle9i server provides a PL/SQL package to encrypt and decrypt stored data. The package, DBMS_OBFUSCATION_TOOLKIT, supports bulk data encryption using the Data Encryption Standard (DES) algorithm, and includes procedures to encrypt and decrypt using DES. In addition to single DES, the DBMS_OBFUSCATION_TOOLKIT supports triple DES (3DES) encryption, in both two and three key modes, for those who demand the strongest commercial available level of encryption. The toolkit also supports the MD5 secure cryptographic hash to ensure data integrity, and a random number generator for generating secure encryption keys.

Oracle Label Security

- **Out-of-the-box VPD, fine grain access control:**
 - Label-based access control
 - Based on government and defense labeling
 - Designed for commercial usage
- **No coding required**
- **Quickly implement sophisticated fine grained access control**

ORACLE

6-18

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Label Security

The Virtual Private Database technology in Oracle9i is the foundation for a new product from Oracle called Oracle Label Security. Oracle Label Security is based on labeling concepts used by government and defense organizations to protect sensitive information and provide data separation. In these environments, labels are typically composed of a hierarchical level and one or more categories or compartments. Oracle Label Security can be used to retrofit existing applications for the ASP model. It can also be used to provide security in healthcare and CRM applications.

Oracle Label Security offers the following benefits:

- Oracle Policy Manager, a GUI for administering labels and authorizations. Oracle Policy Manager will also serve as the administration tool for user-defined Virtual Private Database policies.
- Oracle Label Security automatically provides out-of-the-box VPD enforcement. Customers no longer need to code a VPD application.
- Provides data labeling required by government and defense organizations on standard operating system platforms
- Label-based access control for application service providers. Data labels provide a degree of granularity which cannot be easily achieved with raw application data.



ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

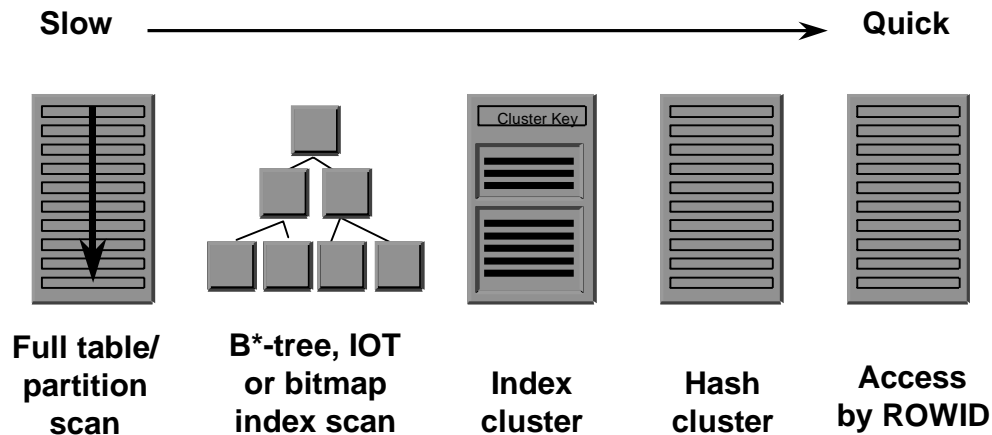
Objectives

After completing this lesson, you should be able to discuss the following aspects of the Oracle9i optimizer:

- **Access paths**
- **Execution plans**
- **SQL Trace and TKPROF**
- **Cost-based and rule-based optimization**
- **Statistics management**
- **Optimizer plan stability**
- **Cursor sharing**

ORACLE

Access Paths



Access Paths

Optimization is the process of choosing the most efficient way to execute a SQL statement. This is an important step in the processing of any data manipulation language (DML) statement: SELECT, INSERT, UPDATE, or DELETE. Many different ways to execute a SQL statement often exist, for example, by varying the order in which tables or indexes are accessed. The procedure Oracle uses to execute a statement can greatly affect how quickly the statement executes. A part of Oracle called the optimizer calculates the most efficient way to execute a SQL statement. The optimizer evaluates many factors to select among alternative access paths. It can use a cost-based or rule-based approach.

You can influence the optimizer's choices by setting the optimizer approach and goal and by gathering statistics for cost-based optimization. Sometimes the application designer, who has more information about a particular application's data than is available to the optimizer, can choose a more effective way to execute a SQL statement. The application designer can use hints in SQL statements to specify how the statement should be executed.

One of the most important choices the optimizer makes when formulating an execution plan is how to retrieve data from the database. For any row in any table accessed by a SQL statement, there may be many access paths by which that row can be located and retrieved. The optimizer chooses one of them.

Full Table Scans

A full table scan retrieves rows from a table. To perform a full table scan, Oracle reads all rows in the table, examining each row to determine whether it satisfies the statement's `WHERE` clause. The Oracle scan can be performed very efficiently using multiblock reads.

Sample Table Scans

A sample table scan retrieves a random sample of data from a table. This access method is used when the statement's `FROM` clause includes the `SAMPLE` option or the `SAMPLE BLOCK` option. To perform a sample table scan when sampling by rows (the `SAMPLE` option), Oracle reads a specified percentage of rows in the table and examines each of these rows to determine whether it satisfies the statement's `WHERE` clause. To perform a sample table scan when sampling by blocks (the `SAMPLE BLOCK` option), Oracle reads a specified percentage of the table's blocks and examines each row in the sampled blocks to determine whether it satisfies the statement's `WHERE` clause.

Table Access by Row ID

A table access by row ID also retrieves rows from a table. The row ID of a row specifies the datafile and data block containing the row and the location of the row in that block. Locating a row by its row ID is the fastest way for Oracle to find a single row. To access a table by row ID, Oracle first obtains the row ID of the selected rows, either from the statement's `WHERE` clause or through an index scan of one or more of the table's indexes. Oracle then locates each selected row in the table based on its row ID .

Index Scans

An index scan retrieves data from an index based on the value of one or more columns of the index. To perform an index scan, Oracle searches the index for the indexed column values accessed by the statement. If the statement accesses only columns of the index, Oracle reads the indexed column values directly from the index, rather than from the table.

Skip Scan on Indexes

This new index scan in Oracle9i makes it possible to use a composite index even if the leading column of that index is not in the `WHERE` clause of the `SELECT` statement. This reduces the number of indexes on a table, saving both space in the database and time when performing DML on the table, and still has the performance benefit of the index for `SELECT` statements.

Cluster Scans

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

Hash Scans

Oracle can use a hash scan to locate rows in a hash cluster based on a hash value. In a hash cluster, all rows with the same hash value are stored in the same data blocks. To perform a hash scan, Oracle first obtains the hash value by applying a hash function to a cluster key value specified by the statement. Oracle then scans the data blocks containing rows with that hash value.

EXPLAIN PLAN

```
SQL> EXPLAIN PLAN
  2  SET STATEMENT_ID = 'example' FOR
  3  SELECT * from employees
  4  WHERE job_id='ST_CLERK'
  5  AND department_id=20;
```

ID Operation

```
-- -----
0  SELECT STATEMENT Cost =
1  TABLE ACCESS BY ROWID (EMP)
2    AND-EQUAL
3      INDEX RANGE SCAN (JOB_INDEX)
4      INDEX RANGE SCAN (DEPTNO_INDEX)
```

ORACLE

7-5

Copyright © Oracle Corporation, 2001. All rights reserved.

The EXPLAIN PLAN Command

To execute a DML statement, Oracle may have to perform many steps. Each of these steps either physically retrieves rows of data from the database or prepares them in some way for the user issuing the statement. The combination of the steps Oracle uses to execute a statement is called an execution plan. An execution plan includes an access method for each table that the statement accesses and an ordering of the tables (the join order). “Access Methods” describes the various access methods, which include indexes, hash clusters, and table scans.

The EXPLAIN PLAN SQL command is available to display execution plans, as shown in the slide above. You can also use the AUTOTRACE setting in SQL*Plus.

Cached Execution Plans

- **Cached execution plans preserve the actual execution plan of a cached SQL statement in memory in a view called `v$sql_plan`.**
- **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.**
- **The view contains all of the `plan_table` columns, except the `level` column, in addition to four new columns.**

ORACLE

7-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Cached Execution Plans

Prior to Oracle9i, the `V$SQLTEXT` performance view could be used to gather useful information about cached cursors and execution statistics, such as the number of executions and the number of buffer gets from `V$SQL`. In order to determine the execution plan for a specific query, the developer had to run `EXPLAIN PLAN` for that query. 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 reducing the need for one extra step, and giving accurate and exact information for every query still in the library cache of the database.

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.

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.

SQL Trace and Tkprof

```
select customer_id from orderss where order_id = 49999
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.43	0.54	3	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	61.76	79.36	5883	5883	0	1
total	3	62.19	79.90	5886	5883	0	1

Misses in library cache during parse: 1
Optimizer hint: CHOOSE Parsing user id: 9

```
select customer_id from orderss where order_id = 49999
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.44	0.66	3	17	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.02	0.22	4	4	0	1
total	3	0.46	.88	7	21	0	1

Misses in library cache during parse: 1
Optimizer hint: CHOOSE Parsing user id: 9

ORACLE

7-7

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL Trace and TKPROF

The SQL trace facility and TKPROF enable you to accurately assess the efficiency of the SQL statements your application runs. For best results, use these tools with EXPLAIN PLAN, rather than using EXPLAIN PLAN alone. The SQL trace facility provides performance information on individual SQL statements. It generates the following statistics for each statement:

- Parse, execute, and fetch counts
- CPU and elapsed times
- Physical reads and logical reads
- Number of rows processed
- Misses on the library cache
- Username under which each parse occurred
- Each commit and rollback

You can enable the SQL trace facility for a session or for an instance. When the SQL trace facility is enabled, performance statistics for all SQL statements executed in a user session or in an instance are placed into a trace file. The additional overhead of running the SQL trace facility against an application with performance problems is normally insignificant, compared with the inherent overhead caused by the application's inefficiency.

Rule-Based Optimizer

1	Single row by rowid
2	Single row by Cluster Join
3	Single row by hash cluster key with unique key
4	Single row by unique or primary key
5	Cluster Join
6	Hash Cluster Key
7	Index Cluster Key
8	Composite Key
9	Single-column indexes
10	Bounded range search on indexed columns
11	Unbounded range search on indexed columns
12	Sort-merge join
13	MAX or MIN of indexed column
14	ORDER BY on indexed columns
15	Full Table Scan

ORACLE

7-8

Copyright © Oracle Corporation, 2001. All rights reserved.

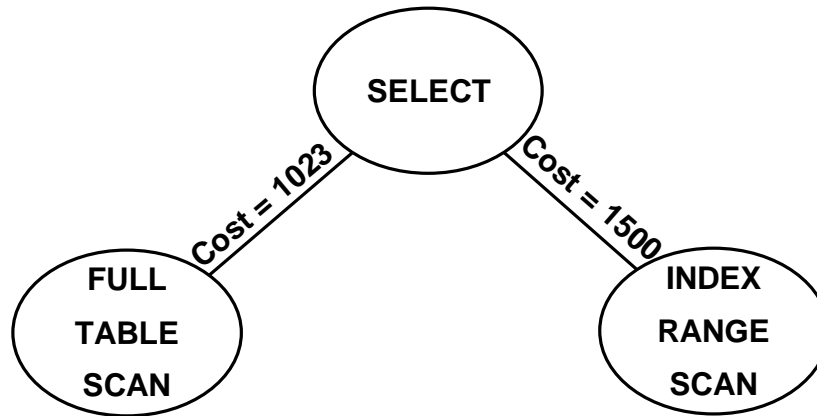
Rule-Based Optimizer

Using the rule-based approach, the optimizer chooses an execution plan based on the access paths available and the ranks of these access paths (shown on the slide). You can use rule-based optimization to access both relational data and object types. Oracle's ranking of the access paths is heuristic. If there is more than one way to execute a SQL statement, the rule-based approach always uses the operation with the lower rank. The assumption is that operations of lower rank execute faster than those associated with constructs of higher rank.

Rule-based optimization is supported for backward compatibility with earlier releases, and should not be used for new systems. The rule-based optimizer does not recognize many advanced features of Oracle9i such as partitions, index-organized tables, bitmapped indexes, function based indexes, histograms, hash joins, and star queries.

Cost-Based Optimizer

```
SQL> SELECT * from employees  
2 WHERE job_id='ST_CLERK';
```



ORACLE

7-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Cost-Based Optimizer

You should design new applications to use cost-based optimization. You should also use cost based optimization for data warehousing applications because the cost-based optimizer supports new and enhanced features for DSS. Much of the more recent performance enhancements, such as hash joins, improved star query processing, and histograms, are only available through cost-based optimization.

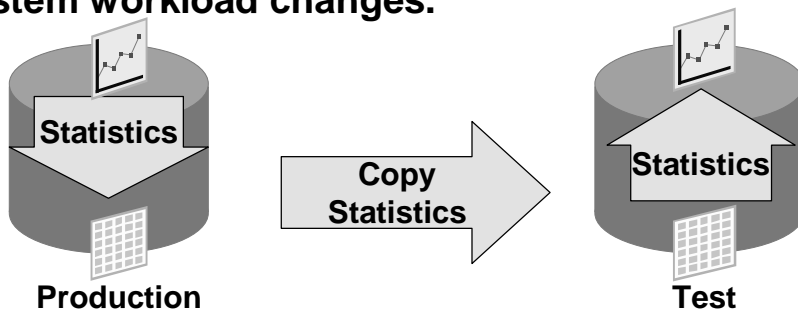
Using the cost-based approach, the optimizer determines which execution plan is most efficient by considering available access paths and statistics for the schema objects (tables or indexes) accessed by the SQL statement as well as system statistics regarding CPU time and I/O performance in the system. The cost-based approach also considers hints, which are optimization suggestions placed in a comment in the statement. You must gather statistics on a regular basis to provide the optimizer with information about schema objects. This task is done by the database administrator most of the time by using the `ANALYZE` command or procedures and functions in the package `DBMS_STATS`.

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

1. The optimizer generates a set of potential execution plans for the SQL statement based on its available access paths and hints.
2. The optimizer estimates the cost of each execution plan based on statistics in the data dictionary for the data distribution and storage characteristics of the tables, indexes, and partitions accessed by the statement as well as system statistics on CPU and disk I/O.
3. The optimizer compares the costs of the execution plans and chooses the one with the smallest cost.

Statistics Management

- **Optimizer statistics can be extracted and moved to another database.**
- **The test database better models the production database: tune execution plans on test database**
- **System statistics can be gathered to help the database use the system more efficiently when system workload changes.**



ORACLE

7-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Copying Statistics Between Databases

Statistics used by the cost-based optimizer can be copied from one database to another database. In the example above, the statistics from the production database can be copied to a test database, so the cost-based optimizer will select query execution plans based on the volume of the production database.

This is also very useful when you want to make a backup of the statistics, so that they can be easily restored at a later time.

System Statistics

You can also gather system statistics during different work hours and use these statistics to make sure that the best optimization path is chosen. If you gather statistics on the system during daytime operations on the first day and save them, it is possible to set these statistics for subsequent days. And during the first night, you can gather information about the system to set during subsequent nights. This makes it possible for the optimizer to know more about the system performance when the machine operates with different types of workloads.

DBMS_STATS Package

System statistics and the procedures for moving statistics between databases are all managed in the package DBMS_STATS.

With this package, you can use several new options when gathering statistics that are not available with the ANALYZE command.

Monitoring Table and Index Usage

- **Administrators need to know when tables are modified:**
 - Accurate statistics
 - Table reorganization
- **Similarly, administrators need to know when indexes are not used:**
 - Bad performance
 - Space wastage

ORACLE

7-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Monitoring Table and Index Usage

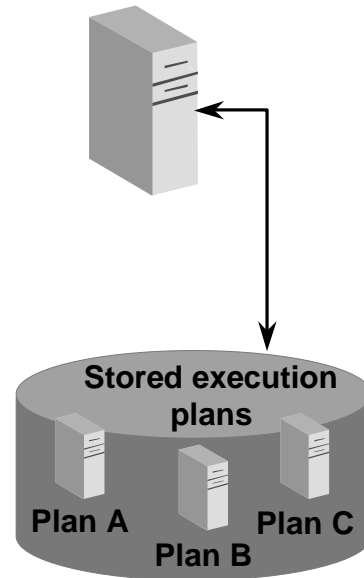
In order to generate accurate execution plans, the cost-based optimizer needs up-to-date statistics on referenced objects in the statement. With Oracle9i, you can ask the system to monitor changes in tables.

When enabled on a table, the system maintains information in the data dictionary for every modification (insert, update, delete, and direct load) on that table or one of its partitions. These changes are visible through the DBA_TAB_MODIFICATIONS view.

Similarly, one of the challenges in a database is to have the proper amount of indexes. Usually, when one index is being used improperly, it is easy to find because it causes bad performance. It is more difficult to find indexes that are not being used at all. To help in this process, Oracle9i monitors index usage with special data dictionary views and various commands for tables.

Optimizer Plan Stability

- **Save and distribute execution plans in stored outlines**
- **Consistency through `ANALYZE` and other DDL commands, and Oracle releases**
- **Protect tuning investments**
- **Support multisite application installations**
- **Possible to change a plan with private outline editing**



ORACLE

7-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Optimizer Plan Stability

After carefully tuning an application, you might want to make sure that the optimizer generates the same execution plan whenever the same SQL statements are executed. Plan stability allows you to maintain the same execution plans for the same SQL statements, regardless of changes to the database such as re-analyzing tables, adding or deleting data, modifying a table's columns, constraints, or indexes, changing the system configuration, or even upgrading to a new version of the optimizer.

This reduces much of the risk of introducing schema changes (new indexes), new applications upgrades, new Oracle releases, and statistics updates. By saving the existing execution plans, the administrator can perform the update and yet still utilize the old execution plans. It is possible to keep multiple named stored plan sets for experimentation and phased roll out.

Private Outline Editing

In Oracle9i, you can edit the created stored outlines. This is done by copying one or more public outlines to a private outline. You can then edit the private outline to get the desired execution path. After that, you copy the private outline back to the public outlines.

This changes the execution path for third party applications so that there is no possibility of changing the source code of the application.

Cursor Sharing

- **Cursor sharing can help the optimizer to reuse execution plans.**
- **Two execution modes:**
 - **Safe**
 - **Unsafe**

ORACLE

7-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Cursor Sharing

The term cursor sharing refers to an operation performed by the optimizer to reduce the overhead of parsing SQL statements. Rather than developing an execution plan each time a similar statement is executed, the optimizer parses the statement only the first time it is executed. As a DBA, you can switch from one mode to the other.

Safe Mode

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 only one execution plan for a similar SQL statement.

Unsafe Mode

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;
```

Using this feature is not safe because it might be better to use a full table scan in the first case and an index scan in the second.

8

Transactions

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

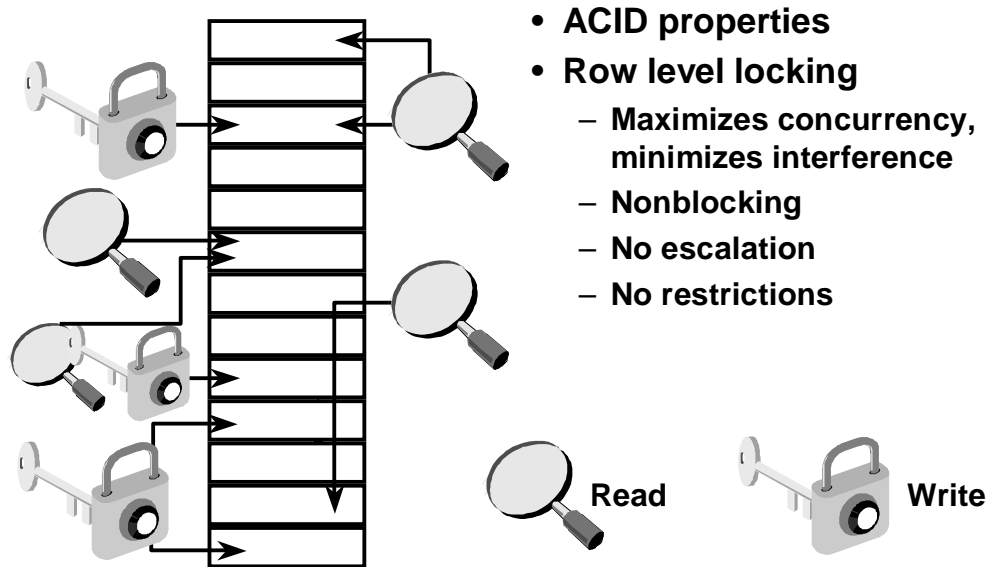
Objectives

After completing this lesson, you should be able to understand the implementation of transactions within Oracle9i:

- Integrity, concurrency, and consistency
- Undo segments
- Resumable statements
- Read consistency
- Transaction management
- Workspace Manager for long transactions
- Transaction types
- Distributed databases
- XA architecture

ORACLE

Data Concurrency Model



8-3

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

Data Concurrency Model

In a single-user database, the user can modify data in the database without concern for other users modifying the same data at the same time. However, in a multiuser database, the statements within multiple simultaneous transactions can update the same data. Transactions executed at the same time need to produce meaningful and consistent results. Therefore, control of data concurrency and data consistency is vital in a multiuser database. Data concurrency means that many users can access data at the same time. Data consistency means that each user sees a consistent view of the data, including visible changes made by the user's own transactions and transactions of other users.

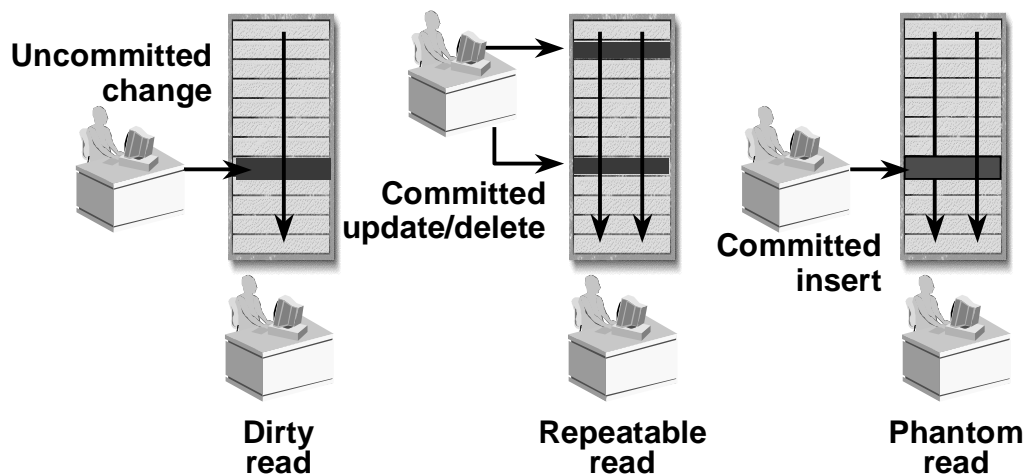
ACID Transaction Characteristics

- **Atomic:** Changes made by a transaction to the database are atomic (*all or nothing*).
- **Coherent:** The transaction changes the state of the database from one coherent state to a new coherent state (conforming to integrity constraints).
- **Isolated:** The transaction is isolated from other transactions and appears to be the only user on the system, even though it is executing with other concurrent transactions.
- **Durable:** The committed changes by a transaction are persistent, capable of surviving machine crashes.

Non-Blocking

Readers do not block readers, readers do not block writers, writers do not block readers; two writers cannot change the same row simultaneously.

Potential Concurrency Problems



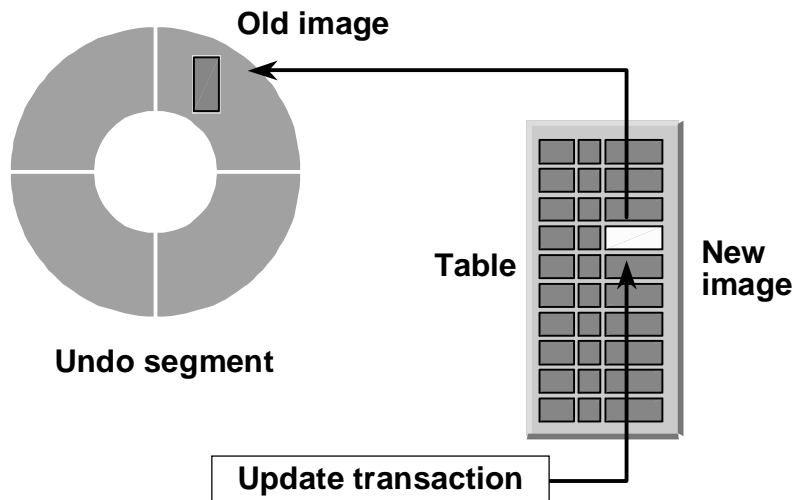
Potential Concurrency Problems

The ANSI/ISO SQL standard (SQL:1999) defines four levels of transaction isolation with differing degrees of impact on transaction processing throughput. These isolation levels are defined in terms of three phenomena that must be prevented between concurrently executing transactions.

The three preventable phenomena are:

- **Dirty reads:** A transaction reads data that has been written by another transaction that has not yet been committed.
- **Nonrepeatable (fuzzy) reads:** A transaction rereads data it has previously read and finds that another committed transaction has modified or deleted the data.
- **Phantom read:** A transaction re-executes a query returning a set of rows that satisfies a search condition and finds that another committed transaction has inserted additional rows that satisfy the condition.

Undo Information



ORACLE

8-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Undo Information

Each database contains one or more Undo tablespaces. An Undo tablespace records the old values of data that were changed by each transaction (whether or not committed). Undo tablespaces are used to provide read consistency, to roll back transactions, and to recover the database.

In one Undo tablespace, there are several undo segments maintained by Oracle. For each undo segment, Oracle maintains a transaction table, that is a list of all transactions that use the associated undo segment and the undo entries for each change performed by these transactions. Oracle uses the undo entries in a undo segment to perform a transaction rollback and to create read-consistent results for queries.

Undo tablespaces record the data prior to change on a per-transaction basis. For every transaction, Oracle links each new change to the previous change. If you must roll back the transaction, Oracle applies the changes in a chain to the data blocks in an order that restores the data to its previous state.

Similarly, when Oracle needs to provide a read-consistent set of results for a query, it can use information in an Undo tablespace to create a set of data consistent with respect to a single point in time.

Resumable Space Allocation

Resumable Space Allocation provides:

- **The ability to suspend and resume execution of large database operations**
- **A statement suspends if the following occurs:**
 - **Out of space condition**
 - **Maximum number of extents reached condition**
 - **Space quota exceeded condition**
- **An opportunity for the DBA to correct the error condition**
- **When the error is resolved, the suspended statement automatically resumes.**

ORACLE

8-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Resumable Space Allocation

Resumable Space Allocation provides the ability to suspend and resume execution of large database operations in the event of repairable failure. Resumable Space Allocation feature supports a class of errors related to space limits and out-of-space conditions.

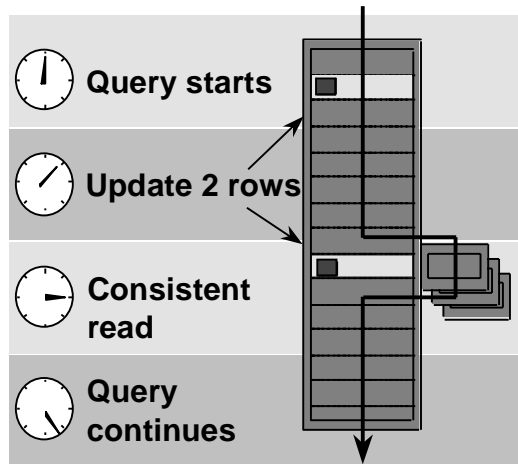
Large operations routinely run out of space or reach space limits after executing for a long time. When this condition occurs, the operation is rolled back and the error is returned to the user. The rollback takes time comparable to the amount of time the operation ran until it failed.

This feature suspends large operations that run into these problems. Suspending the operation gives the DBA an opportunity to take the corrective steps to resolve the error condition. Once the error condition disappears, the suspended statement automatically resumes the statement's execution.

Prior to this feature, customers used their home-grown mechanisms to deal with failure. They divided the operation into smaller chunks and wrote records that tracked the progress of the operation. This feature enables customers to write applications without worrying about running into common space-related errors. These errors can be handled and fixed while the large operation is in progress.

It is desirable for the user to register a procedure to be automatically executed when the statement gets suspended. This gives the user the opportunity to detect the problem and take appropriate actions.

Read Consistency



Multiversion read-consistency:

- Queries see a consistent view of the table
- No matter how many updates
- Reconstructed from undo segments
- Performance due to undo caching

Multiversion Concurrency Control

Oracle automatically provides read consistency to a query so that all the data that the query sees comes from a single point in time (statement level read consistency). Oracle can also provide read consistency to all queries in a transaction (transaction level read consistency).

Oracle uses the information maintained in its undo tablespace to provide these consistent views. The undo tablespace contains the old values of data that have been changed by uncommitted or recently committed transactions. The above slide shows how Oracle provides statement level read consistency using data in an undo tablespace.

As a query enters the execution stage, the current system change number (SCN or database time) is determined. As data blocks are read on behalf of the query, only blocks written with the observed SCN are used. Blocks with changed data (more recent SCNs) are reconstructed from data in the undo tablespace, and the reconstructed data is returned for the query. Therefore, each query returns all committed data with respect to the SCN recorded at the time that query execution began. Changes of other transactions that occur during a query's execution are not observed, guaranteeing that consistent data is returned for each query.

Oracle Flashback

- **Users can see a consistent view of the database at a point in the past.**
- **Point in past is based on a system time or a system change number (SCN).**
- **Only transactions committed up until that time are visible.**
- **Possible applications are:**
 - **Self-service repair**
 - **Packaged applications like e-mail**
 - **Decision support systems for trend analysis**
- **UNDO_RETENTION is set for undo availability.**

ORACLE

8-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Flashback

Applications can be executed on the version of the database as of a certain time. For example, a Windows application for customer information may have a button that allows the user to move back to an earlier point in time to show the account balance as of a certain time. Previously, much of this information had to be saved away by the application. This feature allows accessing information in the past, even when it was not explicitly saved away.

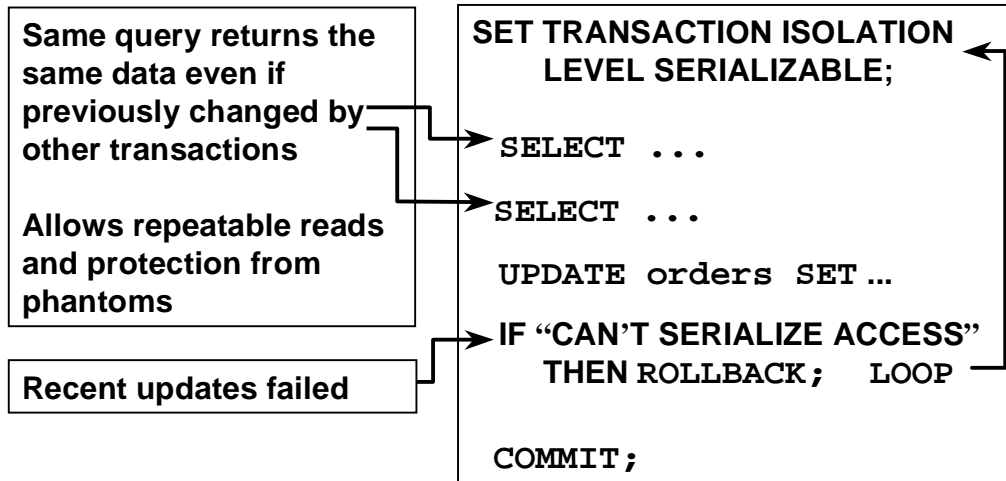
There may be several reasons why users would like to query past data. One important application would be self-service repair, when some important rows were accidentally deleted from a table and the users wanted to recover them. To perform the repair, they could possibly move backwards in time, see the missing rows, and reinsert the deleted row into the current table. Care must be taken not to introduce logical inconsistencies.

Other potential benefits include packaged applications such as e-mail and voice mail. The users may have accidentally deleted a mail or a voice message by pressing the wrong key. Using temporal access, they would be able to restore the deleted mail or message by moving back in time and reinserting the deleted message into the current message box.

Oracle Flashback leverages the Automatic Undo Management functionality which can retain the undo information for a user specified retention interval. Therefore, all queries addressed to a system time which is not earlier than a retention interval from the current time have enough undo information to reconstruct the snapshot. The amount of disk space required will depend on how far back you intend to go.

Serializable Transactions

- Extend *read consistency* at the transaction level
- TPC, ANSI/ISO SQL compliant



ORACLE

8-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Serializable Transactions

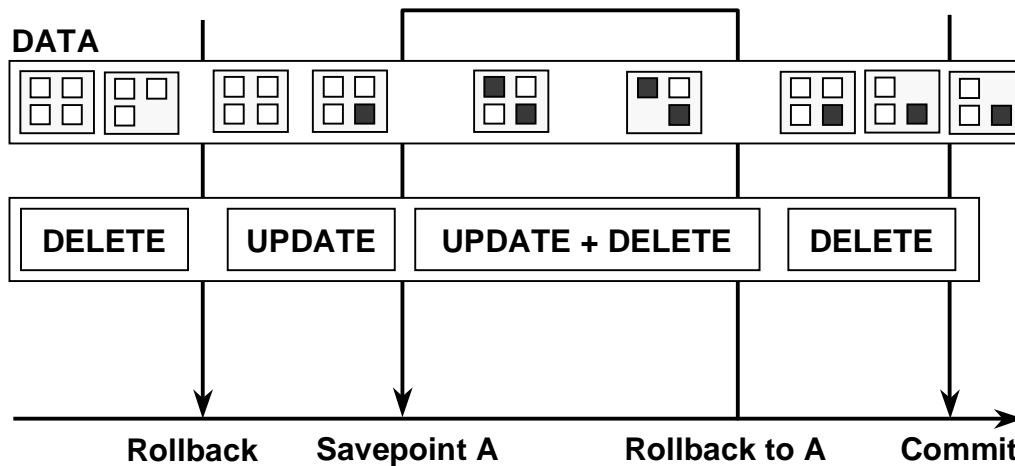
To describe consistent transaction behavior when transactions execute at the same time, database researchers have defined a transaction isolation model called serializability. Serializability tries to ensure that transactions execute in such a way that they appear to be executed one at a time, or serially, rather than concurrently. While this degree of isolation between transactions is generally desirable, running many applications in this mode can compromise application throughput. Complete isolation of concurrently running transactions could mean that one transaction cannot perform an insert into a table being queried by another transaction. Real-world considerations usually require a compromise between perfect transaction isolation and performance.

SQL:1999 defines four levels of isolation in terms of the phenomena that a transaction running at a particular isolation level is permitted to experience:

Isolation level	Dirty read	Nonrepeatable read	Phantom read
Read uncommitted	Possible	Possible	Possible
Read committed	Impossible	Possible	Possible
Repeatable read	Impossible	Impossible	Possible
Serializable	Impossible	Impossible	Impossible

Oracle offers the read committed and serializable isolation levels, as well as a read-only mode that is not part of SQL:1999. Read committed is the default and was the only automatic isolation level provided before Oracle Release 7.3.

Transaction Management



ORACLE

8-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Transaction Management

A transaction is a logical unit of work that comprises one or more SQL statements executed by a single user. According to the ANSI/ISO SQL standard, a transaction begins with the user's first executable SQL statement. A transaction ends when it is explicitly committed or rolled back by that user.

The changes made by the SQL statements that constitute a transaction can be either committed or rolled back. After a transaction is committed or rolled back, the next transaction begins with the next SQL statement. Committing a transaction makes permanent the changes resulting from all SQL statements in the transaction. The changes made by the SQL statements of a transaction become visible to other user session's transactions that start only after the transaction is committed.

Rolling back a transaction retracts any of the changes resulting from the SQL statements in the transaction. After a transaction is rolled back, the affected data is left unchanged as if the SQL statements in the transaction were never executed.

For long transactions that contain many SQL statements, intermediate markers or savepoints can be declared. Savepoints can be used to divide a transaction into smaller parts. By using savepoints, you can arbitrarily mark your work at any point within a long transaction. This allows you the option of later rolling back all work performed from the current point in the transaction to a declared savepoint within the transaction.

Workspace Management for Long Transactions

- **A long transaction can exist for several days.**
- **Is used for analysis and to test changes in data.**
- **A shared, transactionally consistent view of database tables, that:**
 - **Captures changes to version-enabled tables as new row versions within the workspace**
 - **Makes versions invisible outside the workspace until explicitly merged with the parent workspace**
 - **Makes changes directly to tables that are not version-enabled.**

ORACLE

8-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Long Transactions

To help with complex queries and scenarios where you might want to see and test changes to data before they are finalized, it is now possible to use long transactions. A long transaction can exist for several days and is managed by creating different versions of a table. To set the version you want to use, you connect to the proper workspace.

A workspace is a virtual environment that provides a transactionally consistent snapshot of the entire database. One or more users can share this environment to version data in the database.

The unit of versioning is the table. When a user in a workspace updates a row in a version-enabled table, a new version of the row is created. Versions are only accessible within the workspace until explicitly merged with the parent workspace. This also applies to inserts and deletes made within a workspace to a version-enabled table.

There can be one or more versions of a row in a workspace from one or more version-enabled tables. The current or active version of a row in a workspace refers to the version of a row to which changes are currently being made.

Tables that are not version-enabled will be changed as usual, regardless of which workspace a user might be in. Access to any table is still controlled by the privileges a user has, not which workspace a user is in.

Special Transaction Types

- **Discrete transactions**
 - Short, nondistributed
 - Changes deferred until `commit`
 - No undo generated
- **Autonomous transactions**
 - Transactions within a transaction
 - Commit/rollback independently

ORACLE

8-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Discrete Transactions

Application developers can improve the performance of short, nondistributed transactions by using the procedure `BEGIN_DISCRETE_TRANSACTION`. This procedure streamlines transaction processing so that short transactions can execute more rapidly. During a discrete transaction, all changes made to any data are deferred until the transaction commits. Oracle generates redo information, but stores it in a separate location in memory.

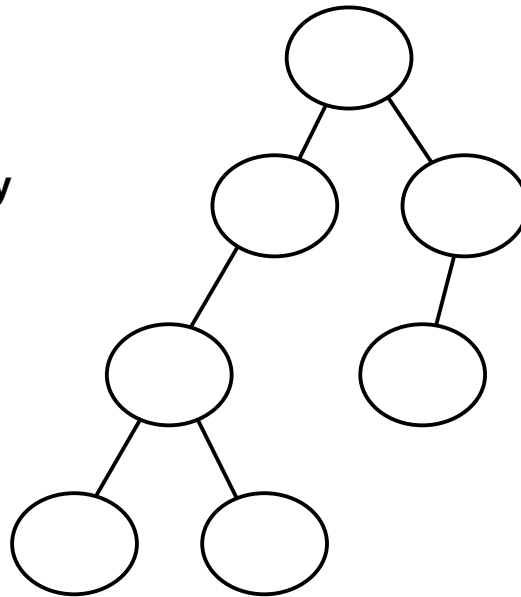
When the transaction issues a commit request, Oracle writes the redo information to the redo log file, and applies the changes directly to the data block. Oracle returns control to the application once the commit completes. This eliminates the need to generate undo information, since the block is actually not modified until the transaction is committed, and the redo information is stored in the redo log buffers.

Autonomous Transactions

Autonomous transactions are independent transactions that can be called from within another transaction. An autonomous transaction lets you *step out* of the context of the calling transaction, perform some SQL operations, commit or roll back those operations, and then return to the calling transaction's context and continue with that transaction. Once invoked, an autonomous transaction is totally independent of the calling transaction (the main transaction). It does not see any of the uncommitted changes made by the main transaction and does not share any locks or resources with the main transaction.

Distributed Databases

- **One logical database**
- **Autonomous nodes**
- **Location transparency**
- **Two-phase commit**



ORACLE

8-13

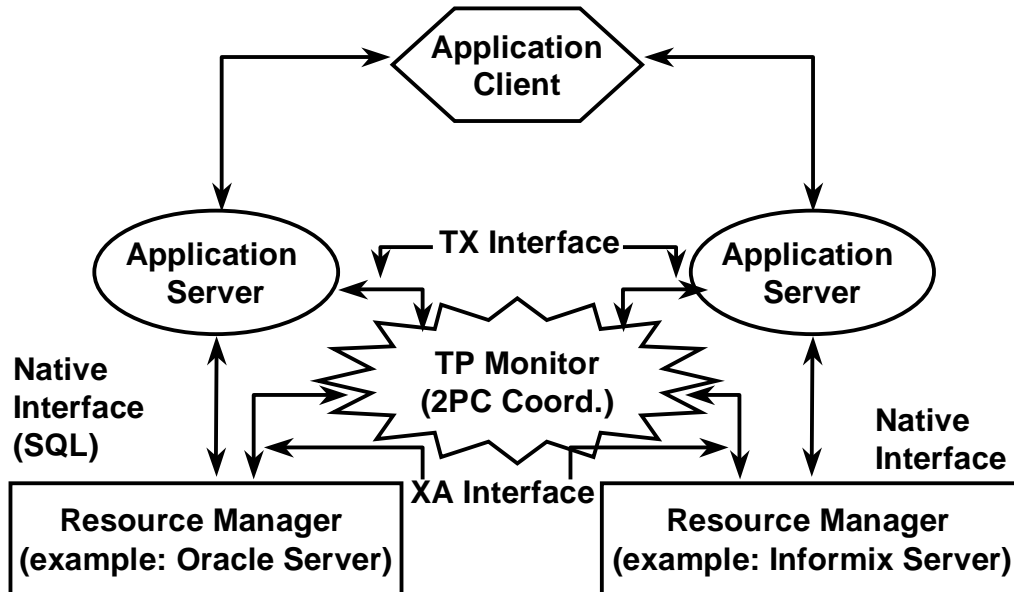
Copyright © Oracle Corporation, 2001. All rights reserved.

Distributed Databases

A distributed database is a network of databases managed by multiple database servers that appear as a single logical database to a user. The data of all databases in the distributed database can be simultaneously accessed and modified. The primary benefit of a distributed database is that the data of physically separate databases can be logically combined and potentially made accessible to all users on a network. Each computer that manages a database in the distributed database is called a node. The database to which a user is directly connected is called the local database. Any additional databases accessed by this user are called remote databases. When a local database accesses a remote database for information, the local database is a client of the remote server (client/server architecture). While a distributed database allows increased access to a large amount of data across a network, it must also provide the ability to hide the location of the data and the complexity of accessing it across the network.

Oracle provides the same assurance of data consistency in a distributed environment as in a nondistributed environment. Oracle provides this assurance using the transaction model and a two-phase commit mechanism. As in nondistributed systems, transactions should be carefully planned to include a logical set of SQL statements that should all succeed or fail as a unit. Oracle's two-phase commit mechanism guarantees that no matter what type of system or network failure might occur, a distributed transaction either commits on all involved nodes or rolls back on all involved nodes to maintain data consistency across the global distributed database.

XA Architecture



ORACLE

8-14

Copyright © Oracle Corporation, 2001. All rights reserved.

XA Architecture

In the traditional Oracle client/server architecture, presentation and business logic is concentrated into a single executable which is distributed to each client machine. Occasionally, database packages and procedures are used to move some business logic to the Oracle Server; however, the architecture is inherently 2-tier. Each application client has a corresponding shadow process (or thread) running within the database server. This architecture provides great flexibility. However, it runs into problems when scaling to large numbers of concurrent users.

One approach for improving scalability is to split the application logic into an application client and application server. The application client becomes primarily responsible for the collection and presentation of data. The application server becomes responsible for providing business services which it implements through accessing various data stores (resource managers).

The application server will usually be written in a 3GL such as C or COBOL, and interface to a resource manager using the normal precompiler or API mechanisms. Application client and server communicate through a messaging interface.

In the Oracle case, application servers can be implemented with precompilers or OCI. Connection to Oracle can be direct or through Oracle Net.

XA Architecture (continued)

The TP architecture can be extended to provide support for distributed data stores by simply routing messages to different application servers. This can be homogeneous (Oracle to Oracle) or heterogeneous (Oracle to any other DBMS). The limitation is that the transaction is separately committed on each data store. In other words, transactions from two data stores cannot be combined into a single global transaction with global transaction properties.

To achieve global heterogeneous transactions you need XA. In the XA architecture you can implement global transactions, protected by 2PC. Oracle9i supports the XA interface.

Each resource manager exports an XA API. The TP monitor becomes the 2PC coordinator and it uses the XA API to control the prepare, commit, and rollback of transactions within the resource managers.

Direct commit and rollback statements are removed from application servers and replaced by calls to the TP monitor through the TX interface.

Unit 1 Summary

- **Relational concepts**
 - Domains, tables, rows, columns
 - Missing information
 - Integrity
- **Languages and interfaces**
 - SQL, embedded SQL, PL/SQL, JAVA
 - XML
 - OCI, ODBC, OLE
 - JDBC, SQLJ, EJB, CORBA

ORACLE

Unit 1 Summary

Database objects

- Tables, clusters, data types
- Indexes, constraints, triggers
- Views, materialized views
- Sequences, queues
- Procedures, functions, packages

ORACLE

Unit 1 Summary

- **Basic Oracle9i architecture**
 - Instances, databases, data files
 - Segments, extents, blocks
 - Data dictionary
 - Shared server
- **Real Application Clusters architecture**
 - Cache fusion
 - Shared server-side initialization parameter file
 - Sharing resources

ORACLE

Unit 1 Summary

- **Security**
 - Users, schemas, authentication
 - Quotas, resource limits, profiles
 - Privileges and roles
 - Auditing
- **The optimizer**
 - Rule based
 - Cost based
- **Transactions**
 - Concurrency
 - Read consistency

ORACLE

Unit 2

Optional Lessons



Oracle9i Object-Relational Features

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

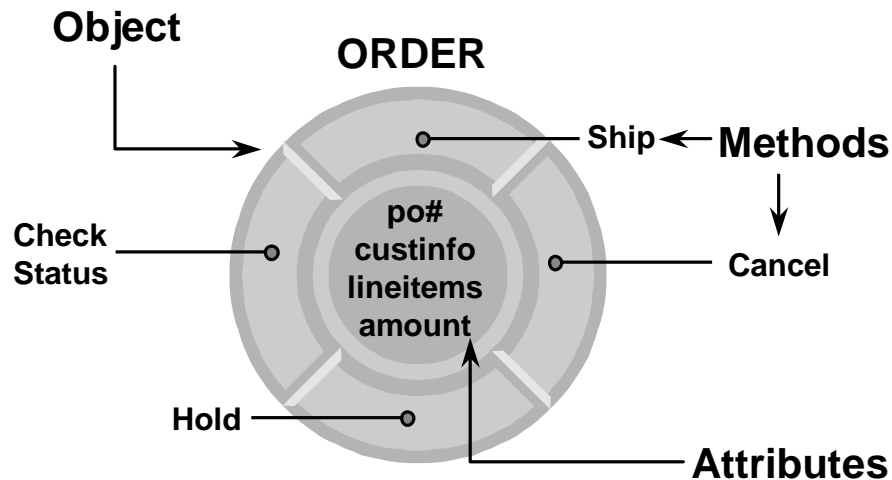
Objectives

After completing this lesson, you should be able to explain the basic concepts of the object-relational theory:

- **Objects, attributes, and methods**
- **Inheritance, polymorphism, and encapsulation**
- **Classes**
- **Object tables and object views**
- **Optimizer extensibility**
- **Object Type Translator (OTT)**
- **Client side object cache**
- **Inheritance**
- **SQLJ object type**

ORACLE

What Is an Object?



ORACLE

9-5

Copyright © Oracle Corporation, 2001. All rights reserved.

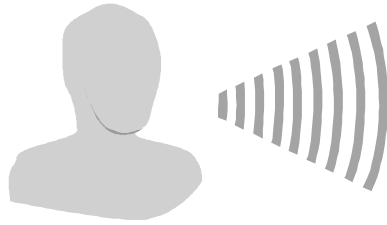
What is an Object?

David Taylor (Object Guru) describes objects as:

“Software packets responsible for their information (attributes), and their methods (operations) for operating on that information.”

In reality, an object is just another step forward in the software engineering life cycle. You model real things and represent these things as objects. In other words, you can develop solutions that simulate real-world problems. The model is stated in terms of the interactions between objects. Object technology has been around for over twenty years. Data-driven analysis was the first step towards objects. It recognizes that data is constant whereas process is in constant change. Some of the benefits of object technology are easy maintenance of applications, faster product development, and higher quality of code at a lower cost.

Object Terminology



- **Object type**
- **Method**
- **Class**
- **Object instance**
- **Inheritance**
- **Subclass**
- **Polymorphism**
- **Encapsulation**

ORACLE

9-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Object Terminology

An object type is a person, place, or thing. An object type knows things about itself and has behavior.

A class is the factory which creates an object; it is the template from which instances of that class are created. The Oracle term for class is object type.

For example, the class *watch* is sent a message *new* which invokes the class method *new* which then creates an instance of the class *watch*. Repeated sends of the class message *new* create additional instances of the class *watch*. These instances are initially identical and have no knowledge of each other. An instance is created from the template for its class. As messages are processed by their methods, their state changes. An instance is an object that belongs to a particular class. In relational terms, instance is analogous to a row or record.

Inheritance allows objects to use behavior and attributes in classes above them in the hierarchy. You can specialize any class by creating a subclass. The subclass will inherit all the previously-defined behaviors and attributes of the superclass.

Object Terminology (continued)

Polymorphism allows the same operation (displayTime) to be performed on objects of different classes with possibly different effects. This is useful because one can ask objects of different types to invoke the same method and acquire behavior appropriate for that type of object.

Encapsulation allows an object's state to be accessed (read or modified) exclusively through its methods. Encapsulation is a way of protecting an object by controlling how it is accessed. Encapsulation hides the low-level implementation details from users and developers. Methods are used to encapsulate objects.

A method is an operation invoked in response to a message. The method generally operates on the object's state. Behaviors are verbs. Attributes are nouns.

More precisely, methods are procedures or functions provided to enable applications to perform useful operations on the attributes of the object type. Methods are an optional element of an object type. They define the behavior of objects of that type and determine what, if anything, that type of object can do.

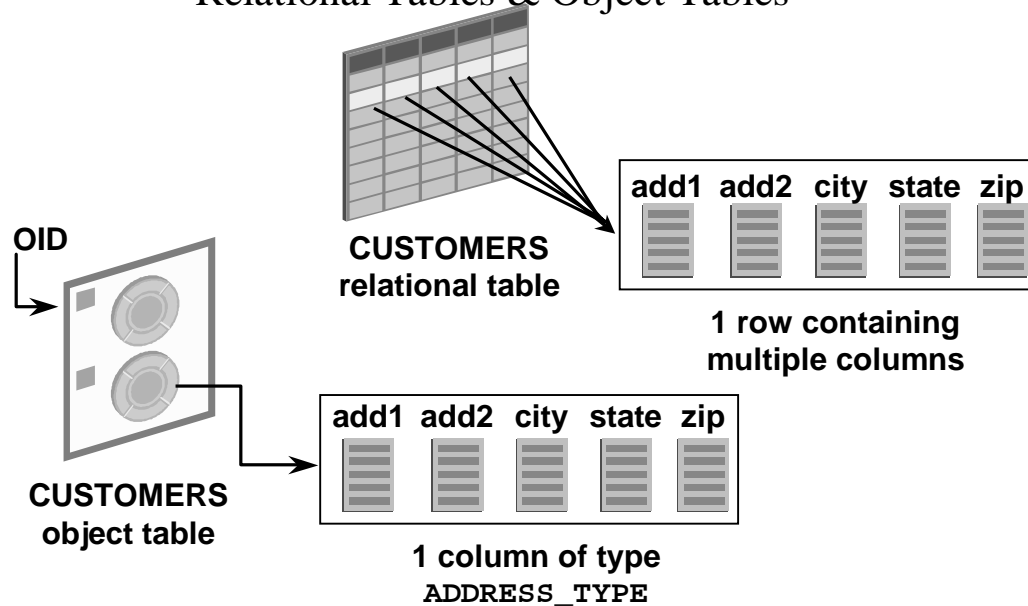
A "Computer World" survey breaks down object language usage as follows:

- C++
- Smalltalk
- ObjectiveC
- Others (for example, Java)

Java is quickly becoming the language of choice.

Relational and Object Tables

Relational Tables & Object Tables



ORACLE

9-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Relational and Object Tables

Relational Tables are the basic unit of data storage in an Oracle database. Data is stored in rows and columns. You define a table with a table name and set of columns. You give each column a column name, a data type, and a width (the width might be predetermined by the data type, as in DATE) or precision and scale (for columns of the NUMBER data type only). A row is a collection of column information corresponding to a single record.

An object table is a special kind of table that holds objects and provides a relational view of the attributes of those objects. Oracle allows you to view an object table in two ways:

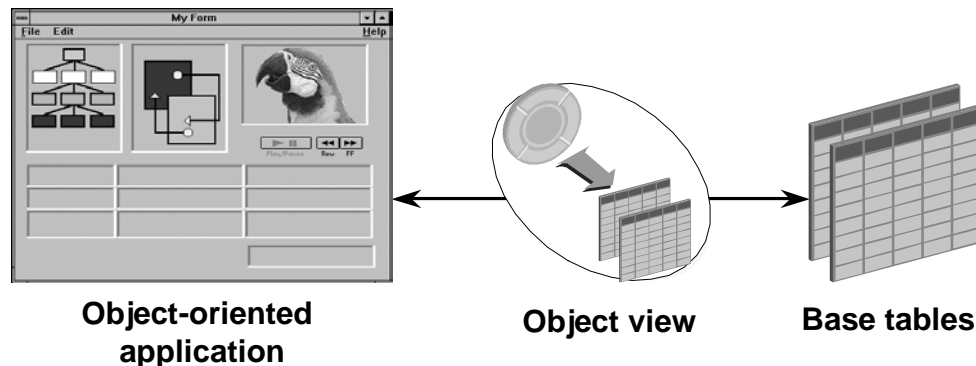
- A single column table in which each entry is an object
- A multicolumn table in which each of the attributes of the object type occupies a column

Each object in an object table has a special identifier called an OID or object identifier that is unique inside the database. Objects that appear in object tables are called row objects. Objects that appear only in table columns or as attributes of other objects are called column objects.

In addition to permanent tables, Oracle can create temporary tables to hold session-private data that exists only for the duration of a transaction or session. For transaction-specific temporary tables, data exists for the duration of the transaction while for session-specific temporary tables, data exists for the duration of the session. Data in a temporary table is private to the session. Each session can only see and modify its own data.

Object Views: An Evolutionary Approach

Enable programmers to evolve existing schemas to support objects



ORACLE

9-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Object Views: An Evolutionary Approach

Migration to object-oriented applications can be done over time, and new object-oriented applications can access and modify the same data as the old applications, with no changes in the legacy applications or procedures.

You can choose to maintain the base tables in the style of relational segments that have been proven in earlier versions of the Oracle server. However, this does not prevent users from building object views on top of the tables to accommodate a specific schema for an object-oriented application.

Object views provide several benefits to Oracle users, such as:

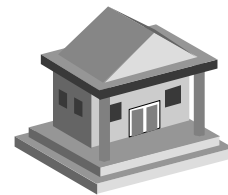
- Access to relational data for object-oriented applications
- Flexibility in supporting multiple object-based schemas for relational and object data
- Coexistence of current applications with new applications that leverage the object-oriented framework

In addition, multiple applications may *look at* the data in different ways. For example, the sales force of a car dealership wants information on features, price, and availability, and may require a picture of a particular car. But management wants to know what types of cars are selling best, what colors do not sell, what is on the lot, and what sales are being missed because of the lack of a type of car. These applications can be built upon separate schemas but based on the same tables (relational or object).

Extensible Type System

Oracle9i has an open type system that provides the basis for object extensions:

- **Simple:** Easy to understand and use
- **Comprehensive:** Includes many or all relevant types (user-defined types)
- **Extensible:** Manages complex data



OPEN

ORACLE

9-10

Copyright © Oracle Corporation, 2001. All rights reserved.

The Object-Relational Model

The object-relational model allows users to define object types, specifying both the structure of the data and the methods of operating on the data, and to use these data types within the relational model. One of the reasons for having a type system is to ensure that all types are type checked. Type checking occurs on user-defined object types as well as intrinsic types supplied with the Oracle9i server.

References

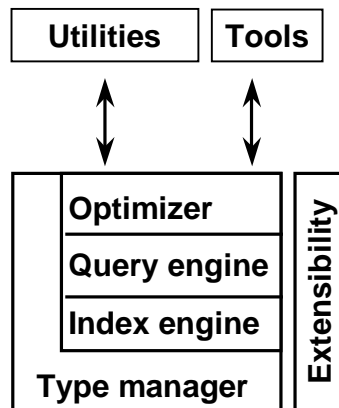
References (REFs) are a natural substitute for a primary key-foreign key relationship found in traditional SQL systems. In the `emp` table, the `deptno` column is a foreign key and records the name of the department of the corresponding employee. In contrast, `deptno_ref` is another field in the `emp` table that serves exactly the same purpose, that is, to identify the employee's department name. The main difference between the two implementations is that with the primary key-foreign key, you must join the two tables (`dept` and `emp`) to obtain the information, whereas with Ref, you only need to use the `DEREF` function.

Rules

Another required feature in a object-relational DBMS is a rules system. Rules protect the integrity of the data in the database. Although relational systems use triggers, object-relational systems need a more flexible system.

The general form of a rule is: "When an event occurs, perform an action."

Extensible Cartridge Development Enabling Multimedia Applications



- **Extensibility framework allows partners to add support for new data types.**
- **Oracle provides a set of extensible APIs to core functions; for example, indexing, optimizer, operators, memory.**
- **Development, packaging, and installation tools**

ORACLE

9-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Extensible Cartridge Development

Increasingly, databases are being used to store more varied types of data, such as spatial, audio, or video data. This brings the need for indexing complex data types and for specialized indexing techniques. The Oracle9i database provides an interface (SQL) that enables developers to define domain-specific operators and indexing schemes, and to integrate them into the Oracle database. A set of built-in operators is provided for use in SQL statements, which include arithmetic operators (+, -, *, /), comparison operators (=, >, <), logical operators (NOT, AND, OR), and set operators (UNION). These operators take as input one or more arguments (operands) and return a result.

The extensibility framework of the Oracle9i database allows developers to define new operators. Their implementation is provided by the user, but the Oracle9i database allows these user-defined operators to be used in SQL statements in the same manner as any of the predefined operators. The framework to develop new index types is based on the concept of cooperative indexing, where an application and the Oracle9i database cooperates to build and maintain indexes for complex data types. The application software is responsible for defining the index structure, maintaining the index content during load and update operations, and searching the index during query processing. The index structure itself can be stored either in the Oracle9i database as tables or externally as files. Indexes created using these new index types are referred to as domain indexes.

Extensible Cartridge Development (continued)

The optimizer functionality has been extended to allow authors of user-defined functions and domain indexes to create statistics collection, selectivity, and cost functions that will be used by the optimizer in choosing a query plan.

Statistics Collection Functions: The `ANALYZE` command can now make a call to a user-specified statistics collection function whenever a domain index is analyzed. User-defined statistics collection functions can also be defined for individual columns of a table and for user-defined data types.

Selectivity Functions: Users can specify user-defined selectivity functions for predicates containing user-defined operators, stand-alone functions, or type methods.

Cost Functions: Users can define costs for domain indexes and for user-defined stand-alone functions, package functions, and type methods. The user-defined costs can be in the form of default costs that the optimizer simply looks up, or they can be computed by functions called by the optimizer.

Object Type Translator

- **Eases application development by providing client-side mappings to Oracle9i object schemas**
- **Generates header files using schema information from an Oracle9i data dictionary**
- **Generates C structs and corresponding indicator variables**

ORACLE

9-13

Copyright © Oracle Corporation, 2001. All rights reserved.

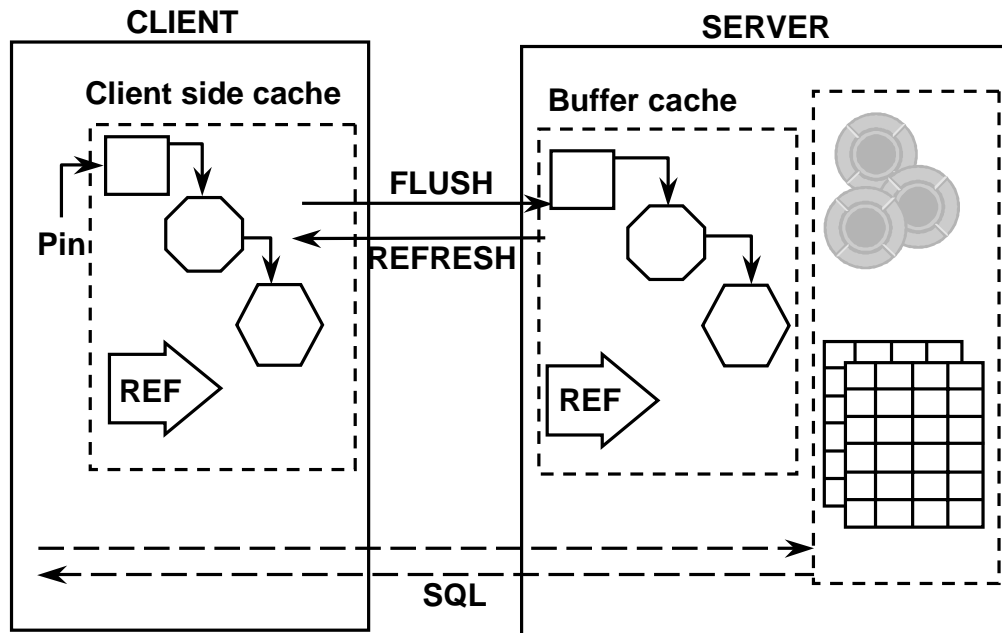
Object Type Translator

To simplify application development with objects, Oracle offers the object type translator (OTT), which automatically maps server-side object schemas to client-side language bindings. For example, a person object type in the server is mapped to a struct *person* in the C language for use with OCI and Pro*C programs.

There are three important benefits that the OTT provides:

- Simplification of application development by mapping complex schemas automatically. This allows users to iteratively develop applications with Oracle9i database objects.
- The mapping is performed intelligently; particularly, the server understands the concept of NULLity while programming languages such as C do not have such a concept. The OTT generates both the structs and appropriate indicator variables that can be used to indicate NULLity.
- The OTT takes as input a data dictionary and generates the corresponding C header files that can be included with OCI and Pro*C programs. It also generates an outtype file which provides information to the Pro*C precompiler. This is used to check the types of the variables used in the program at compile time, significantly improving developer productivity.

Oracle9i Client Side Object Cache



ORACLE

9-14

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i Client Side Object Cache

The object cache is a client-side memory buffer that provides lookup and memory management support for objects. It stores and tracks object instances that have been fetched by an OCI application.

When objects are fetched by the application through a SQL SELECT, or through an OCI pin operation, a copy of the object is stored in the object cache. Objects that are fetched directly through a SELECT statement are fetched by value, and they are nonreferenceable objects which cannot be pinned. Only referenceable objects can be pinned.

If an object is being pinned, and an appropriate version already exists in the cache, it does not need to be fetched from the server. Every client program that uses the Oracle OCI to dereference REFs to retrieve objects, utilizes the object cache. A client-side object cache is allocated for every OCI environment handle initialized in object mode. Multiple threads of a process can share the same client-side cache by sharing the same OCI environment handle. Exactly one copy of each referenceable object exists in the cache, per connection.

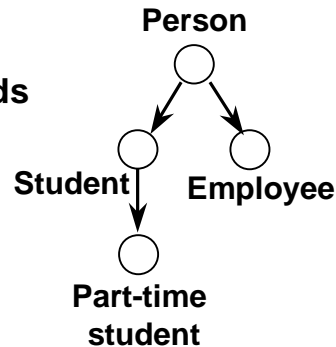
Dereferencing a REF many times or dereferencing several equivalent REFs returns the same copy of the object. If you modify a copy of an object in the cache, you must flush the changes to the server before they are visible to other processes.

Oracle9i Client Side Object Cache (continued)

Objects that are no longer needed can be unpinned or freed, and then swapped out of the cache, freeing the memory space they occupied. The object cache maintains the association between all object copies in the cache and their corresponding objects in the database. The cache does not manage the contents of object copies; it does not automatically refresh object copies. The application must ensure the correctness and consistency of the contents of object copies. For example, if the application marks an object copy for insert, update, or delete, and then aborts the transaction, the cache simply unmarks the object copy but does not purge or invalidate the copy. The application must pin *recent* or *latest*, or refresh the object copy in the next transaction. If it pins *any*, it may get the same object copy, with its uncommitted changes, from the previous aborted transaction.

Type Inheritance

- **Inheritance is a mechanism to organize types in a hierarchy.**
- **This is in accordance to ANSI SQL:1999.**
- **Subtypes inherit supertypes.**
- **A subtype can:**
 - **Add new attributes and methods**
 - **Override supertype methods**



ORACLE

9-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Type Inheritance Methodology

Type inheritance allows sharing similarities between types as well as extending their characteristics. Object applications typically are organized into types and type hierarchies consisting of supertypes and subtypes. A supertype is an object and a subtype is another object which has inherited from (extended) one supertype. A subtype inherits all its supertype's attributes and methods. A subtype can also add new attributes and methods of its own. All attributes and methods of a type are accessible by other methods in the same type and its subtypes, or to methods that have access to that type.

A subtype can override any method in its supertype chain. The subtype value appears to the surrounding code just as the value of the supertype would, even if it uses separate mechanisms within its specialization methods. Instance substitutability refers to the ability to use an object value of a subtype in a context declared in terms of a supertype. REF substitutability refers to the ability to use a REF to a subtype in a context declared in terms of a REF to a supertype.

SQLJ Object Type

- **SQLJ Object is a special SQL type.**
- **Each SQLJ Object type maps to a Java class inside the database.**
- **A Java application can INSERT or SELECT objects directly into or from the database through the Oracle9i JDBC drivers.**
- **The database SQL engine can access the attributes of the Java class as a regular SQL type.**

ORACLE

9-17

Copyright © Oracle Corporation, 2001. All rights reserved.

SQLJ Object Type

SQLJ object is a special SQL type that is fully implemented in Java. The SQLJ Part2 standard API allows access to object relational features. Each SQLJ object type maps to a Java class inside the database.

This will make it easier to map Java classes in the application to objects in the database. By using Java classes as SQL types, the user can define columns or rows of this Java type and query and manipulate the objects of this type.

Once the mapping is registered through the `CREATE TYPE SQL DDL`, the Java application can INSERT or SELECT copies of the Java objects directly into or from the database through the Oracle9i JDBC drivers.

The database SQL engine can access the attributes of the Java class as a regular SQL type.

10

Globalization Support

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

After completing this lesson, you should be able to discuss the Globalization Support features of Oracle9i including:

- **Oracle Globalization Support architecture**
- **Standard features**
- **Customization features**
- **Unicode**

ORACLE

Globalization Support Features

- Language support
- Territory support
- Character set support
- Linguistic sorting
- Message support
- Date and time formats
- Numeric formats
- Monetary formats
- Time zones



ORACLE

10-3

Copyright © Oracle Corporation, 2001. All rights reserved.

Globalization Support Features

Oracle's Globalization Support architecture allows you to store, process, and retrieve data in native languages. It ensures that database utilities and error messages, sort order, date, time, monetary, numeric, and calendar conventions automatically adapt to the native language and locale. Parameter settings determine the behavior of individual conventions.

Language Support

Oracle9i allows users to store, process, and retrieve data in 57 native languages.

Message Support

Utilities and error messages can be made to appear in the native language.

Territory Support

Oracle9i supports 88 different cultural conventions which are specific to a given geographical location. Local time, date, numeric, and monetary conventions are handled.

Date and Time Formats

The world's various conventions for hour, day, month, and year can be handled in local formats.

Monetary and Numeric Formats

Currency, credit, and debit symbols can be represented in local formats. Dual currency is supported as well. Radix symbols and thousands separators can be defined by locales.

Globalization Support Features (continued)

Calendars

Gregorian, Japanese Imperial, ROC Official, Thai Buddha, Persian, English Hijrah, and Arabic Hijrah are supported.

Linguistic Sorting

Oracle9i provides linguistic sorts for culturally accurate sorting.

Character Set Support

Oracle supports a large number (more than 150) of single-byte, multibyte, and fixed-width encoding schemes that are based on national, international, and vendor-specific standards.

Customization Features

Oracle allows you to customize character sets and calendars. User-defined characters are sometimes needed to support special symbols, vendor-specific characters, or characters that represent proper names, historical terms, and so on.

Calendar Customization

You can define ruler eras for imperial calendars, and deviation days for lunar calendars.

Time Zones

You can create or alter a database with a specific time zone by specifying a displacement from UTC (Coordinated Universal Time, formerly Greenwich Mean Time). The database time zone is relevant only for `TIMESTAMP WITH LOCAL TIME ZONE` columns.

Encoding Scheme Types

- **Single-byte character sets**
 - 7-bit (US7ASCII)
 - 8-bit (WE8ISO8859P1)
- **Varying-width multibyte character sets (JA16EUC)**
- **Fixed-width multibyte character sets must be Unicode**
- **Unicode (UTF8, AL16UTF16)**

ORACLE

10-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Encoding Scheme Types

A character encoding scheme specifies numeric codes corresponding to characters that a computer or terminal can display and receive. Character encoding schemes are used to interpret data into meaningful symbols from a terminal to a host machine.

Oracle provides different classes of encoding schemes:

- **Single Byte:** In a single byte character set, each character occupies one byte.
- **Varying-width:** A multibyte character set is represented by one or more bytes per character. Multibyte character sets are commonly used for Asian language support. Varying-width character schemes do not have a fixed number of bytes for each character. Some multibyte encoding schemes use the value of the most significant bit to indicate if a byte represents a single or is part of a series of bytes representing a character.
- **Fixed-width:** This provides similar support for multibyte character sets, except that the format is a fixed number of bytes for each character.
- **Unicode:** This is a worldwide character encoding standard that can represent all characters for computer usage. Oracle9i supports Unicode 3.0, the third version of the Unicode Standard. For more information about the Unicode Standard Version 3.0, see Unicode Standard Version 3.0, published by Addison-Wesley, or on <http://www.unicode.com>.
- Oracle Unicode now supports the fixed-width character encoding. This enables easier loading of multinational data.

Database Character Sets and National Character Sets

Database Character Sets	National Character Sets
Defined at creation time	Defined at creation time
Can be changed without re-creation	Can be changed without re-creation
Store data columns of type CHAR, VARCHAR2, CLOB, LONG	Store data columns of type NCHAR, NVARCHAR2 and NCLOB
Can store varying-width character sets	Can store fixed-width and varying-width multibyte character sets

ORACLE

10-6

Copyright © Oracle Corporation, 2001. All rights reserved.

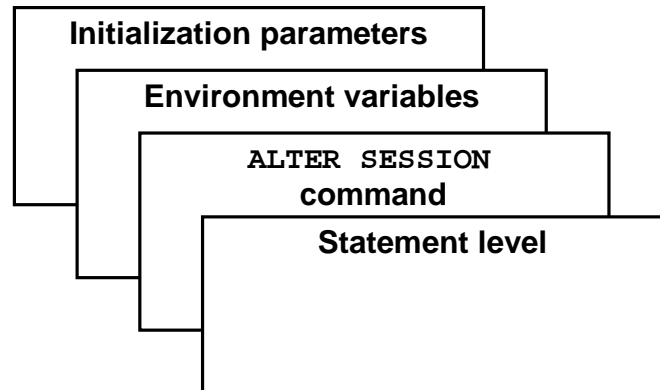
Database Character Sets and National Character Sets

When you create an Oracle9i database, you can specify a `DATABASE CHARACTER SET` and a `NATIONAL CHARACTER SET`. Because the database character set is used to identify and hold SQL and PL/SQL source code, it must have either EBCDIC or 7-bit ASCII as a subset, whichever is native to the platform. Therefore, it is not possible to use a fixed-width multibyte character set as a database character set.

The data types `NCHAR`, `NVARCHAR2` and `NCLOB` are provided to declare columns as variants of the basic types `CHAR`, `VARCHAR2` and `CLOB`, to note that they are stored using the national character set and not the database character set.

Note: In Oracle9i, the National Character Set (for `NCHAR`, `NVARCHAR2`, and `NCLOB` datatypes) will be limited to the Unicode character sets `AL16UTF16` and `UTF8` only.

Specifying Language-Dependent Behavior



ORACLE

10-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Specifying Language-Dependent Behavior

There are three ways to specify NLS parameters:

- Initialization parameters on the server side specify the default server NLS environment.
- Environment variables for the client specify locale-dependent behavior overriding the defaults set for the server.
- `ALTER SESSION` parameters override the default set for the session or the server.

These are some of the NLS parameters that can be changed:

- `NLS_LANGUAGE`
- `NLS_TERRITORY`
- `NLS_DATE_FORMAT`
- `NLS_SORT`
- `NLS_CURRENCY`
- `NLS_NUMERIC_CHARACTERS`
- `NLS_CALENDAR`
- `NLS_DUAL_CURRENCY`
- `NLS_TIMESTAMP_TZ_FORMAT`

Character Set Scanner

- **Used before converting database character sets**
- **Scans database character data for problems**
- **Does not convert data in the database**
- **Scans for any character conversion, not limited to Unicode**
- **Output is a report of potential conversion faults**
- **Available since Oracle8i Release 3 (8.1.7)**

ORACLE

10-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Character Set Scanner

This utility scans all the character data and the related column definitions in the database, to identify data loss if the data is converted to a new character set. It does not alter any character data.

When converting the character data in the database to a different character set, or redeclaring the character set, you should perform the following:

- Run the Character Set Scanner
- Correct the problems it finds, or find another solution than character set conversion
- After fixing the problem, perform the conversion

The version of the Character Set Scanner must match the database version. A version for Release 8.1.7 is distributed with the Oracle8i Release 8.1.7 database. Versions for earlier Oracle8i releases can be requested for backport through Oracle Support Services.

Oracle Locale Builder

A GUI-based tool to build your own locale:

- **Language**
- **Territory**
- **Character set**
- **Collation**
- **Does not include building own message files**

ORACLE

10-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Expanded Locale Coverage and User Definable Locales

Oracle9i provides a large number of locales. If you need another locale, or need to make an adjustment to an existing locale, you can use the Locale Builder. Viewing the definitions in the current supplied locales gives you an accurate and easy-to-read description, without having to understand the formatting of the relating files.

The globalization definitions are stored in files and are read by the server. You can use the Locale Builder to view and modify these. You can also create your own.

11

Oracle9i Business Intelligence

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

After completing this lesson, you should be able to explain some of the Data Warehousing and VLDB possibilities of Oracle9i:

- **Partitioning, parallel execution**
- **Summary management**
- **Extraction, transformation, and loading processes**
- **Transportable tablespaces, direct path loads**
- **OLAP services and data mining**

ORACLE

Partitioning

- **Makes VLDBs easier to manage**
- **Increases manageability, availability, and performance**
- **Transparent to users and applications**
- **Supported partition techniques:**
 - Range
 - Hash
 - Composite
 - List

ORACLE

11-3

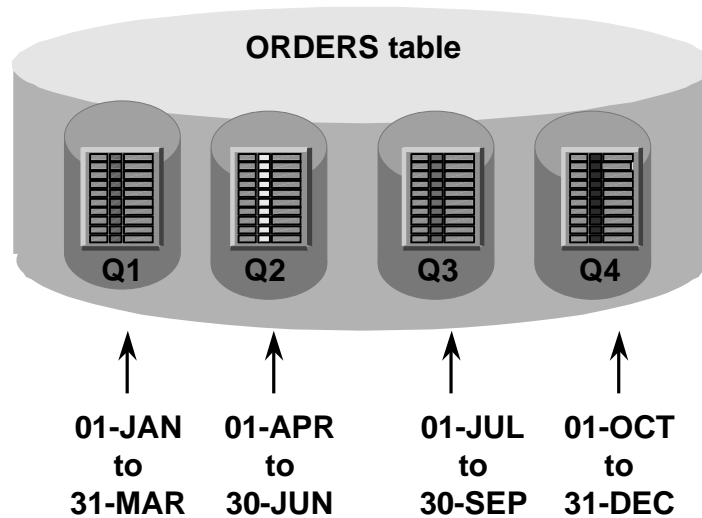
Copyright © Oracle Corporation, 2001. All rights reserved.

Partitioning

Partitioning stores large tables and indexes in pieces, instead of as one large monolithic object. Partitions are a *divide and conquer* technique that provides scalable performance with a large amount of data. Each partition can be individually managed, and can operate independently of the other partitions, increasing availability and making administrative tasks easier.

Oracle9i supports range, hash, composite, and list partitioning, which provides better manageability, availability, and performance.

Range Partitioning



ORACLE

11-4

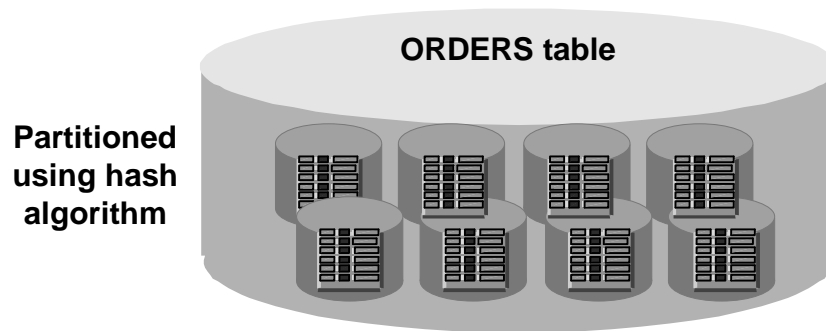
Copyright © Oracle Corporation, 2001. All rights reserved.

Range Partitioning

Range partitioning provides the ability to subset data into individual partitions, based on a range that the administrator defines. For example, an order table could be range-partitioned by sales date, putting each order record into one of four partitions, one for each quarter of the year. This method of partitioning data is useful when there is a logical range into which to divide the data, such as by quarter of the year. However, in this case, partition size could vary dramatically, which could affect performance of maintenance operations, if the data is skewed into one particular partition.

Hash Partitioning

- Insert rows into partition, based on hash of partition key
- Good for data striping and parallel DML



ORACLE

11-5

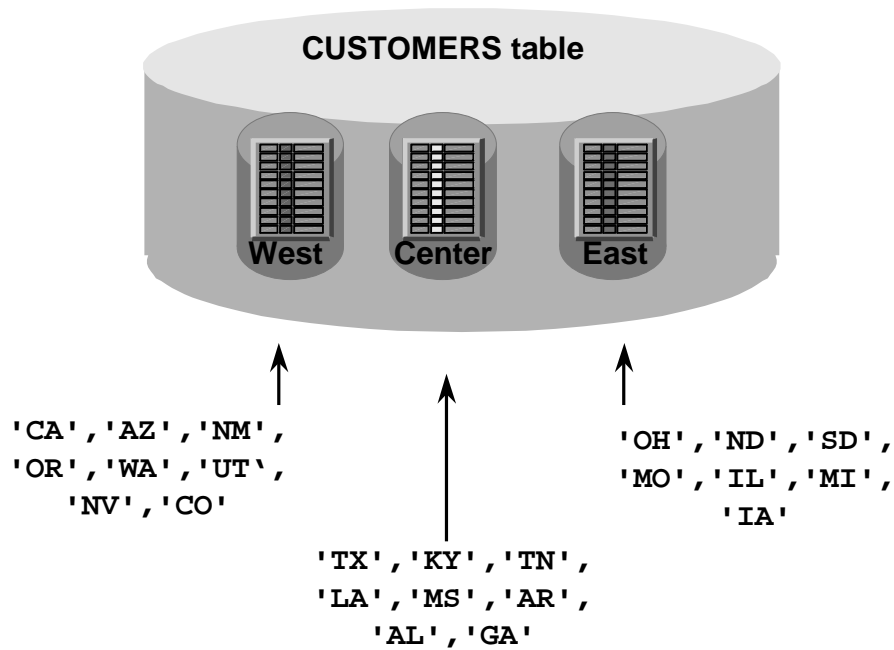
Copyright © Oracle Corporation, 2001. All rights reserved.

Hash Partitioning

Oracle9i hash partitioning is an alternative to range partitioning. Hash partitioning:

- Controls the physical placement of data across a *fixed* number of partitions.
- Uses a hash function on the partition columns to place data into required partitions, sometimes called *hash buckets*.
- Allows data that does not lend itself to range partitioning to be easily partitioned for predominantly performance reasons (PDML, partition elimination, piece-wise joins).

List Partitioning



ORACLE

11-6

Copyright © Oracle Corporation, 2001. All rights reserved.

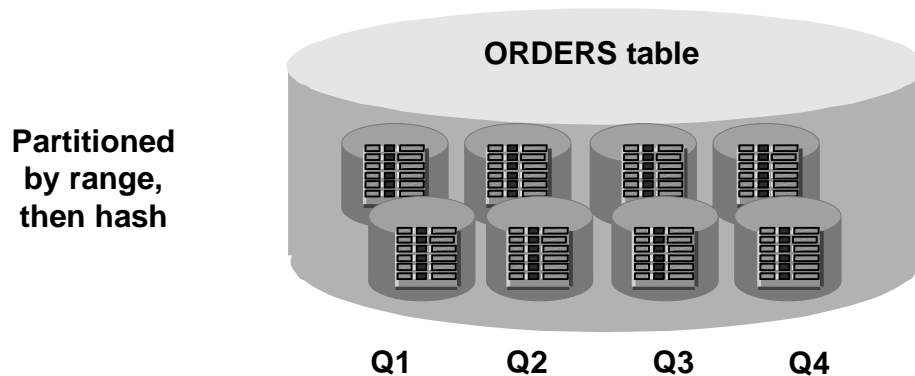
List Partitioning Overview and Benefits

The Partitioned Objects feature has incrementally added new partition methods to the Oracle RDBMS over two releases: Oracle8 and Oracle8i. However, there are constantly emerging requirements generated by customers who cannot take full advantage of Oracle8i partition methods provided so far, because their data model does not dovetail Oracle partition methods.

Thus, Oracle9i adds a new partitioning model called LIST partitioning to the set of partition methods already being supported in the Oracle RDBMS. LIST method allows explicit control over how rows map to partitions. This is done by specifying a list of discrete values for the partitioning column in the description for each partition. LIST partitioning is different from RANGE partitioning where a range of values is associated with a partition, and from HASH partitioning where the user has no control of the row-to-partition mapping. This partition method has been specifically added to model data-distributions that follow discrete values. This cannot be easily done with Oracle8i.

Composite Partitioning

- **Best of both range and hash partitioning**
- **Good for striping, parallel DML**
- **Manageability and availability of range partitioning**



ORACLE

11-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Composite Partitioning

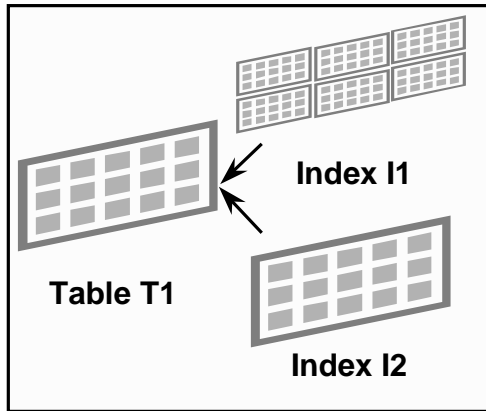
There is also a need for partitioning methods that are well-suited for historical data and simplify management for the support of parallelism. To satisfy this need, Oracle9i introduces composite partitioning. Composite partitioning partitions data using the range method, and within each partition, subpartitions it using the hash method. This new type of partitioning supports historical operations data at the partition level, and parallelism (PDML) and data placement at the sub-partition level.

In the example, the ORDERS table is first range partitioned on order date into four separate ranges, representing quarters. Each range partition is then explicitly subpartitioned into two hash partitions, on the PRODUCTID key.

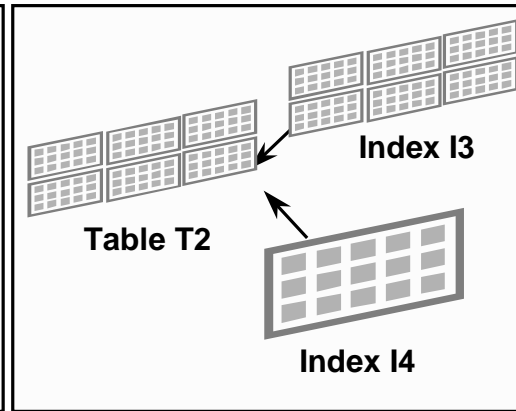
This results in eight individual subpartitions (4 x 2).

Partitioned Tables and Indexes

A nonpartitioned table can have partitioned or nonpartitioned indexes.



A partitioned table can have partitioned or nonpartitioned indexes.



ORACLE

11-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Partitioned Tables and Indexes

You can mix partitioned and nonpartitioned indexes with partitioned and nonpartitioned tables:

- A partitioned table can have partitioned and nonpartitioned indexes.
- A nonpartitioned table can have partitioned and nonpartitioned B*-tree indexes.
- Bitmap indexes on nonpartitioned tables cannot be partitioned.

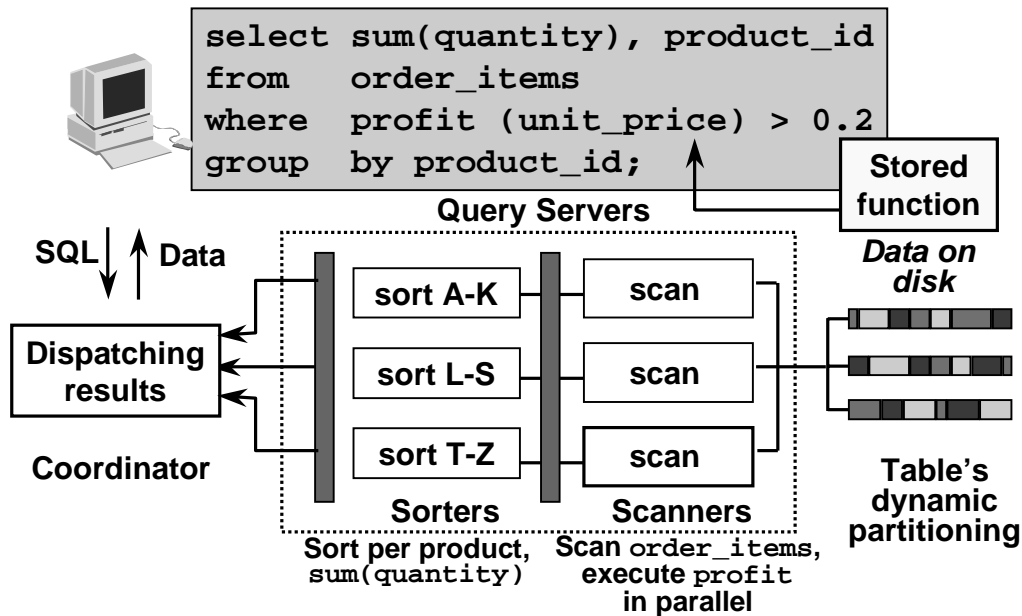
Partitioned indexes are more complicated than partitioned tables because there are four types of partitioned indexes: local prefixed, local nonprefixed, global prefixed, and global nonprefixed. These types are described below. Oracle supports three of the four types; global nonprefixed indexes are not useful in real applications.

In a local index, all keys in a particular index partition refer only to rows stored in a single underlying table partition. A local index is prefixed if it is partitioned on a left prefix of the index columns.

In a global index, the keys in a particular index partition may refer to rows stored in more than one underlying table partition or subpartition. A global index can only be range-partitioned, but it can be defined on any type of partitioned table (range, hash, or composite partitioned).

As a result of partition level maintenance operations indexes normally get invalidated; for example, all global indexes will have to be rebuilt. However, local indexes belonging to non-affected partitions will stay valid.

Parallel Execution



ORACLE

11-9

Copyright © Oracle Corporation, 2001. All rights reserved.

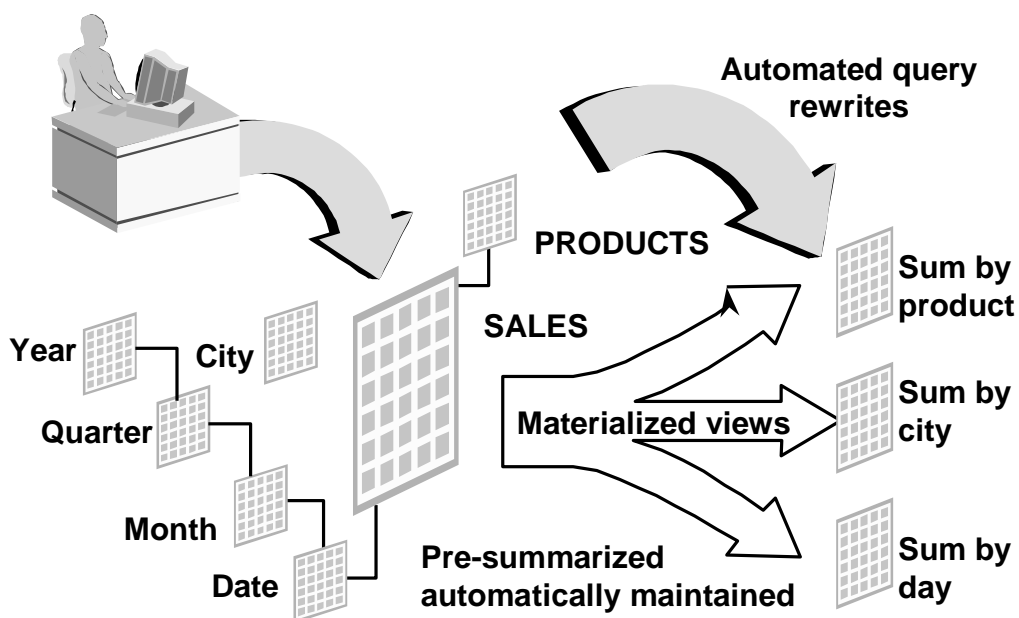
Parallel Execution

When Oracle is not parallelizing SQL statements, each statement is executed sequentially by a single process. With parallel execution, however, multiple processes work together simultaneously to execute a single SQL statement. By dividing the work among multiple processes, Oracle can execute the statement more quickly. Parallel execution can dramatically improve performance for data-intensive operations associated with DSS applications and VLDB environments.

Symmetric multiprocessing (SMP), clustered, and massively parallel systems (MPP) gain the largest benefits from parallel execution because statement processing can be split up among many CPUs. Parallel execution helps systems scale in performance by making optimal use of system resources. If your CPUs and disk controllers are already heavily loaded, you need to alleviate that load or increase hardware resources before using parallel execution to improve performance.

The Oracle9i server can use parallel execution for many operations, and each new release comes with new parallel execution enhancements.

Materialized Views Management



ORACLE

11-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Storing Aggregated Data in Materialized Views

Summaries are typically used in data warehouse or data mart environments. A materialized view can be used to significantly improve the query performance in these situations.

The optimizer may choose to rewrite a query on the base tables to access a materialized view. This process is known as a query rewrite and is only possible if the cost-based optimizer is used. Accessing a materialized view may be significantly faster than accessing the underlying base tables.

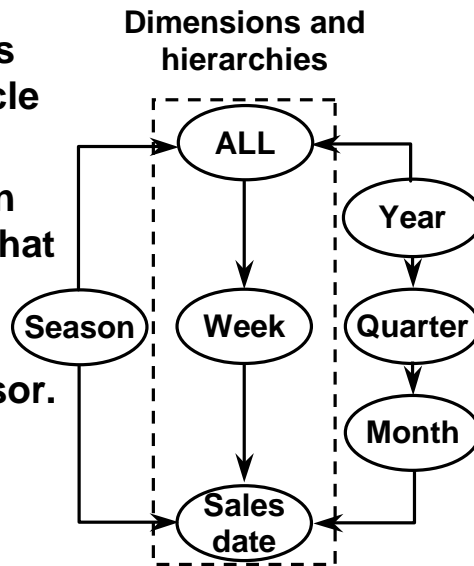
The existence and use of materialized views is transparent to applications. You do not need explicit privileges on materialized views to use them. Queries on the underlying tables can be rewritten to access the materialized view, if you have access privileges on the base tables.

A materialized view can be enabled or disabled for query rewrite. A materialized view that is enabled is available for rewrites. Materialized views are enabled by default.

A full refresh of a materialized view involves truncating existing data and reinserting all the data that is based on the detail tables. Fast refreshes using materialized view logs are possible. In this case, all changes to the base tables are captured in a log and then applied to the materialized view.

Making Full Use of Materialized Views

- Dimensions and hierarchies can be defined so that Oracle can rewrite queries.
- Materialized view tables can be used to satisfy queries that need *rollups*.
- Enterprise Manager provides a dimension advisor.



ORACLE

11-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Making Full Use of Materialized Views

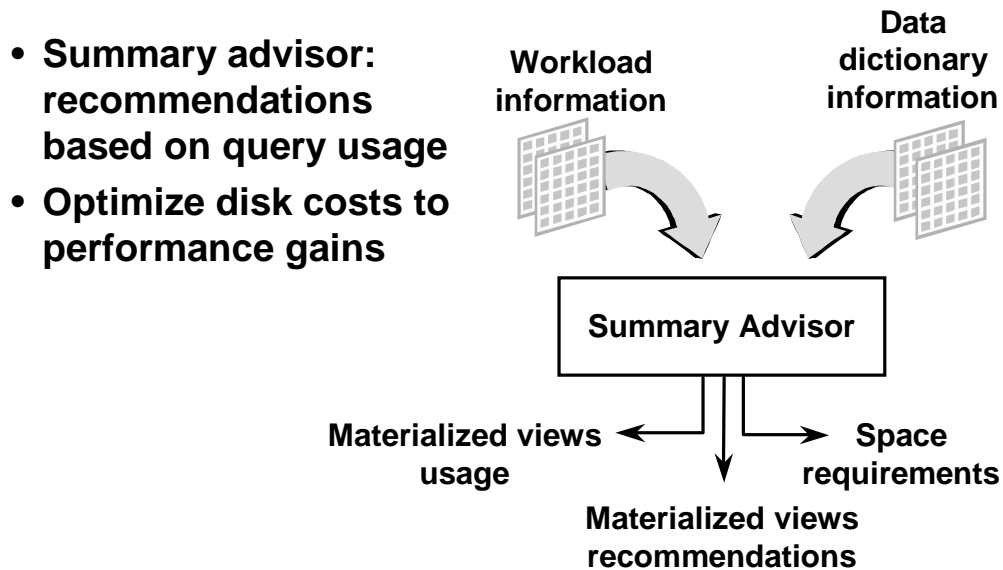
Dimensions describe business entities, such as products, departments, and time, in a hierarchical, categorized manner. A dimension may consist of one or more hierarchies. In the example shown, the time dimension consists of three hierarchies.

Consider the TIME table in the star schema. If ten years of Sales are being stored online, then the TIME table will have 3650 rows—one row for any possible sales date within the ten years. Columns in the time table represent attributes of the time dimension, such as week number, month, season, and so on. Relationships between the columns in the tables can be defined as a hierarchy of dimension attributes.

Each hierarchy consists of multiple levels. Each value at a lower level in the hierarchy is the child of one and only one parent at the next higher level. A hierarchy consists of 1:n relationships between levels, with the parent level representing a level of aggregation of the child level. In the example, the calendar hierarchy consists of sales date, month, quarter, and year.

Dimensions and Hierarchical relationships can be declared between columns in a dimension table, or between columns in different tables, in the case of normalized or “snowflake” schemas. A new schema object type, dimension, is provided for this purpose.

Management of Materialized Views



ORACLE

11-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Management of Materialized Views

Oracle9i includes a feature set called Summary Advisor, which contains several procedures and functions to manage Materialized Views (MV) in a data warehouse. The summary advisory functions within the package use two major sources of information:

- **Workload statistics:** Workload can be gathered from
 - Oracle Trace in the Enterprise Manager tuning pack, or through third party tools, can be used to collect workload statistics. A new event, MATERIALIZED VIEW USAGE, is available with Oracle Trace to collect this information.
 - SQL statements cached inside the SGA.
 - User generated workload stored in particular dictionary tables.
- **Data dictionary:** The data dictionary information that the advisory functions use include summary and dimension data.

Information that can be obtained from the summary advisor includes:

- **MV usage:** Such as the number of times a query rewrite was made to use a MV, the space used by a MV and a cost-benefit ratio for each MV.
- **MV recommendations:** Such as creation, retention, and dropping of MVs.
- **Space requirements:** Based on queries for possible MVs.

Explain Materialized View

- **New procedure `DBMS_MVIEW.EXPLAIN_MVIEW` accepts a:**
 - **Materialized view name, or**
 - **SQL statement**
- **Advises what is and is not possible with this MV or potential MV before you create it**
- **Results are stored in `MV_CAPABILITIES_TABLE`**

ORACLE

11-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Explain Materialized View

It is very difficult for a user to understand why fast refresh is not possible for a particular Materialized View (MV). The purpose of the “explain materialized view” procedure, available with Oracle9i, is to advise what is and is not possible with a given MV or potential MV. This package advises the user in the following ways:

- Is this materialized view fast refreshable ?
- What types of query rewrite Oracle can do with this MV ?

The process for using this package is very simple. The procedure `DBMS_MVIEW.EXPLAIN_MVIEW` is called, passing in as parameters, the schema and MV name for an existing MV. Alternatively, you can specify the select string for a potential MV. The MV or potential MV is then analyzed and the results are written into a table called `MV_CAPABILITIES_TABLE`.

Explain Query Rewrite

- **Goal is to explain:**
 - Why query did not rewrite?
 - Or, if rewrite is possible, which materialized view is used?
- **Use procedure `DBMS_MVIEW.EXPLAIN_REWRITE`**
- **Parameters**
 - SQL statement, or
 - Materialized view name
- **Results are stored in `REWRITE_TABLE` table**

ORACLE

11-14

Copyright © Oracle Corporation, 2001. All rights reserved.

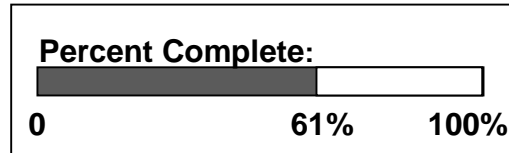
Explain Query Rewrite

Users of query rewrite do not have any tools to find out why query rewrite failed to occur or which materialized view is used in case query can be rewritten. They also don't have any ways to find out why query rewrite algorithm did not choose a particular materialized view, even though it may appear to be the right choice. The rules governing query rewrite eligibility are quite complex, involving various factors such as constraints, dimensions, query rewrite integrity modes, freshness of the materialized views, and the types of queries themselves. In addition, you may want to know why query rewrite chose a particular materialized view instead of another.

To help with this matter, Oracle9i provides a PL/SQL procedure (`DBMS_MVIEW.EXPLAIN_REWRITE`) to advise you when a query can be rewritten and, if not, why not. Using the results from `DBMS_MVIEW.EXPLAIN_REWRITE`, you can take the appropriate action needed to make a query rewrite if at all possible.

Monitor Long-Running Operations

- **Track progress of:**
 - Archiving, backup, and restore
 - Parallel query
 - Recovery
 - Sorts
 - Full table scans
 - Analyze (DBMS_STATS)
- **Applications can populate the `v$sqlsession_longops` view.**



ORACLE

11-15

Copyright © Oracle Corporation, 2001. All rights reserved.

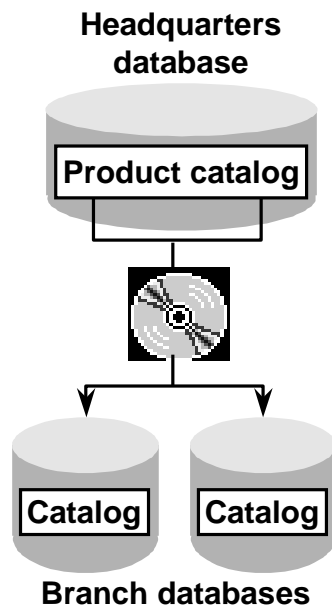
Monitor Long-Running Operations

As databases become large, certain operations, such as complex queries, index builds, and other DDL operations, can take a long time to complete. During a prolonged database operation, users often have no way of knowing if the database server is doing any work or is hung.

In such situations, it is nice to have a mechanism for users and DBAs to monitor operations progress to get an idea of the estimated completion time. To provide such a mechanism, Oracle9i maintains statistics to track the progress of these operations, and makes them available to the users via the dynamic performance view `V$SESSION_LONGOPS`.

The package `DBMS_APPLICATION_INFO` contains a procedure that allows application developers to insert statistics about their own custom long running operations.

Transportable Tablespaces



- **Copy tablespaces between databases:**
 - Same OS
 - Same DB version
- **No metadata export/import required**
- **Exact full copies only**

ORACLE

11-16

Copyright © Oracle Corporation, 2001. All rights reserved.

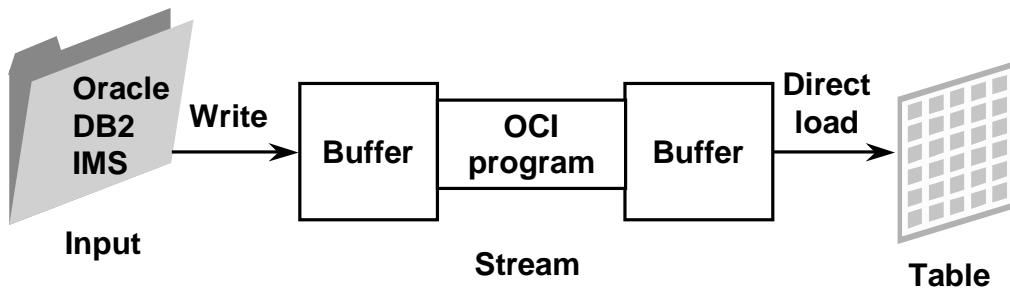
Transportable Tablespaces

Moving data from a data warehouse to a data mart, or from an OLTP system to a staging area for a data warehouse, can be cumbersome and time-consuming. Direct path loading via SQL*Loader or parallel DML makes the task faster, but the process should be simpler for data movement between identical databases. Oracle9i provides a mechanism for copying data files between identical systems and allows both systems to access the same data. Now data movement can be as fast as a simple transfer of files between machines. This greatly improves performance and provides operational simplicity for the transfer of data.

Companies can also use this feature to publish data on CD ROMs that can be easily integrated into Oracle databases at regional or area offices. For example, PRODUCT tables with product descriptions and price can be populated or updated at the head office and moved to regional or branch offices, for use in an order processing system.

Fast and Direct Data Loading

- **Complete access to all load functionality through Oracle9i Call Interface (OCI) API**
- **Partners and customers can program powerful load programs that load data through a stream instead of a file.**



ORACLE

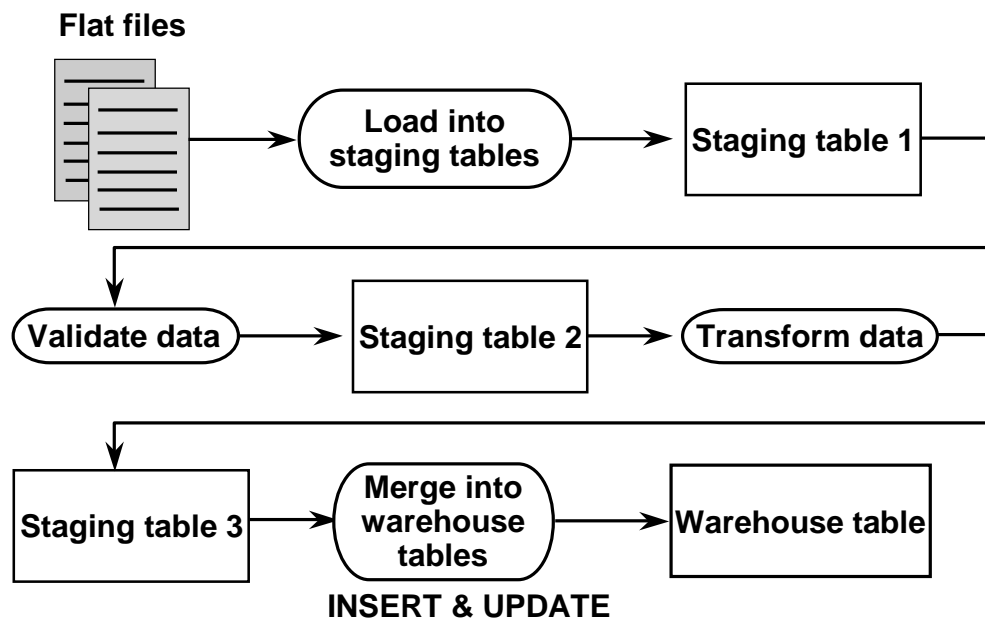
11-17

Copyright © Oracle Corporation, 2001. All rights reserved.

Data Transportation Direct-path Load API

Oracle9i provides an Application Programming Interface (API) in OCI to the direct path load mechanism. This provides a way for Independent Software Vendors and system management tool partners to create easy-to-use and high-performance customized data-loading tools. Access to all load functionality is available through the API, allowing performance of any data loading tool to be comparable to SQL*Loader.

Common Extraction, Transformation, and Loading (ETL) Process



ORACLE

11-18

Copyright © Oracle Corporation, 2001. All rights reserved.

Common Extraction, Transformation, and Loading (ETL) Process

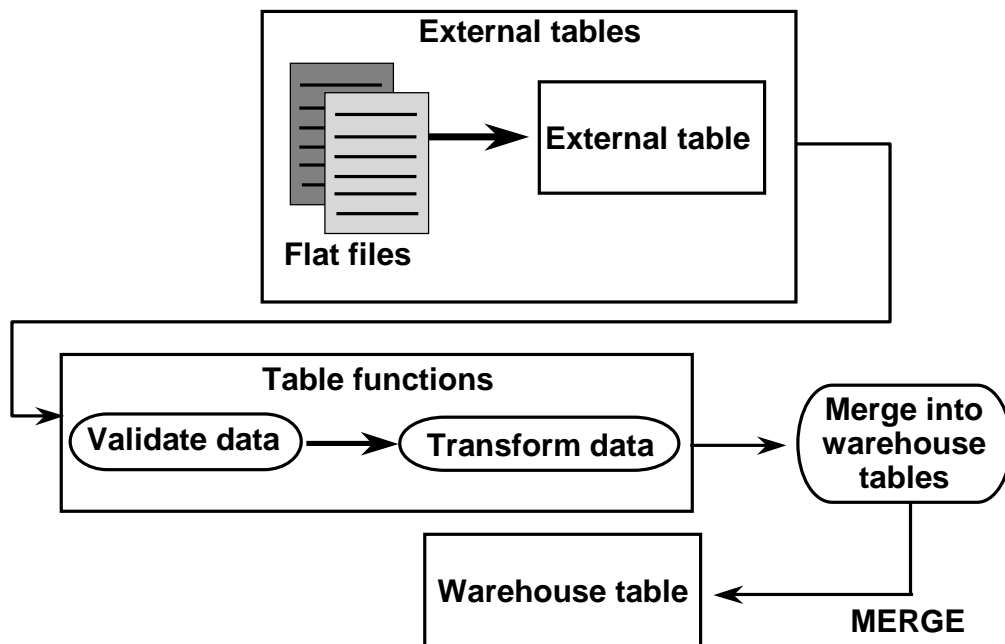
Data transformations are often the most complex and, in terms of processing time, the most costly part of the Extraction Transformation Loading (ETL) process. They can run the gamut from simple data conversions to extremely complex data scrubbing techniques. Many, if not all, data transformations can 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:

- Multi-Stage Data Transformation
- Pipelined Data Transformation

With Multi-Stage 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 may 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 to the entire transformation process, which enables the process to be more easily monitored and restarted. However, a disadvantage is that, due to the multistaging, the required space and time increases.

Pipelined Data Transformation in Oracle9i



ORACLE

11-19

Copyright © Oracle Corporation, 2001. All rights reserved.

Pipelined Data Transformation in Oracle9i

With the introduction of Oracle9i, Oracle's database capabilities have been significantly enhanced to address specifically some of the tasks in 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 former necessary process steps become obsolete while some others can be remodeled to enhance the data flow and the data transformation to become more scalable and noninterruptive. We are no longer talking about serial transform-then-load (with most of the tasks done outside the database) or load-then-transform; rather we are talking about an enhanced transform-while-loading.

Oracle9i offers a wide variety of new capabilities to address all the issues and tasks relevant in an ETL scenario. It is important to understand that the database offers toolkit functionality rather than trying to address a one-size-fits-all solution. The underlying database has to enable the most appropriate 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 throughout later slides in this lesson.

Oracle9i OLAP Services

- **Analysis-ready Oracle database**
 - Support for complex, multidimensional queries, such as Express server
 - Highly scalable
- **Development platform for internet-ready analytical applications:**
 - Java OLAP API
 - Business Intelligence Beans and JDeveloper

ORACLE

11-20

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i OLAP Services

When people think of OLAP, they typically think of multidimensional databases which are separate and apart from the relational databases that collect and warehouse data to be analyzed. Although multidimensional database have been the most effective OLAP solution to date, the added design and administrative costs have limited their reach into most organizations.

With Oracle9i, the Oracle database is not only the data source and the historical data store, it is an OLAP ready relational database and the platform for analytical applications.

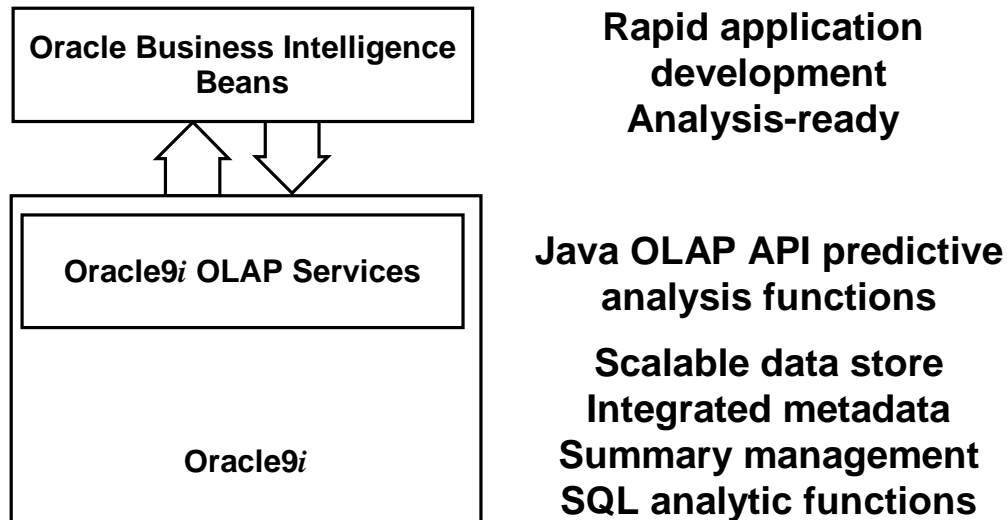
Oracle9i combines the best of Express Server - support for complex multidimensional queries and rapid query response - with the highly scaleable, manageable, and secure Oracle database. Oracle9i is the first OLAP-ready relational database.

Oracle9i OLAP Services provides the query performance and calculation capability of a multidimensional database. In addition, it provides a Java OLAP API that is appropriate for the development of internet-ready analytical applications.

Unlike other marriages of OLAP and RDBMS technology, Oracle9i OLAP Services is not a thinly disguised multidimensional database using bridges to move data from the relational data store to a multidimensional data store. Instead, it is truly an OLAP enabled relational database.

As a result, Oracle9i provides the performance and calculations of a multidimensional database along with the scalability, accessibility, security, manageability, and high availability of the Oracle9i database. In addition, developers can use the Java OLAP API, which specifically designed for internet based analytical applications.

OLAP Application Platform



11-21

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

OLAP Application Platform

Oracle9i OLAP Services OLAP fully enables the Oracle database. Oracle9i consists of:

- The Oracle9i RDBMS, which provides:
 - Scaleable data storage
 - Summary management with materialized views
 - Meta data stored in the RDBMS and managed by Oracle Enterprise Manager
 - New OLAP SQL functions which provide analytical features and improve performance
- Oracle9i OLAP Services, which provides the support for analytical applications by providing:
 - An Java OLAP API designed for the internet
 - Predictive OLAP functions.

In addition, the Business Intelligence Beans provides the OLAP ready application building blocks to support rapid application development of internet-based business intelligence applications.

Together Oracle9i and the Business Intelligence Beans provide a complete platform for analytical applications without any of the drawbacks of multidimensional databases.

12

Backup and Recovery

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

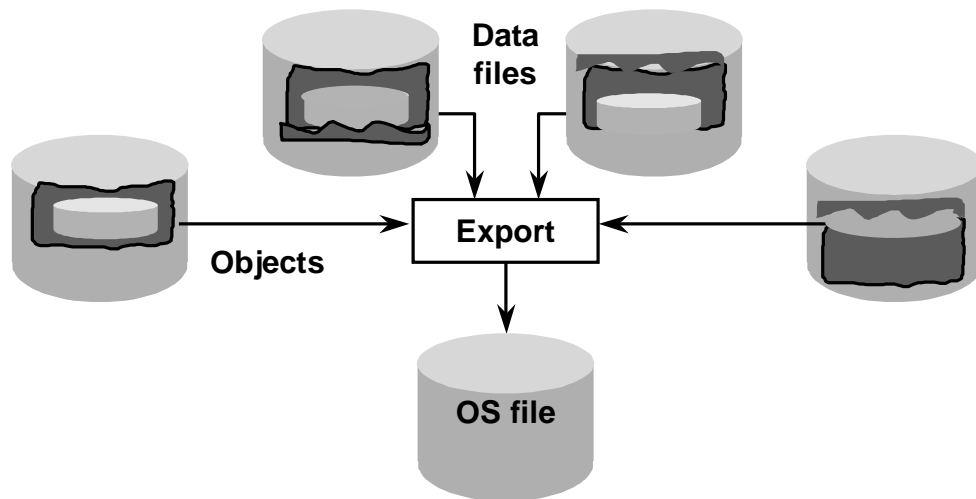
Objectives

After completing this lesson, you should be able to discuss the basic principles of backing up and recovering an Oracle9i database.

- **Backup strategies**
- **Backup types**
- **Recovery manager (RMAN)**
- **Data guard**
- **LogMiner**

ORACLE

Data Export



Create a “SQL” script (binary file)

ORACLE

12-3

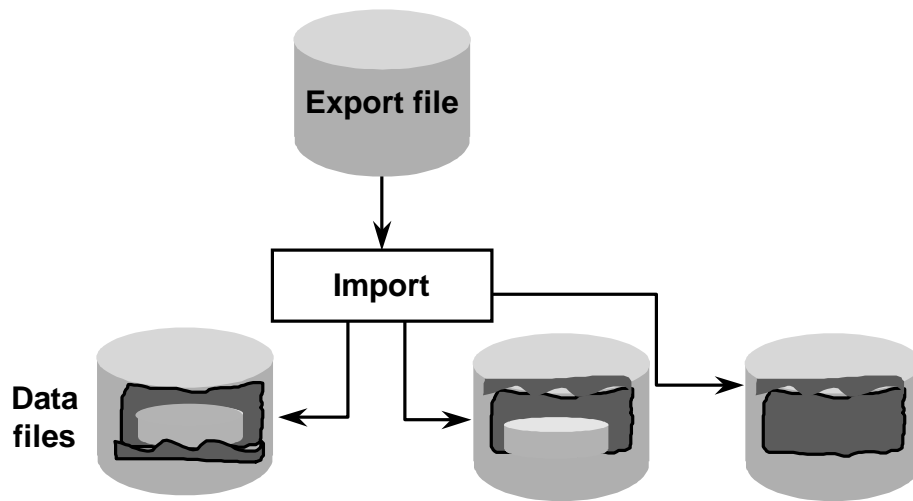
Copyright © Oracle Corporation, 2001. All rights reserved.

Data Export or Logical Backups

The Export utility provides a simple way for you to transfer data objects between Oracle databases, even if they reside on platforms with different hardware and software configurations. Export extracts the object definitions and table data from an Oracle database and stores them in an Oracle binary-format Export dump file located typically on disk or tape. Such files can then be copied through FTP or physically transported (in the case of tape) to a different site and used, with the Import utility, to transfer data between databases that are on machines not connected through a network, or as backups in addition to normal backup procedures.

When you run Export against an Oracle database, objects such as tables are extracted, followed by their related objects (such as indexes, comments, and grants) if any, and then written to the Export file.

Data Import



ORACLE

12-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Data Import

The Import utility inserts the data objects extracted from one Oracle database by the Export utility (and stored in an Export dump file) into another Oracle database. Export dump files can only be read by Import.

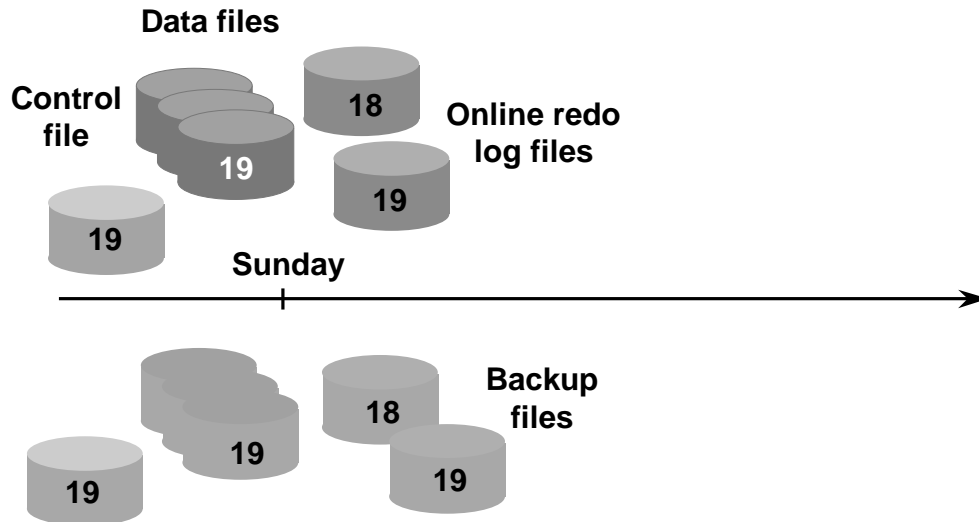
Import reads the object definitions and table data that the Export utility extracted from an Oracle database and stored in an Oracle binary-format Export dump file located typically on disk or tape.

The Export and Import utilities can also facilitate certain aspects of Oracle Advanced Replication functionality, such as offline instantiation.

If you are using Export/Import as a backup and recover utility, you must understand that you will lose data changes between the export time and import time.

For this reason, you should not rely solely on these utilities for your backup strategies.

NOARCHIVELOG Mode: Offline Backups Only



ORACLE

12-5

Copyright © Oracle Corporation, 2001. All rights reserved.

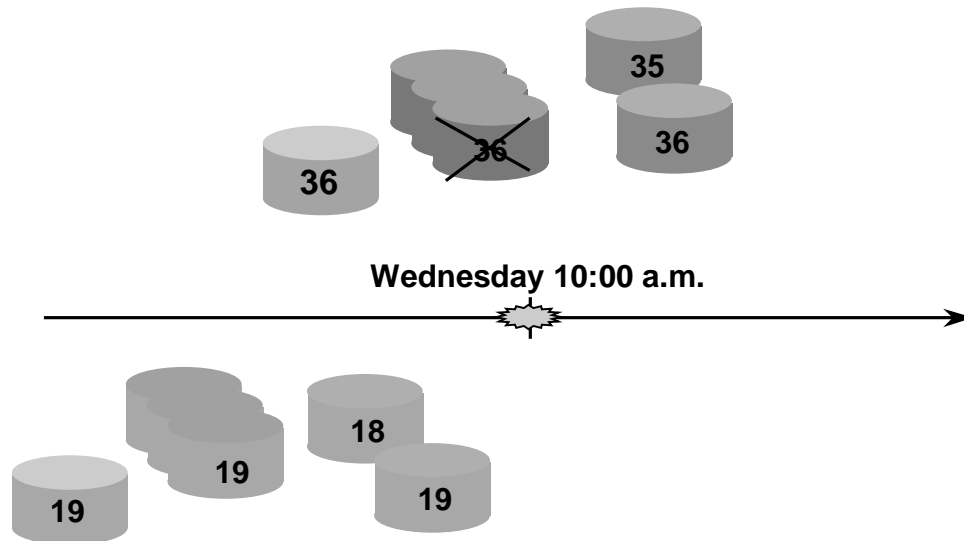
NOARCHIVELOG Mode

An Oracle database can operate in two distinct modes: NOARCHIVELOG mode (media recovery disabled) or ARCHIVELOG mode (media recovery enabled).

If a database is used in NOARCHIVELOG mode, the archiving of the online redo log is disabled. Information in the database's control file indicates that filled groups are not required to be archived. Therefore, as soon as a filled group becomes inactive, the group is available for reuse by the LGWR process. NOARCHIVELOG mode protects a database only from instance failure, not from disk (media) failure. Only the most recent changes made to the database, stored in the groups of the online redo log, are available for crash recovery or instance recovery. This is sufficient to satisfy the needs of crash recovery and instance recovery, because Oracle will not overwrite an online redo log that might be needed until its changes have been safely recorded in the data files. However, it will not be possible to do media recovery. This slide demonstrates the only possible way to restore and recover your database in case of a media crash. Here, a complete cold backup is made, which means that the database is shut down cleanly and that you back up all the files pertaining to the database. This includes the redo log, and control and data files.

Note: This page and the following two belong together, and illustrate a typical NOARCHIVELOG scenario.

Loss of a Data File



ORACLE

12-6

Copyright © Oracle Corporation, 2001. All rights reserved.

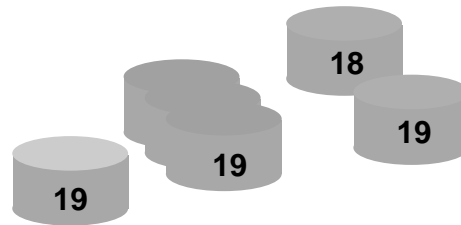
Loss of a Data File

Assume that a media crash occurs at some point in time after the backup. In the example, one of the data files was removed from the database configuration.

Note that the log sequence number at the time of the backup was 19 and is 36 at the time of the crash. This means that you have lost the transaction history since the backup, because redo log 19, 20, and so on, have been overwritten.

Restore

**Restore Sunday's
*whole backup***



Wednesday 10:30 a.m.



Changes since Sunday lost

ORACLE

12-7

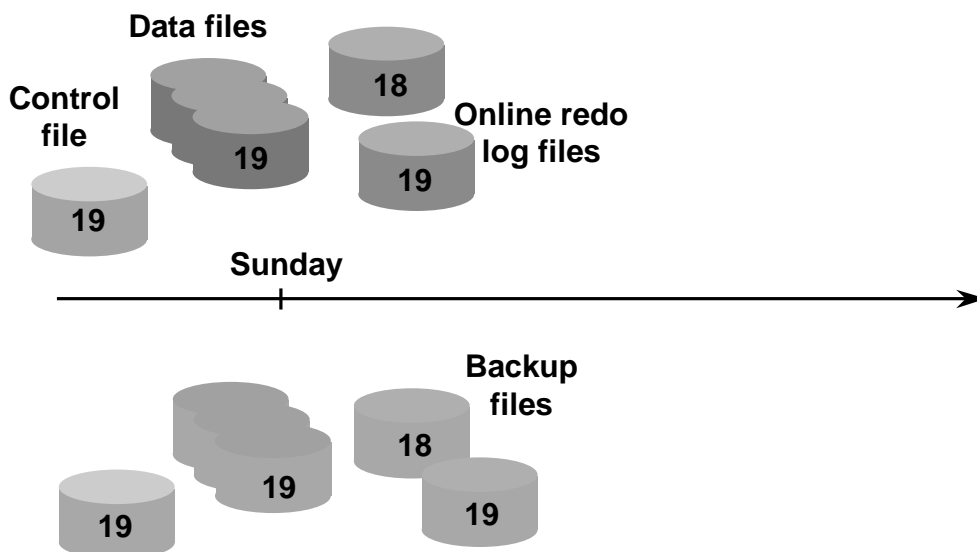
Copyright © Oracle Corporation, 2001. All rights reserved.

Restore

Because you lost the transaction history, you cannot recover from this situation.

The only possibility is to restore the full backup from Sunday. The consequence is that you have lost every data change since Sunday.

ARCHIVELOG Mode: Offline and Online Backups



ORACLE

12-8

Copyright © Oracle Corporation, 2001. All rights reserved.

ARCHIVELOG Mode (Media Recovery Enabled)

If an Oracle database is operated in ARCHIVELOG mode, the archiving of the online redo log is enabled. Information in a database control file indicates that a group of filled online redo log files cannot be reused by LGWR until the group has been archived.

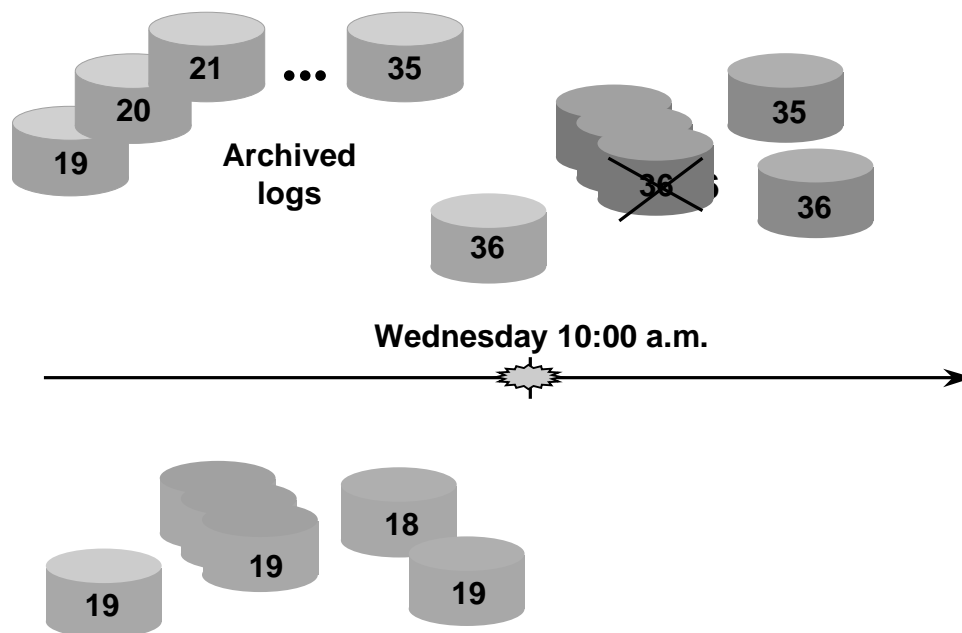
ARCHIVELOG mode permits complete recovery from disk failure as well as instance failure, because all changes made to the database are saved in an archived redo log.

Automatic Archiving and the ARCn (Archiver) Background Processes

You can configure an instance with an additional background process, the archiver (ARC0), to automatically archive groups of online redo log files once they become inactive. Automatic archiving frees the database administrator from having to keep track of, and archive, filled groups manually. For this convenience alone, automatic archiving is the choice of most database systems that run in ARCHIVELOG mode. For heavy workloads, such as bulk loading of data, multiple archiver processes (up to 10) can be configured. However, the database administrator can interactively start or stop automatic archiving at any time.

Note: On the above slide, the same example as before is used, but now the database is in ARCHIVELOG mode. This page and the following three belong together, and illustrate a typical ARCHIVELOG scenario.

Loss of a Data File



ORACLE

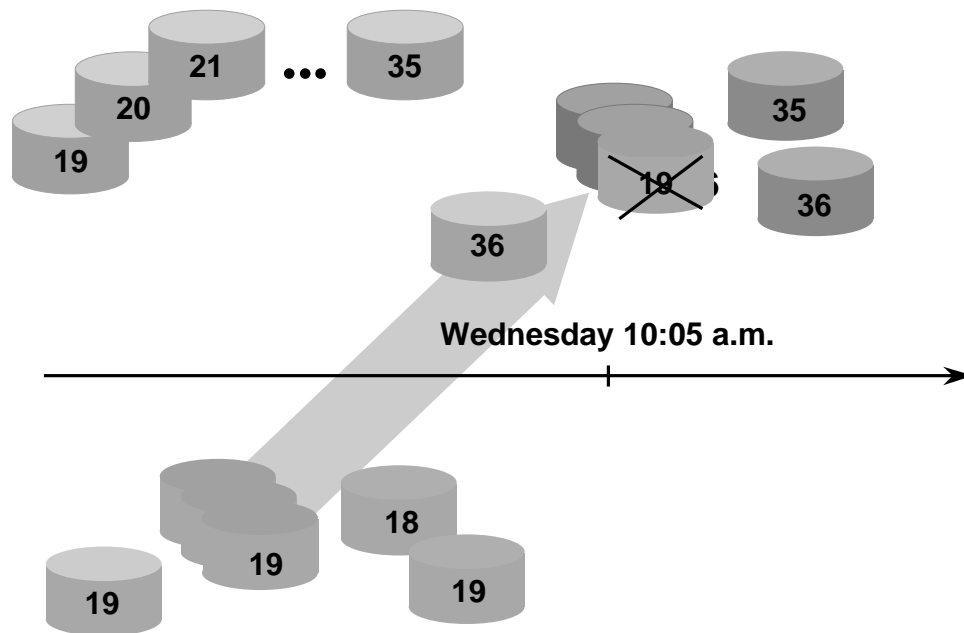
12-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Loss of a Data File

Note that now, each time a redo log group is full, the ARC0 background process archives it to disk so that you have the transaction history.

Partial Restore



ORACLE

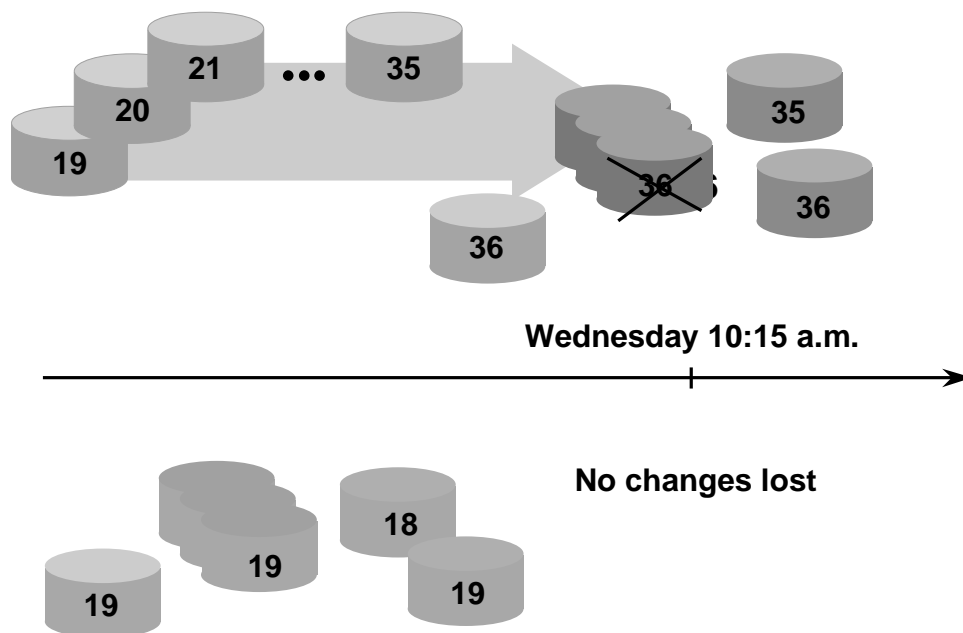
12-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Partial Restore

Now, when there is a media failure, you only need to restore the crashed data file to restore the database.

Datafile Recovery



ORACLE

12-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Recovery Concept

The idea is to recover from the crash by applying the transaction history to the crashed data file. The RECOVER command will do that automatically. At the end of the recovery process, the crashed datafile will be usable again. So, in this case, only the noncommitted transactions relative to this file are lost.

Trial Recovery

- **Several problems may prevent media recovery from completing successfully.**
- **In Oracle9i, the following enhancements are implemented:**
 - DBAs can invoke a trial recovery.
 - DBAs can instruct media recovery to mark a data block *corrupt*, if required to proceed.
 - Recovery always leaves behind a database that can be opened.
- **These enhancements are based on an optimistic redo application algorithm.**

ORACLE

12-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Trial Recovery

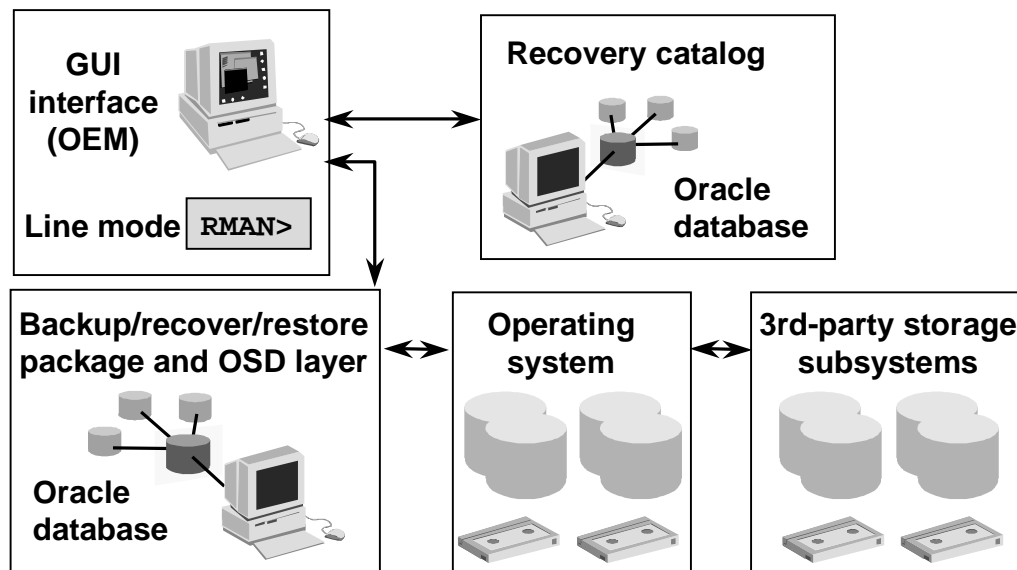
Some problems that may occur during media recovery are not recoverable. For example, if a redo log is corrupted and recovery information cannot be applied, and the recovery process stops, then the resulting database is useless. For media recovery, the user must restore the backup and recover the database to a SCN before where the corruption occurred. For a large database, restoring a backup and recovering the database can take a long time.

The following enhancements are provided:

- If media recovery of a full database encounters a problem, recovery always leaves behind a consistent database, which can be opened read-only or with resetlogs.
- The DBA can instruct the media recovery process to mark a data block as being corrupted. The block's inconsistency is ignored and the recovery can proceed. The block will subsequently be inaccessible.
- The DBA can invoke a Trial Recovery in order to investigate if the recovery problem is an isolated problem.

The optimistic redo application algorithm assumes that no problem will occur during media recovery. If any problem does occur, the Oracle server will automatically undo the last applied changes, and not leave an inconsistency in the recovered database.

Recovery Manager (RMAN)



ORACLE

12-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Recovery Manager

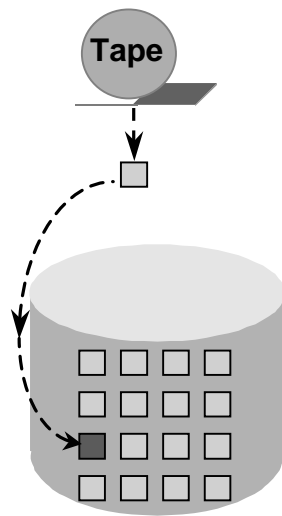
Recovery Manager (RMAN) is a utility that manages the processes of creating backups of all database files (datafiles, control files, and archived redo log files) and restoring or recovering files from backups.

Recovery Catalog

RMAN maintains a repository called the recovery catalog, which contains information about backup files and archived log files. RMAN uses the recovery catalog to automate both restore operations and media recovery. The recovery catalog is maintained solely by RMAN. The database server of the backed-up database never accesses the recovery catalog directly. RMAN propagates information about backup datafile sets, archived redo logs, backup control files, and datafile copies into the recovery catalog for long-term retention. When doing a restore, RMAN extracts the appropriate information from the recovery catalog and passes it to the database server. The server performs various integrity checks on the input files specified for a restore. Incorrect behavior by RMAN cannot corrupt the database. The recovery catalog is stored in an Oracle database. It is the database administrator's responsibility to make such a database available to RMAN. Taking backups of the recovery catalog is also the database administrator's responsibility. Since the recovery catalog is stored in an Oracle database, you can use RMAN to back it up. If the recovery catalog is destroyed and no backups are available, then it can be partially reconstructed from the current control file or control file backups.

Note: Use of a recovery catalog is not required, but is recommended. Since most information in the recovery catalog is also available from the control file, RMAN supports an operational mode where it uses only the control file. This operational mode is appropriate for small databases.

Block Media Recovery (BMR)



- A block becomes the smallest unit of media restore and recovery.
- BMR main benefits:
 - Lowers the mean time to recover
 - Increases data availability during media recovery
- RMAN must be used for BMR:
 - Restores individual data blocks from available backups
 - Coordinates with the server to have them recovered

ORACLE

12-14

Copyright © Oracle Corporation, 2001. All rights reserved.

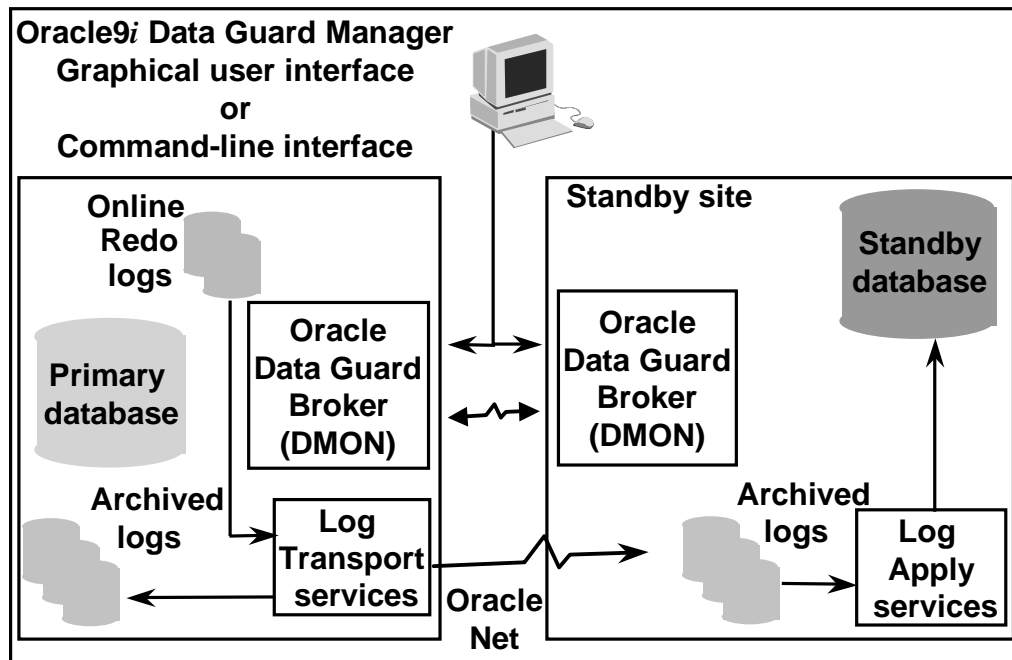
Block Media Recovery (BMR)

BMR reduces the smallest recoverable unit of media recovery from a data file to a block. The basic premise is that when a small subset of blocks in the database are known to require media recovery, it is more efficient to selectively restore and recover just those blocks. Only blocks being recovered need to be unavailable, allowing continuous availability of the rest of the database during recovery. Block media recovery (BMR) provides two main benefits over file-level recovery:

- Lowering the mean time to recover (MTTR)
- Allowing increased availability of data during media recovery as the data file being recovered remains online.

BMR uses existing recovery mechanisms to apply changes from the redo stream to block versions restored from suitable backups. Recovery Manager (RMAN) must be used to perform BMR. The existing SQL interface for media recovery does not support BMR. RMAN restores individual data blocks from available backups and coordinates with the Oracle server process to have them recovered. If a backupset repository is maintained, RMAN can also assist in selecting the correct backup of a block to use as the starting point for media recovery.

Oracle9i Data Guard Architecture



ORACLE

12-15

Copyright © Oracle Corporation, 2001. All rights reserved.

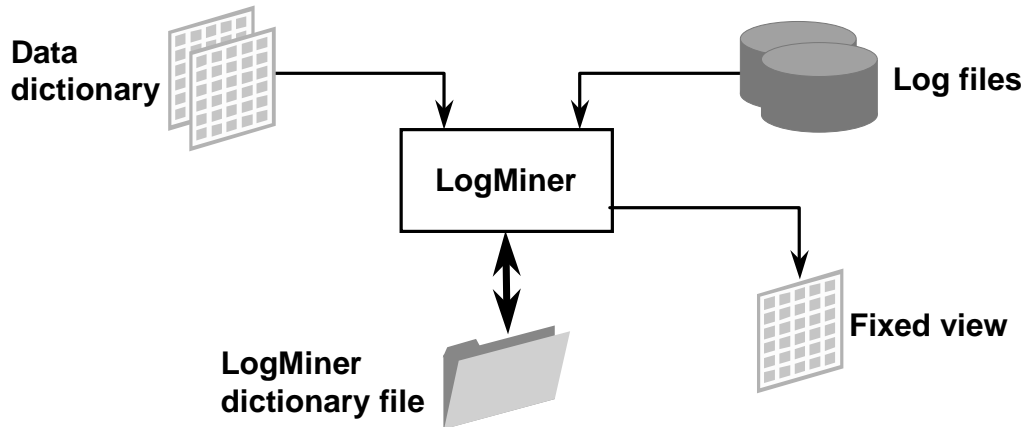
Oracle9i Data Guard Architecture

The Oracle9i Data Guard architecture consists of the following components:

- **Primary database:** A primary database is a production database. The primary database is used to create a standby database.
- **Physical standby database:** A physical standby database is a database replica created from a backup of a primary database and is in read-only mode or in recovery mode.
- **Log transport services:** Log transport services control the automated transfer of archived redo logs from the primary database to the standby site.
- **Network configuration:** The primary database is connected to the standby database through Oracle Net.
- **Log apply services:** Log apply services apply the archived redo logs to the standby database if not in read-only mode.
- **Data Guard broker:** Data Guard broker is the management and monitoring component that helps you configure, control, and monitor a fault tolerant system consisting of a primary database protected by one or more physical standby databases. After it has created the Data Guard configuration, the broker monitors the activity, health, and availability for all systems in the Data Guard configuration. You can control the Data Guard Broker by using the Oracle9i Data Guard Manager tool (GUI or CLI versions).

Analyzing Redo Log Files

- Track changes to database
- Undo changes to the database
- Perform tuning and capacity planning



ORACLE

12-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Analyzing Redo Log Files

Log files contain a wealth of useful information about the activities and history of an Oracle database. LogMiner is a PL/SQL interface that enables redo log files to be read, analyzed, and interpreted using standard SQL. Analysis of log files is useful for:

- Tracking changes to the database without the overhead of auditing. Value-based auditing can be performed offline without directly affecting the online performance of the database.
- Performing granular logical recovery by undoing specific changes (except DDL commands) made by one or more transactions. This minimizes the need for performing point-in-time recovery to recover from a logical application error.
- Tuning and capacity planning. Since the redo log files record changes to the database, this information can be analyzed to study usage patterns and to estimate data growth patterns.

LogMiner Features

LogMiner provides a procedure to analyze and present the data in the form of a virtual table. Since the redo log files only contain object identifiers and not object names, to enable interpretation of the object identifiers, the data dictionary information must be available at the time the log files are analyzed.

13

Tools to Administer Oracle9i

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

After completing this lesson, you should be able to discuss the tools used to administer an Oracle9i database:

- **SQL*Plus**
- **iSQL*Plus**
- **Enterprise Manager**
- **Export, Import**
- **SQL*Loader**
- **Online table redefinition**
- **SNMP support**

ORACLE

13-2

Copyright © Oracle Corporation, 2001. All rights reserved.

SQL*Plus

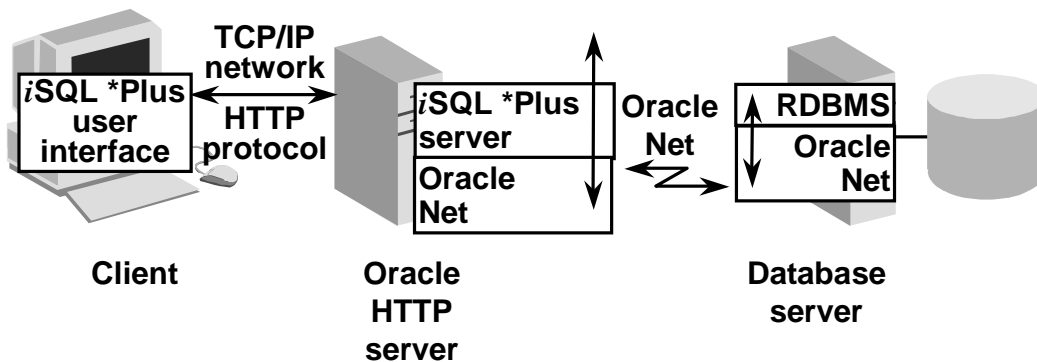
SQL*Plus has always been one of the most efficient tools to issue ad hoc SQL statements for any purpose, including database administration tasks such as managing tablespaces, rollback segments, users and schemas, privileges and roles, and auditing.

Oracle8i, SQL*Plus also supports database administration commands, including STARTUP and SHUTDOWN, RECOVER, and ARCHIVE LOG. This means that a database administrator can perform all regular tasks using SQL*Plus.

iSQL*Plus Architecture

Three tier implementation:

- **iSQL*Plus user interface (Web browser)**
- **iSQL*Plus Server, Oracle HTTP Server, Oracle Net**
- **Oracle9i**



ORACLE

13-3

Copyright © Oracle Corporation, 2001. All rights reserved.

iSQL*Plus Architecture

iSQL*Plus is a browser-based implementation of SQL*Plus. It uses no client software, just a Web browser. It is a component of SQL*Plus and is available on Windows NT/2000. In the near future, it will also be available on other platforms. The three tiers may be on the same machine, or on separate machines.

iSQL*Plus User Interface (Client Tier)

The iSQL*Plus user interface runs in a Web browser connected to the Internet or your intranet. You only need to know the URL of the Oracle HTTP Server to access Oracle9i, or any Oracle database. There is no installation or configuration required for the iSQL*Plus user interface other than a Web browser.

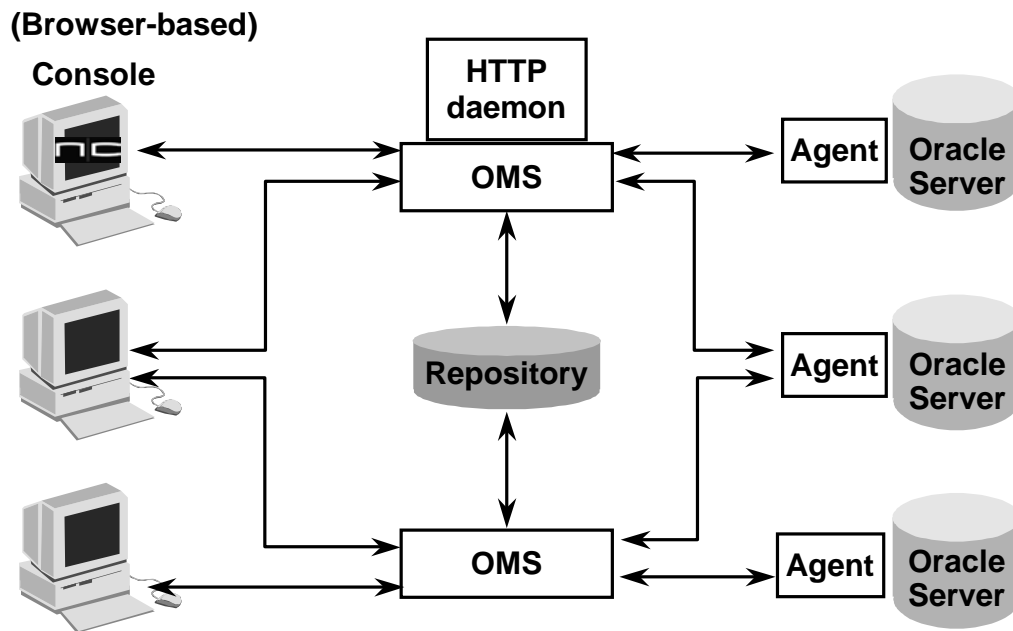
Oracle HTTP Server (Middle Tier)

The iSQL*Plus Server is installed as part of the Oracle HTTP Server. The iSQL*Plus Server enables communication and authentication between the iSQL*Plus user interface and the RDBMS.

Oracle9i (Database Tier)

Oracle Net components provide communication between the iSQL*Plus Server and Oracle9i.

Enterprise Manager Architecture



ORACLE

13-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Enterprise Manager Architecture

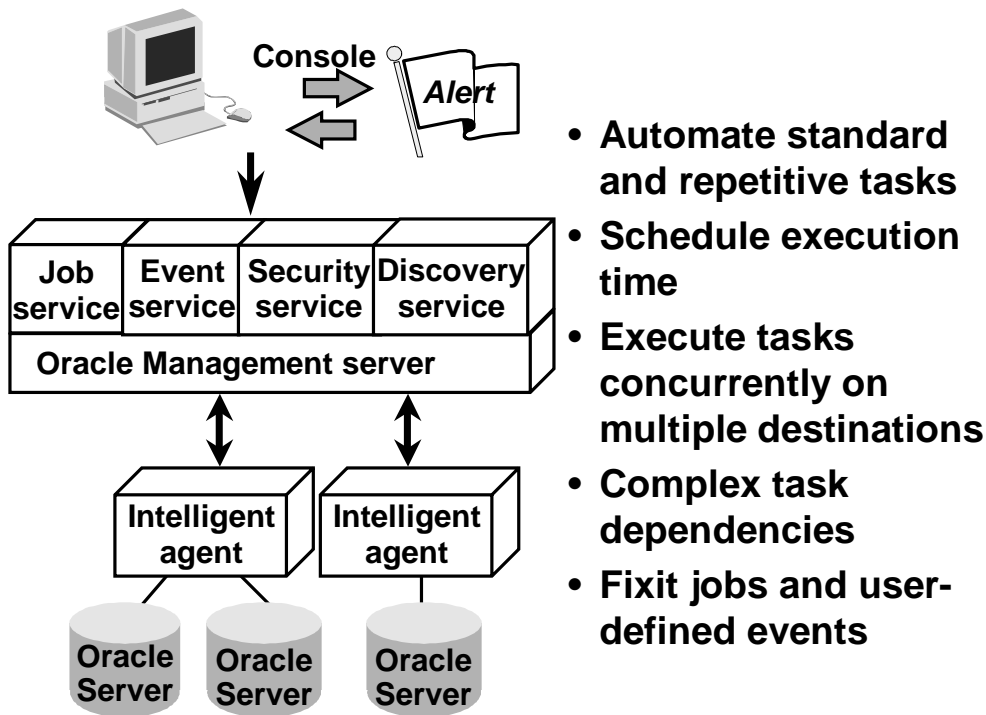
Enterprise Manager (EM) is a Java-based framework consisting of multiple components that integrate into a powerful graphical user interface (GUI). EM combines a central console, agents, common services, and tools to provide an integrated, comprehensive systems management platform for managing Oracle products. From EM's console, you can:

- Administer, diagnose, and tune multiple databases
- Schedule jobs on multiple nodes at varying time intervals
- Monitor objects and events throughout the network
- Allow users to share objects with other Enterprise Manager users
- Customize your display using multiple graphic views and groups of network objects, such as nodes and databases
- Administer Real Application Clusters environments
- Integrate participating Oracle and third-party tools

Enterprise Manager Architecture (Continued)

EM extends the client-server architecture to a highly scalable three-tier model. The first tier consists of a Java-based console and integrated applications that can be installed or run from a Web browser. The second tier component is the Oracle Management Server (OMS). The main function of the OMS is to provide centralized intelligence and distributed control between clients and managed nodes, processing and administering all system management tasks. As the number of managed systems increases, the architecture scales through the addition of OMSs. Failover and load balancing are also automated within the OMS, providing much greater reliability in notification processing. The OMS uses the EM repository as its persistent back-end store. This repository maintains system data, application data, and the state of managed entities distributed throughout the environment. Multiple users can access and share repository data for systems where management responsibilities are shared. The third tier is comprised of targets, such as databases, nodes, or other managed services. The intelligent agent functions as the executor of jobs and events sent by the OMS. Client/server connections to the database through the EM console are also supported if you do not want to use a repository database.

Job and Event System Features



ORACLE

13-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Job and Event System Features

The Job system allows you to automate standard and repetitive tasks, such as executing a SQL script or executing an operating system command. With the Job system, you can create and manage jobs, share jobs with other users, schedule execution of jobs, and view information about the jobs. Jobs can be scheduled on a single node or multiple nodes in the network, provided that the node has an Intelligent Agent running on it.

Events

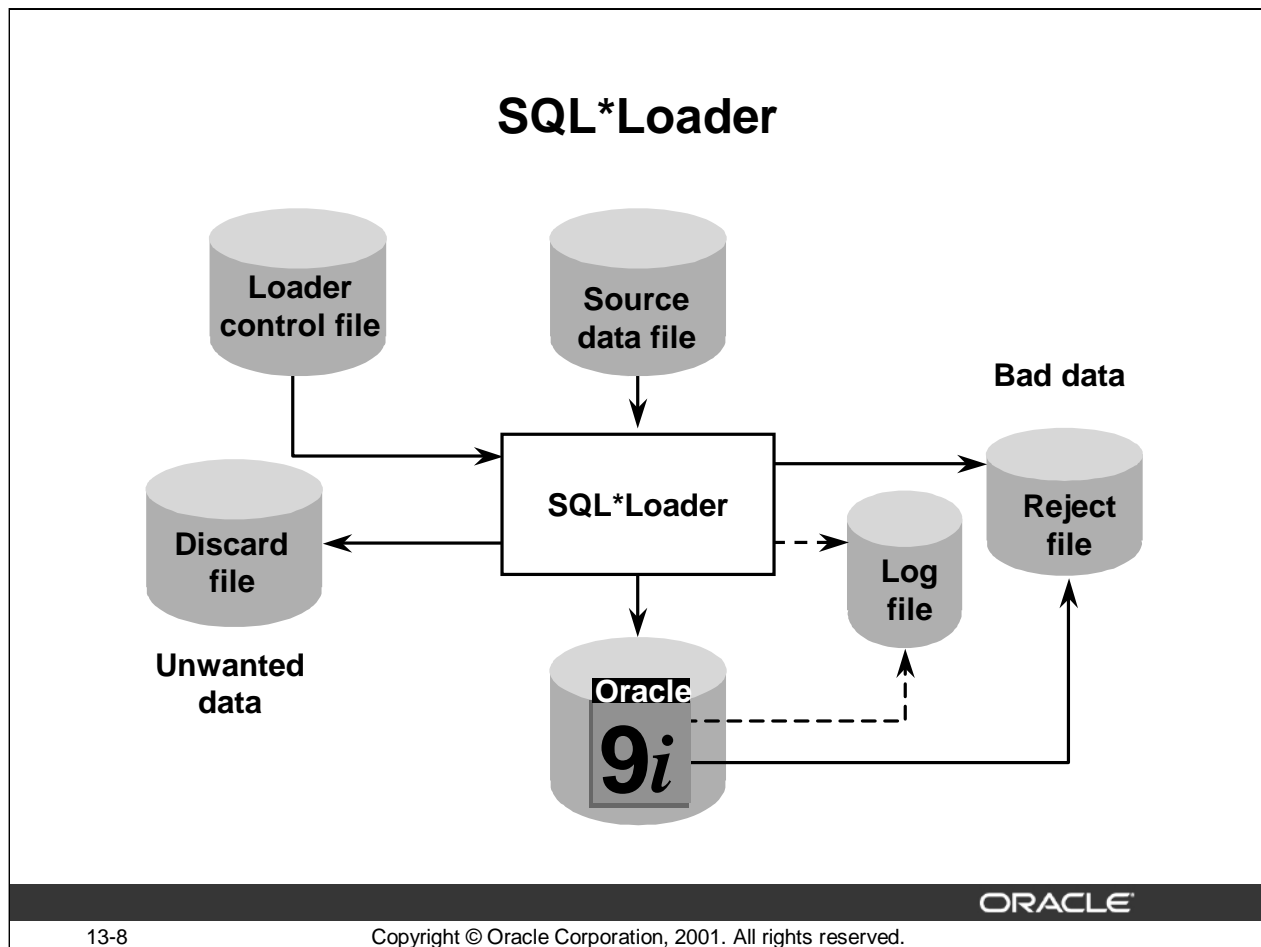
The Event system allows you to monitor specific event conditions, such as loss of service or lack of storage, that occur in your network environment. You choose events on databases, listeners, or nodes, and then select the threshold parameters for which you want to be notified. You can share the events with other administrators and notify specific administrators when an event condition occurs. For some events, you can also choose to execute a job that automatically corrects the problem. The Event system allows you to efficiently monitor a large system. Using EMS and Intelligent Agents, you can effectively monitor any number of databases 24 hours a day, and be alerted when a problem is detected. You can also pinpoint only the services you wish to monitor. The Event system can be extended to include other third-party applications that detect events independent of the Intelligent Agents. These applications can be integrated into the Event system and communicate directly with the Intelligent Agents.

Intelligent Agent

The Oracle Intelligent Agent is an intelligent process running on a remote node in the network. An agent resides on the same node as the service it supports. However, the agent can support more than one service on a particular node. For example, if two databases are installed on one machine, a single agent can support both databases. The agents are responsible for:

- Providing local services or calling operating system dependent services to interact locally with the managed targets
- Accepting jobs or events from the Console or other third-party applications
- Running jobs, collecting their results and output, and queuing them for the Oracle Management Server
- Checking for events, and queuing the resulting event reports
- Returning job and event reports to the appropriate Oracle Management Server
- Canceling jobs or events as directed by the Console or other applications
- Handling Simple Network Management Protocol (SNMP) requests, if supported

Intelligent Agents are autonomous because they function without requiring that the Console or Oracle Management Server be running. An agent that services a database can run when the database is down, allowing the agent to start up or shut down the database. The Intelligent Agents can independently perform administrative job tasks at any time, without active participation by the administrator. Similarly, the agents can autonomously detect and react to events, allowing them to monitor the system and execute a job to correct problems without the intervention of the administrator.



SQL*Loader

SQL*Loader loads data from external files into tables of an Oracle database.

- Powerful data parsing engine puts little limitation on the format of the data in the data file
- Loads data from multiple data files into multiple tables during the same load session
- Character set aware (you can specify the character set of the data)
- Selectively loads data (you can load records based on the records' values)
- Manipulates the data before loading it, using SQL functions
- Generates unique sequential key values in specified columns
- Uses the OS file system to access data files
- Loads data from disk, tape, or named pipe
- Performs sophisticated error reporting that greatly aids troubleshooting
- Supports two *paths*: while conventional path loading is very flexible, direct path loading provides superior loading performance
- Loads arbitrarily complex object-relational data
- Supports secondary data files for loading of LOBs and collections
- Compatible with the DB2 Load Utility from IBM; consequently, with a few changes, DB2 Load Utility control files can be used as a SQL*Loader control file

SQL*Loader (continued)

SQL*Loader takes as input a control file that controls the behavior of SQL*Loader, and one or more data files. Output of the SQL*Loader is an Oracle database, where the data is loaded, a log file, a bad file, and potentially a discard file.

The control file is a text file written in a language that SQL*Loader understands. The control file describes the task that the SQL*Loader is to carry out.

The other input to SQL*Loader, other than the control file, is the data. SQL*Loader reads data from one or more files specified in the control file. The data in the data file is organized as records. A particular data file can be in fixed record format, variable record format, or stream record format.

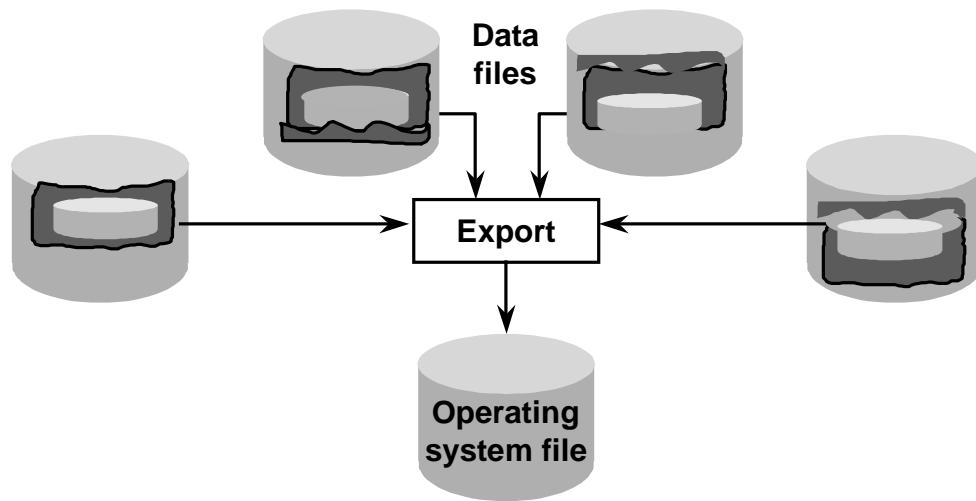
The bad file contains records rejected either by SQL*Loader or by Oracle. Records are rejected by SQL*Loader when the input format is invalid.

After a record is accepted for processing by SQL*Loader, a row is sent to Oracle for insertion. If Oracle determines that the row is valid, then the row is inserted into the database. If not, the record is rejected, and SQL*Loader puts it in the bad file.

As SQL*Loader executes, it may create a file called the discard file. It contains records that were filtered out because they did not match any record-selection criteria specified in the control file.

When SQL*Loader begins execution, it creates a log file. The log file contains a detailed summary of the load, including a description of any errors that occurred during the load.

Exporting Data



Create logical copies of database objects

ORACLE

13-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Export

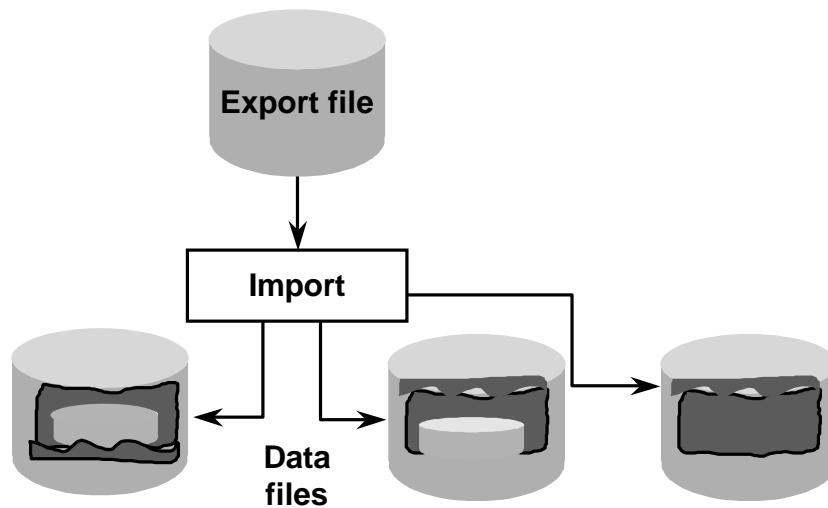
Export provides a simple way for you to transfer data objects between Oracle databases, even if they reside on platforms with different hardware and software configurations. Export extracts the object definitions and table data from an Oracle database and stores them in an Oracle binary-format Export dump file located typically on disk or tape. You can then use FTP or physically transport (in the case of tape) to a different site and use, with the Import utility, to transfer data between databases that are on machines not connected through a network or as backups in addition to normal backup procedures.

The Export and Import utilities can also facilitate certain aspects of Oracle Advanced Replication functionality like offline instantiation.

The Export utility can also do the following:

- Specify multiple dump files for an export command
- Specify a query for the select statements that Export uses to unload tables
- Export and import precalculated optimizer statistics instead of recomputing the statistics at import time

Importing Into a Database



ORACLE

13-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Import

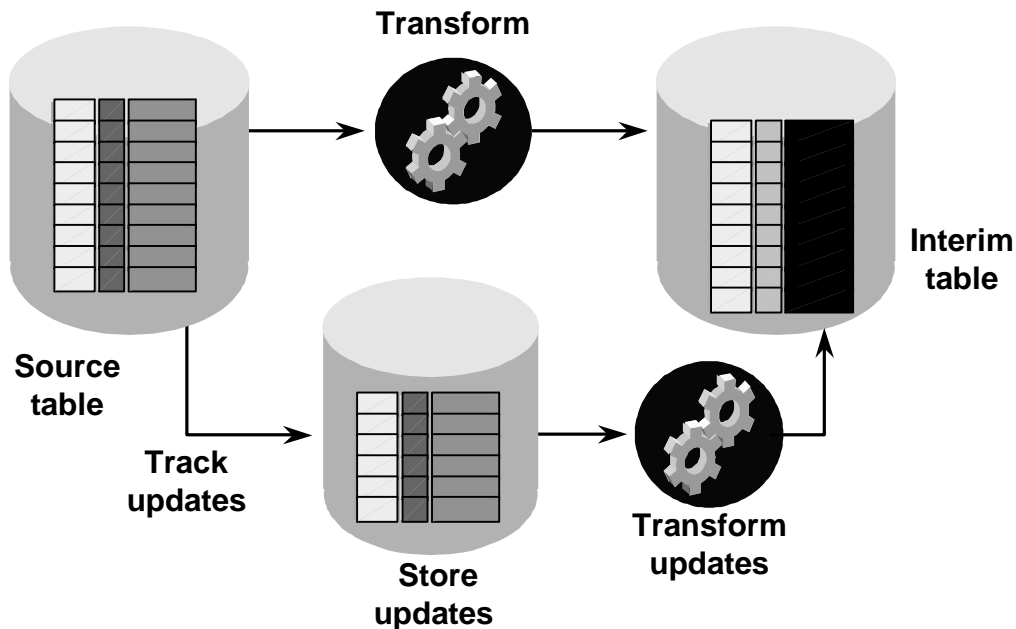
The basic concept behind Import is simple. Import inserts the data objects extracted from one Oracle database by the Export utility (and stored in an Export dump file) into another Oracle database or the same database. Export dump files can only be read by Import. It is not mandatory to import every object that has been exported; the Import utility is selective just as when you export, you are not required to import everything in the database.

Export and Import

Export and Import together provide a powerful toolset for:

- Software migrations and upgrades
- Logical backups
- Database defragmentation
- Copying cost-based optimizer statistics between databases

Online Table Redefinition



ORACLE

13-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Table Redefinition

In online transaction processing (OLTP) systems, there is an occasional need to reorganize large, frequently used tables to improve the performance of queries or DML statements performed on these tables.

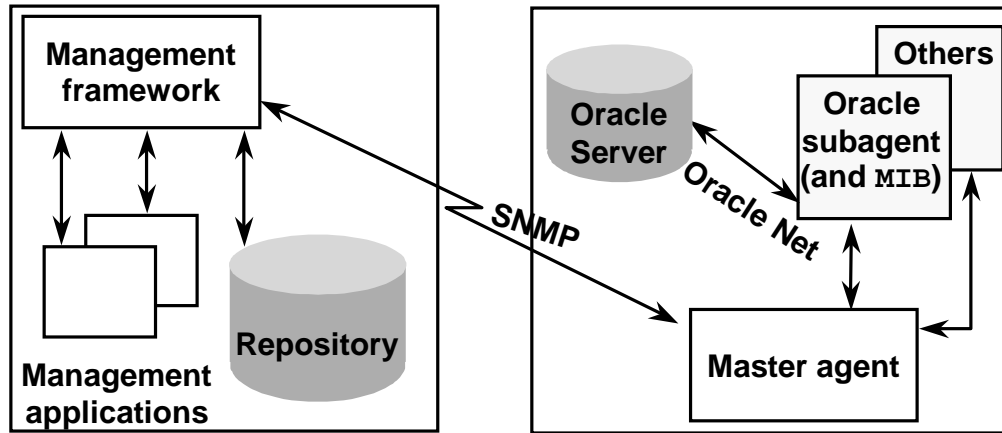
Until now, there was no simple mechanism to achieve this reorganization of the tables. Moreover, while performing the reorganization of the tables, the tables had to be locked in shared mode, which prevented any DML operations on the tables while they were being reorganized. For very large and frequently modified tables in OLTP systems, this was not acceptable. Additionally, if the reorganization was achieved by exporting the tables and importing them back, they were offline during the entire process.

The solution is to provide a mechanism to reorganize the tables while keeping them online (this is, accessible to DML) during the reorganization process. In this case, the table is locked in Exclusive mode only during a very small period of time, which is independent of the size of the table and the complexity of the reorganization.

The original table is copied and transformed into the new table. While this process takes place, online DML operations are captured and stored in a temporary table.

When the main transformation is complete, the updates stored in the temporary table are transformed and merged with the new table.

Oracle SNMP Support



ORACLE

13-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Management Station

The management station refers to a node from which managed elements are monitored using the SNMP protocol. Typically, it is a stand-alone workstation that is on the same network or internetwork as the managed elements.

Management Framework

At the management station, the management framework uses SNMP to request management information from other nodes. The framework collects, graphs, and possibly acts on that SNMP data, and saves some or all of it in a repository for historical analysis and reporting.

Management frameworks include many tools and options. In addition to directly requesting information from managed nodes, frameworks typically use daemons to alert them when a managed node has sent a trap in response to a specific set of conditions. The traps also can be used to trigger management applications.

Management Application

The management applications are the tools integrated with the management framework to accomplish more specialized network or database tasks. These applications contain virtually all of the sophisticated logic associated with network management.

Managed Node

The managed node is a platform, such as a UNIX server, on which elements to be monitored reside.

SNMP Support and Oracle Products

Oracle SNMP support is provided for the following products:

- **Oracle7 Server, release 7.2, and higher**
- **Two networking services:**
 - Network Listener
 - Oracle Names
- **Enterprise Manager**

ORACLE

13-14

Copyright © Oracle Corporation, 2001. All rights reserved.

Master Agent

The master agent is the process on a managed node that accepts queries, also called *polls*, from the management framework and communicates with the elements to be managed in order to answer the query. It also can send SNMP traps independently in response to specific conditions. Only one master agent can exist on each managed node.

Subagents

The subagent is a process that receives queries for a particular managed element from the master agent and sends back the appropriate answers to the master agent. One subagent exists for each managed element residing on the managed node.

Management Information Base

A management information base (MIB) is a text file, which describes the variables containing the information that SNMP can access. The variables described in a MIB, which are also called MIB objects, are the items that can be monitored using SNMP. There is one MIB for each element being monitored. The actual values of the variables are not part of the MIB, but are retrieved through a platform-dependent process called *instrumentation*. The concept of the MIB is very important because all SNMP communications refer to one or more MIB objects. What is transmitted to the framework is, essentially, MIB variables and their current values.

Unit 2 Summary

- **Object (relational) theory:**
 - Type definitions, object references
 - methods, inheritance, polymorphism, encapsulation
- **Globalization support**

ORACLE

Unit 2 Summary

Business Intelligence

- Partitioning, parallelization
- Materialized views, summary management
- Transportable tablespaces
- Direct path loads
- OLAP Services

ORACLE

Unit 2 Summary

- **Backup and recovery**
 - Offline and online backups
 - Physical and logical backups
 - Recovery manager (RMAN)
 - Data Guard
- **Oracle9i administration tools**
 - *iSQL*Plus*
 - EM
 - Export, Import, SQL*Loader
 - Online table redefinition

ORACLE

Oracle9i Network Services



ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

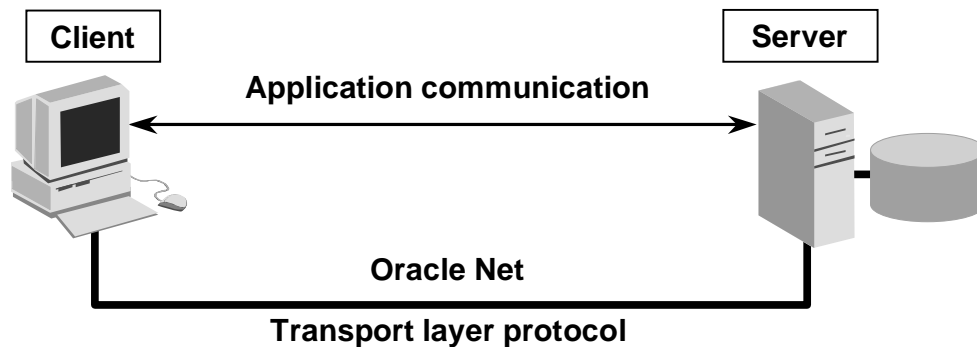
Objectives

After completing this lesson, you should be able to discuss the following Oracle9i networking topics:

- **Client-server and server-server architecture**
- **Shared Server and connection load balancing**
- **Connection pooling and multiplexing**
- **Oracle Connection Manager**
- **Directory Repository**
- **Heterogeneous Services**

ORACLE[®]

Client-Server Architecture



ORACLE

A-3

Copyright © Oracle Corporation, 2001. All rights reserved.

Client-Server Architecture

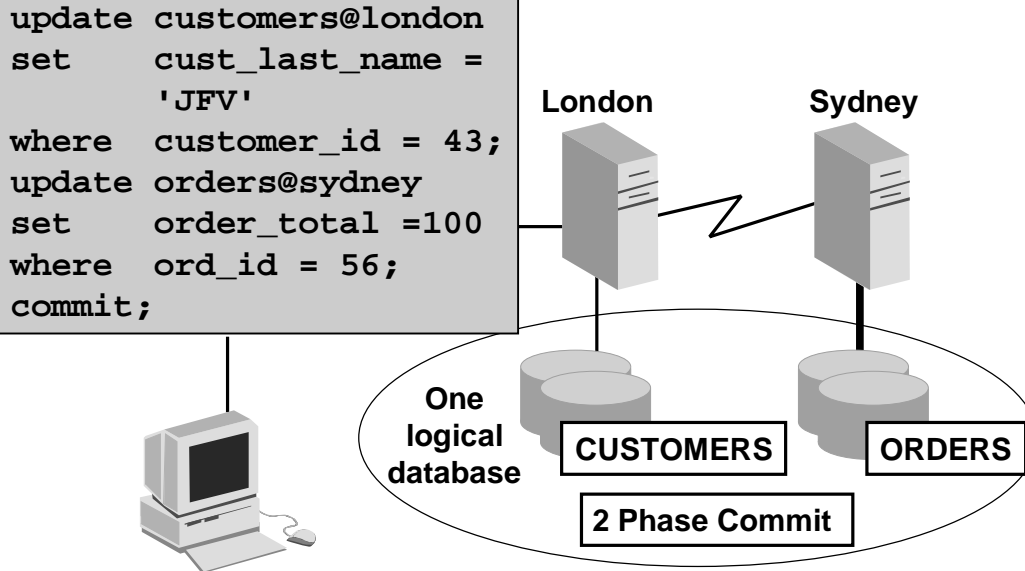
Oracle Net is the foundation of Oracle's family of networking products that allow services and their applications to reside on different computers and communicate as peer applications. The main function of Oracle Net is to establish network sessions and transfer data between a client machine and a server or between two servers. Oracle Net is located on each machine in the network. After a network session has been established, Oracle Net acts as a data courier for the client and the server.

Network sessions are established with the help of a listener. The listener is a separate process that resides on the server whose responsibility is to listen for incoming client connection requests and manage the traffic to the server.

Oracle Net uses the Transparent Network Substrate (TNS) and industry-standard networking protocols to connect a client to a server and establish an Oracle network session.

Oracle protocols are Oracle's implementation of the transport layer. Oracle protocols include: LU6.2, Named Pipes, VI, TCP/IP, TCP/IP with SSL.

Server-Server Architecture: Distributed Databases



A-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Distributed Processing

Oracle databases and client applications operate in a distributed processing environment. Distributed or cooperative processing involves interaction between two or more computers to complete a single data transaction. Applications such as an Oracle tool act as clients requesting data to accomplish a specific operation. Database servers store and provide the data.

In a typical network configuration, clients and servers can exist as separate logical entities on separate physical machines.

Distributed Databases

With this type of client-server architecture, you can also distribute databases across a network. A distributed database is a network of databases stored on multiple computers that appears to the user as a single logical database. Distributed database servers are connected by a database link, or path from one database to another. One server uses a database link to query and modify information on a second server as needed, thereby acting as a client to the second server.

Naming

Oracle Net provides these naming methods:

- **Local naming**
- **Directory naming**
- **Oracle names**
- **Host naming**
- **External naming**

```
sales=(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=stc)(PORT=1521))
  (CONNECT_DATA=(SERVICE_NAME=dbtest)))
```

```
CONNECT oe/oe@sales
```

ORACLE

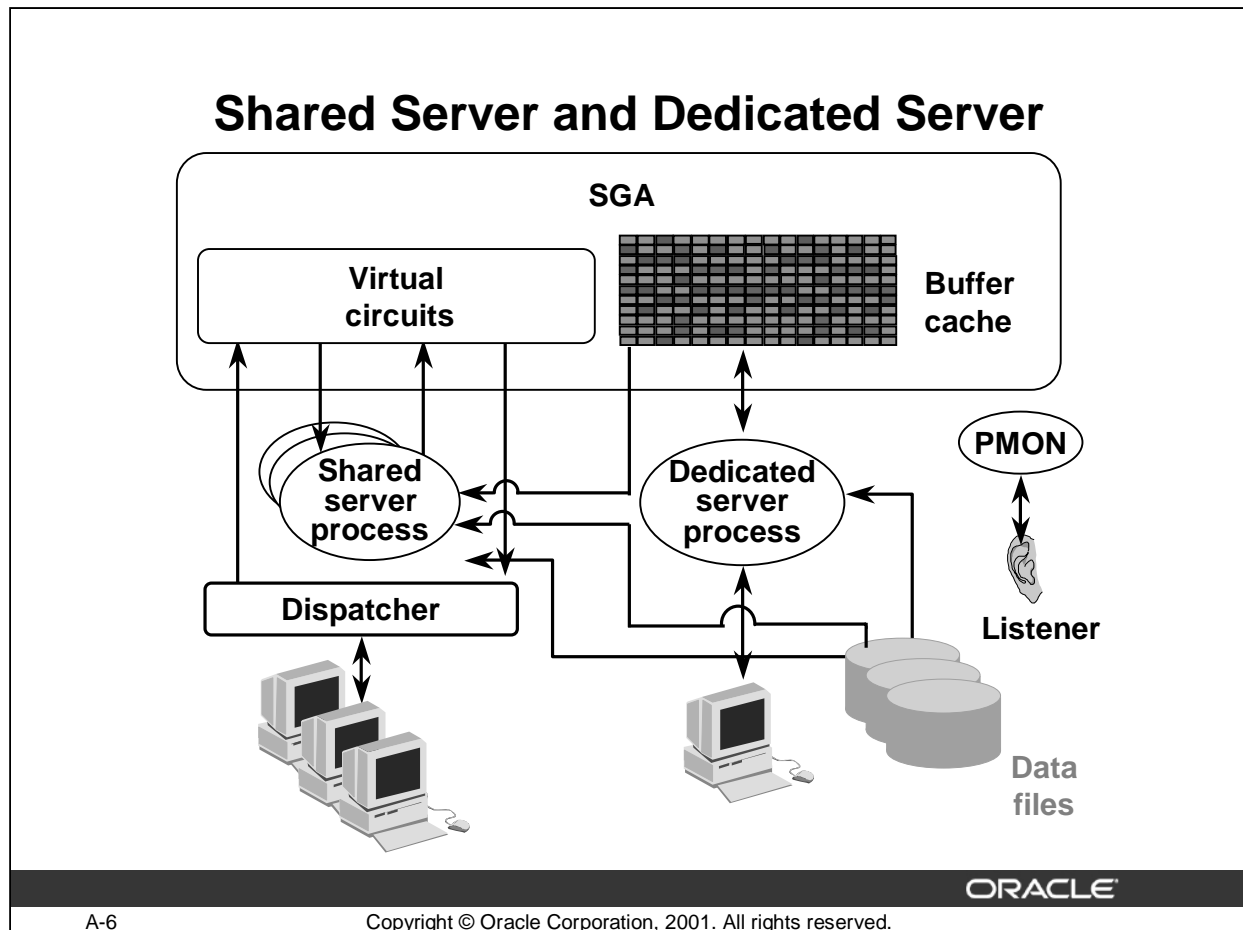
A-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Naming

Naming is generally used to identify an entity with a simple name. Oracle Net uses this simple name, called a connect identifier, to identify a connect descriptor. A connect identifier is specified in several different ways. One way is through use of a net service name (*sales* in the above slide) that maps to a connect descriptor. Users initiate a connect request by providing a connect string. A connect string includes a user name and password, along with a connect identifier, or the complete connect descriptor for the service to which they want to connect. When a net service name is used, connection processing takes place by first mapping the net service name to the connect descriptor. This mapped information is stored in one or more repositories of information that are accessed with naming methods:

- Local naming stores net service names and connect descriptors in localized files.
- Directory naming stores net service names and database service names in a centralized LDAP-compliant directory server to access a database service.
- Oracle Names uses Oracle proprietary software to store the names and addresses of all database services on a network. Oracle Names will no longer be supported in the future.
- Host naming enables users to connect to an Oracle database server by using a host name alias using existing names resolution service, such as Domain Name System (DNS), or Network Information Service (NIS).
- External naming stores net service names and descriptors in a supported non-Oracle naming service like NIS and DCE.



Shared Server and Dedicated Server

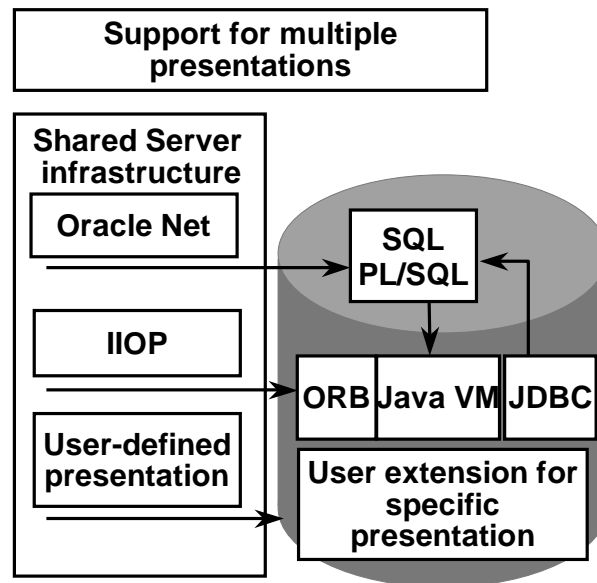
Based on the service handler type registered with the listener, the listener forwards requests to either a shared server process or a dedicated server process.

Shared server processes are utilized in the shared server architecture. With shared server architectures, client processes ultimately connect to a dispatcher. The PMON process registers the location and load of the dispatchers with the listener, enabling the listener to forward requests to the least loaded dispatcher. A dispatcher can support multiple client connections concurrently. Each client connection is bound to a virtual circuit. A virtual circuit is a piece of shared memory used by the dispatcher for client database connection requests and replies. The dispatcher places a virtual circuit on a common queue when a request arrives. An idle shared server picks up the virtual circuit from the common queue, services the request, and relinquishes the virtual circuit before attempting to retrieve another virtual circuit from the common queue. This approach enables a small pool of server processes to serve a large number of clients.

With a dedicated server architecture, each client process connects to a dedicated server process. The server process is not shared by any other client. PMON registers information about dedicated server processes with the listener. The listener then starts up a dedicated server process when a client request arrives, and forwards the request to it.

Shared Server and Internet-Based Protocols

- Clients can connect through internet protocols
- Built-in support for component models
- Dynamic presentations
- Extended networking functionality



ORACLE

A-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Shared Server and Internet-Based Protocols

With Shared Server, Oracle9i understands internet-based protocols.

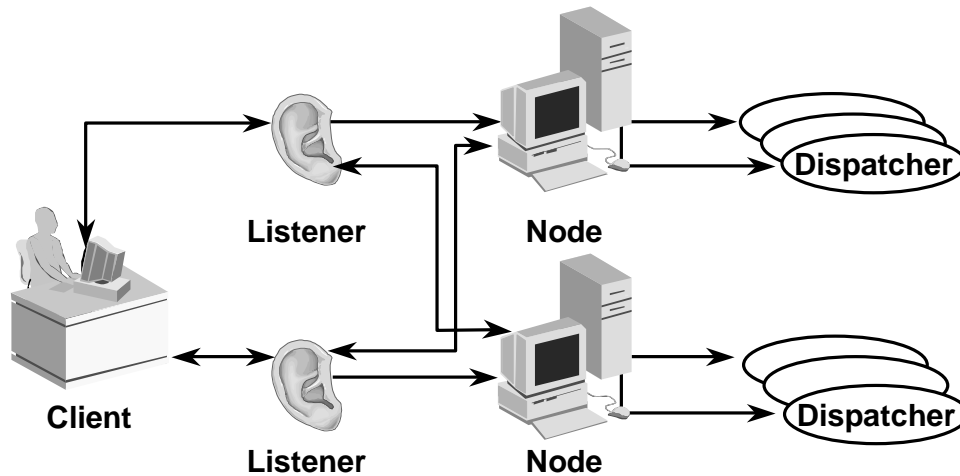
Oracle Net supports a new presentation layer called General Inter-Orb Protocol (GIOP) that is used for clients connecting to the Java option. Internet Inter-Orb Protocol (IIOP) is an implementation of GIOP over TCP/IP or TCP/IP with SSL. Oracle provides the GIOP service implementation.

Oracle Net also supports a new presentation called HTTP presentation to directly access the database as a Web Server. Built-in infrastructure supports component models such as Enterprise JavaBeans.

Oracle9i users can create new presentations dynamically and extend the networking functionality provided by the Oracle server.

Shared Server provides the ability to configure a dispatcher for each presentation.

Connection Load Balancing



1. Client randomly chooses from available listeners
2. Node with least CPU usage identified
3. Dispatcher/Listener with least number of connections is used

ORACLE

A-8

Copyright © Oracle Corporation, 2001. All rights reserved.

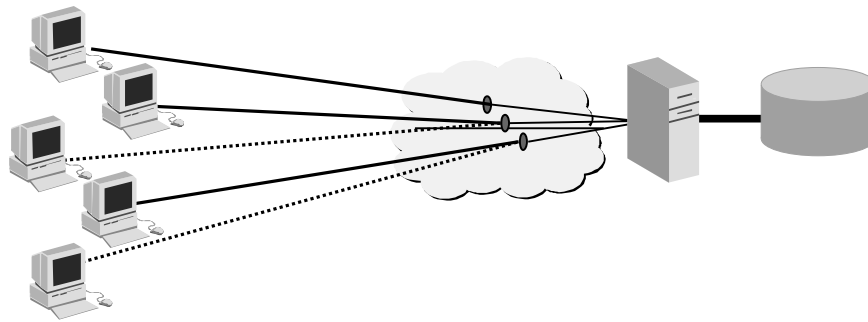
Connection Load Balancing

The goal of connection load balancing is to divide the load evenly across the available resources in an Real Application Clusters environment.

- A client requesting a connection to a service randomly chooses a listener from the list of listeners identified for the service. If the chosen listener is not available, the connection request can failover to the next available listener. This capability is called client load balancing and failover.
- Each listener has information about the load on all available nodes, because each instance has registered with all identified listeners, and is providing continuous load information (CPU usage). For each connection request, the listener identifies which node has the least amount of usage.
- The listener then also identifies which dispatcher (Shared Server configuration) is currently handling the least number of connections on that node. The connection is then routed to that dispatcher on the chosen node. In case of Dedicated Server configuration, the connection is rerouted to the least loaded listener that will ultimately start a dedicated server process

Scalability: Connection Pooling

- Idle network sessions release transport connection for use by other network sessions.
- It is ideal for interactive *high think/search time* applications.



ORACLE

A-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Connection Pooling

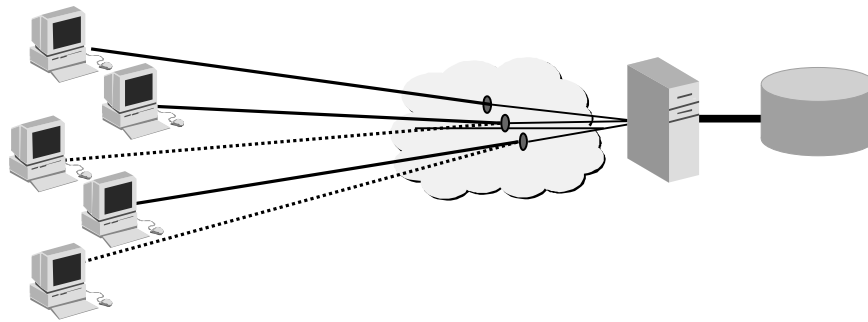
Connection pooling is a resource utilization feature that enables you to reduce the number of physical network connections to a dispatcher. This is achieved by sharing or pooling a set of connections among the client processes.

Connection pooling allows inactive physical connections to the machine to be temporarily disconnected while a logical session remains. This releases OS resources so that additional users can connect to the same machine. When network activity is resumed on the dropped connection, the logical session reestablishes a physical connection through the same process the new connection was established.

Connection pooling frees system resources and allows more users to access the database. It is ideal for interactive *high think/search time* applications, such as e-mail and data warehouse query tools.

Connection Pooling Benefits

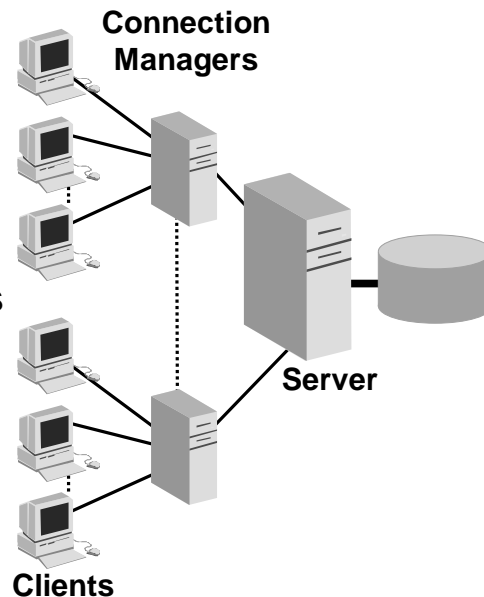
- More client/server connections are possible across a fixed number of physical server ports.
- Optimized resource utilization



ORACLE

Scalability: Oracle Connection Manager

- **Leverage multiplexing**
 - Predictable, consistent performance
 - User identity passed to the server
- **Simultaneous data access for many users**
- **Access control and cross protocol support**



ORACLE

A-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Scalability: Oracle Connection Manager

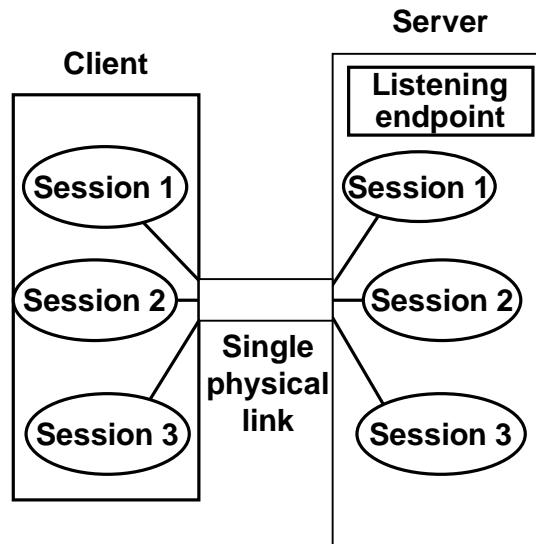
Benefits of multiplexing are magnified significantly when used in conjunction with a mid-tier concentrator application. System resources are constrained at the server side by the number of sockets that can be opened. If the number of network connections is kept below this threshold point, and sessions are *funneled* through a concentrator, then large numbers of clients can connect to a single database.

Oracle Net introduces a new application called Oracle Connection Manager that performs this functionality. Oracle Connection Manager uses multiplexing to feed numerous client network sessions across a single network connection to the server. When multiple Connection Managers are employed, thereby connecting their multiple clients, the total number of clients connecting to a database can grow exponentially.

Oracle Connection Manager also provides the opportunity to introduce additional network control and connectivity functionality. Oracle Connection Manager enables Oracle connectivity across disparate protocols without the necessity to install multiple protocol stacks on all clients; for example, TCP/IP clients can communicate with VI based servers. This is the same functionality formerly provided by the MultiProtocol Interchange in Oracle7 environments. This application proxy provides network access control so that an administrator can control which clients can access Oracle data sources based on where they are coming from, where they are going, and the service they are connecting to.

Scalability: Multiplexing

- Multiple network sessions use a single physical link
- Clients running multiple applications
- Minimized lock outs



ORACLE

A-12

Copyright © Oracle Corporation, 2001. All rights reserved.

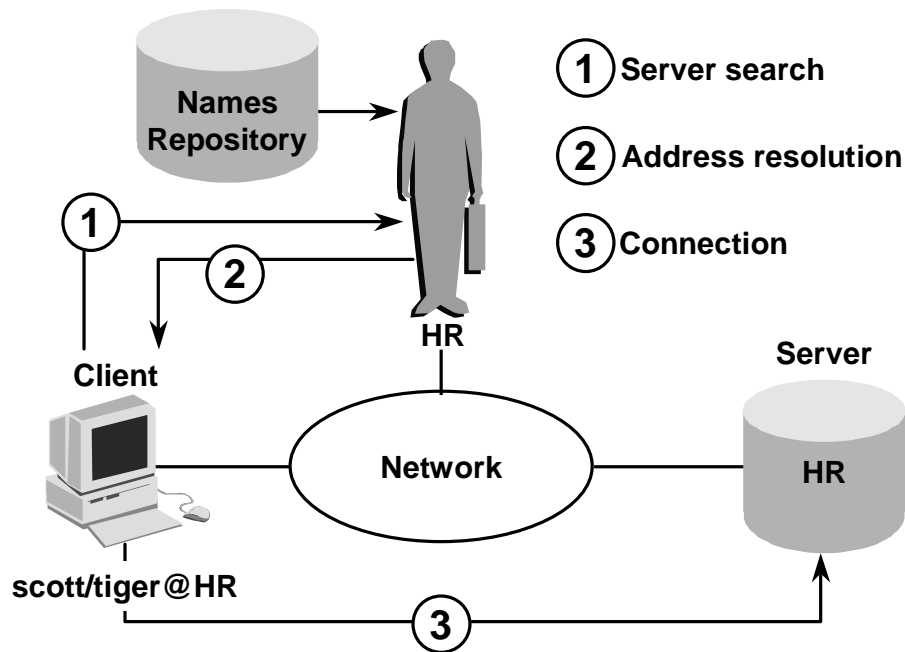
Scalability: Multiplexing

Similar to connection pooling, multiplexing also seeks to free system resources by minimizing those required to establish connectivity. Multiplexing allows multiple network sessions to share a single physical transport connection by interspersing the network traffic across the network connection.

An example of multiplexing might be a client running three applications that connect to the same machine, requiring three network sessions. Rather than requiring three separate network connections which reduces the number of sockets available on the machine that the client connects to, multiplexing takes up one socket because the three sessions share the same network connection. This reduces the overall resources used.

Note: Multiplexing is only available with Shared Server and Connection Manager.

Oracle Names Server



ORACLE

A-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Names Server

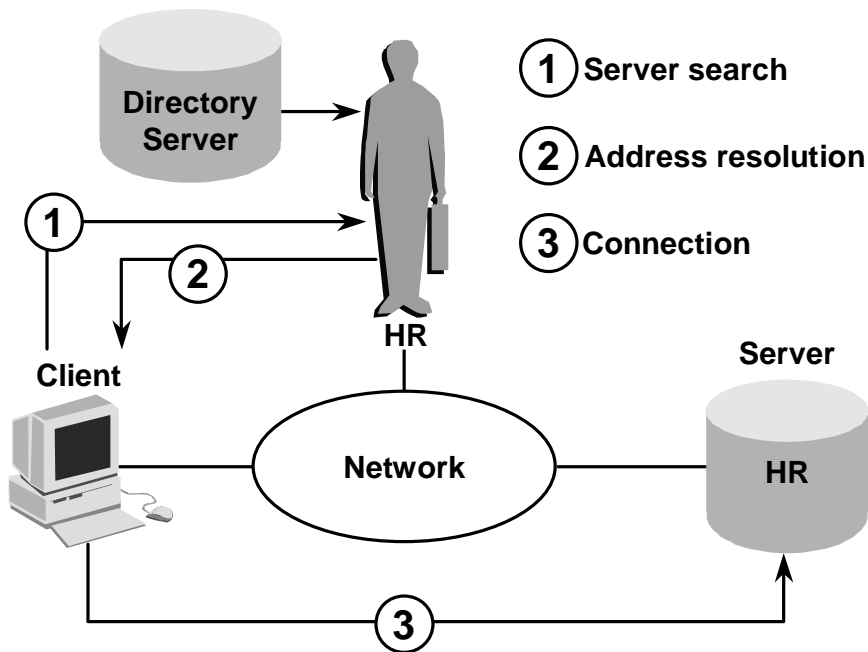
Oracle Names is a distributed naming service developed for Oracle environments to help simplify the setup and administration of global, client/server computing networks.

Oracle Names establishes and maintains an integrated system of Oracle Names servers that work together like a directory service, storing addresses for all services on a network and making them available to clients who wish to make a connection.

Similar to a caller who uses directory assistance to locate a telephone number, clients configured to use Oracle Names will refer their connection requests to a Oracle Names server. The Oracle Names server attempts to resolve the service name provided by the client to a network address. If the Oracle Names server finds the network address, it returns that information to the client. The client can then use that address to connect to the service.

Oracle Names provides an alternative to file-based or local name resolution methods, in which net service names and addresses must be configured and maintained with each individual client. By maintaining this information in a central administrative location, Oracle Names reduces the work effort associated with adding or relocating services.

LDAP-Compliant Directory Servers



ORACLE

A-14

Copyright © Oracle Corporation, 2001. All rights reserved.

LDAP-Compliant Directory Servers

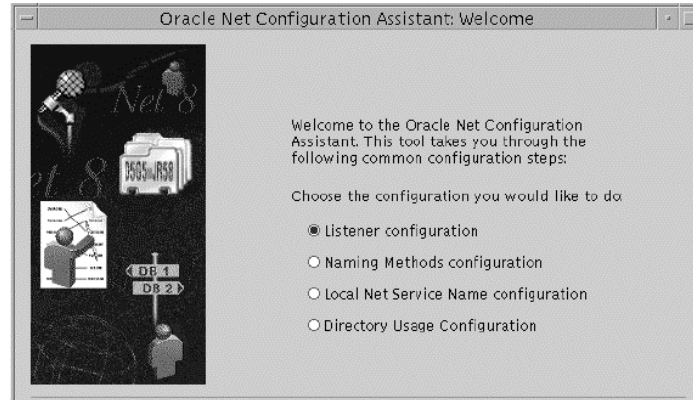
Today, network information is stored in multiple systems and in multiple directory formats. With new requirements for Internet computing and new e-business technologies, a common repository infrastructure is needed as a foundation for management and configuration of all data and resources. This kind of infrastructure reduces the cost of managing and configuring resources in heterogeneous networks.

Support of LDAP-compliant directory servers provides a centralized vehicle for managing and configuring a distributed Oracle network. The directory server can act as a central repository for all data on database network components, user and corporate policies, and user authentication and security, thus replacing clientside and serverside localized network files.

Oracle Net Services use a centralized directory server as one of the primary methods for storage of connect identifiers. Clients configured to use the directory server, use the connect identifiers in their connect string. The directory server resolves the connect identifier to a connect descriptor that is passed back to the client.

Oracle Net Services support Oracle Internet Directory and Microsoft Active Directory.

Oracle Net Configuration Assistant



ORACLE

A-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Net Configuration Assistant

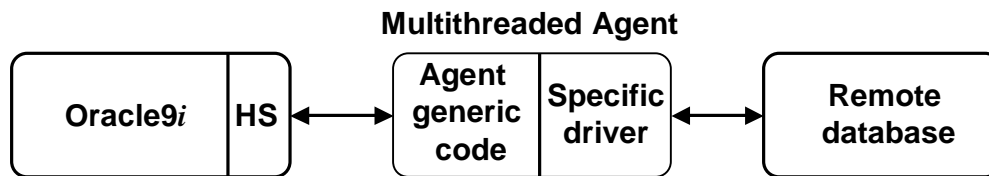
Oracle Net Configuration Assistant is provided primarily to configure basic network components during installation, including:

- Listener names and protocol addresses
- Naming methods the client will use to resolve connect identifiers
- Net service names in a network file
- Directory server access

Oracle Net Configuration Assistant runs automatically during software installation, as described in your Oracle installation guide. It can also be run after installation in stand-alone mode to configure naming method usage, the listener, net service names in a network file, and directory server usage in a way similar to that which is provided during installation.

Oracle Net Manager is a graphical user interface tool that combines configuration abilities with Oracle Names component control to provide an integrated environment for configuring and managing Oracle Net. It can be used on either the client or server. Oracle Net Manager is also integrated with Oracle Enterprise Manager. It is more sophisticated than the Configuration Assistant and allows you to fine tune most of the parameters.

Heterogeneous Services



- **Oracle Transparent Gateway**
 - Sybase
 - Teradata
 - SQLServer
- **Generic Connectivity**
 - ODBC
 - OLE DB (SQL or FS)

ORACLE

A-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Heterogenous Services

Heterogeneous access is a challenge that affects many organizations. Many run several different database systems. Each of these systems stores data and has a set of applications that runs against it. Consolidation of this data into one database system is often difficult. Heterogeneous Services, an integrated module within the Oracle9i database server, has been designed to access data in non-Oracle systems by means of either Oracle Transparent Gateways or generic connectivity. Most of the heterogeneous connectivity related processing is done in this module.

- An Oracle Transparent gateway is a gateway that is designed and optimized for accessing a specific non-Oracle system. Oracle Corporation provides gateways to access several commercially produced non-Oracle systems like DB2, Ingres, IBM DRDA, Rdb, Informix.
- In addition to transparent gateways to various non-Oracle database systems, there is a set of agents that comprise the Oracle generic connectivity feature. These agents contain only generic code and the customer is responsible for providing the necessary drivers. Oracle has generic connectivity agents for ODBC and OLE DB that enable you to use ODBC and OLE DB drivers to access non-Oracle systems that have an ODBC or an OLE DB interface. Generic Connectivity is mainly used for low-end data integration situations requiring ad hoc querying capabilities like with Foxpro, Access, or dBase.

Agents are used for security reasons avoiding Oracle Server crash if problems occur on the non-Oracle system. Also, an agent can reside anywhere on the network and can be truly multithreaded.

Summary

In this lesson, you should have learned about:

- **Client-server and server-server architecture**
- **Shared Server and connection load balancing**
- **Connection pooling and multiplexing**
- **Oracle Connection Manager**
- **Directory Repository**
- **Heterogeneous Services**

ORACLE[®]

B

Oracle9i Replication

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- **List the different ways data can be replicated between Oracle databases**
- **Describe the difference between asynchronous and synchronous replication**
- **List the different types of materialized views**
- **Describe the tools used to manage Advanced Replication**

ORACLE

What Is Replication?

- **Multiple copies of data at different sites**
- **Increased availability**
- **Manual data replication implementations:**
 - **Export/Import**
 - **CREATE TABLE AS SELECT**
 - **COPY**

ORACLE

B-3

Copyright © Oracle Corporation, 2001. All rights reserved.

What is Replication?

The Data Replication feature provides:

- The ability to maintain multiple copies of data at different sites in a distributed environment
- The ability of the systems to function autonomously even when other systems in the distributed environment fail

The Oracle server can implement a very basic form of data replication, using asynchronous data propagation, with the following:

- Export and Import
 - Whole table only
 - Must manually perform export and import (with TRUNCATE command) repeatedly for replica to see changes
- CREATE TABLE AS SELECT
 - Table cannot exist in order for command to succeed
 - One-time data copy procedure
- COPY command in SQL*Plus
 - Used for copying data
 - Provides a way to copy LONG columns

Unidirectional Replication

The following data structures can be configured or used to automatically propagate changes one-way to a remote site:

- **Triggers**
- **Read-only materialized views**

ORACLE

B-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Unidirectional Replication

You can configure the following forms of data replication to replicate data changes automatically:

- Database triggers
 - Real-time, synchronous data propagation
 - Must create trigger for each DML activity: UPDATE, INSERT, DELETE
 - Database and network must be available, or commit at local site will fail
- Read-only materialized view
 - Local copy of entire remote table, or partial copy of rows and columns for remote table
 - Can use a job to automatically refresh materialized view with new changes created at master site

Bi-Directional Replication

Advanced Replication within the Oracle database has the following features:

- **Table-level replication**
 - Entire contents of tables are replicated to each site.
 - With materialized views, subsets of table data can also be replicated.
- **Data can be modified at each site, which can cause data conflicts.**
- **Synchronous or asynchronous data propagation**
 - Asynchronous propagation can be performed manually or automatically.

ORACLE

B-5

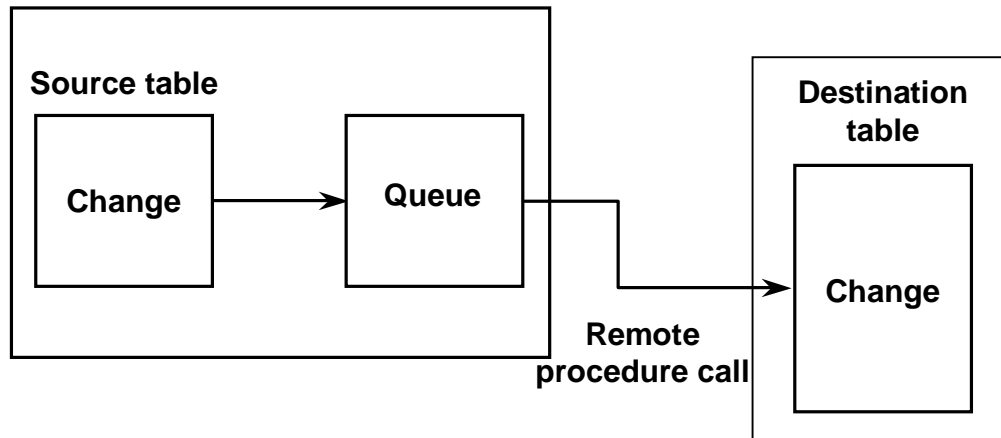
Copyright © Oracle Corporation, 2001. All rights reserved.

Bi-directional Replication

In Advanced Replication, the changes applied to a table at one site are captured and stored locally before being forwarded and applied to the replicated tables at each of the remote locations in the replicated database. This is known as asynchronous replication, sometimes also referred to as *store and forward* replication. Synchronous replication, applying data changes to all replication sites real-time, is also available.

Replicated databases are different from distributed databases. In a replicated database, the tables being replicated exist at each site. In a distributed database, the data for a particular table is available at each system, but the table itself resides in only one database.

Functionality Overview Synchronous Propagation



ORACLE

B-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Features of Synchronous Propagation

Synchronous data replication occurs when an application updates a local replica of a table, and within the same transaction, also updates all other replicas of the same table. Consequently, synchronous replication is also called real-time data replication.

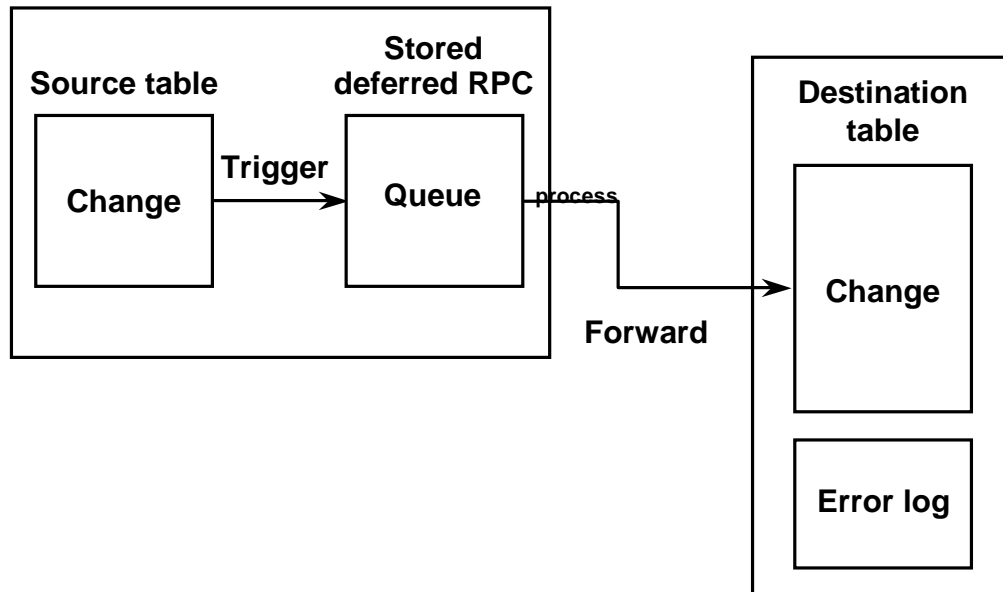
Advantages

- With synchronous data propagation, you can propagate and apply changes between master tables, with the result that all sites in a replicated environment are identical at all times.
- Changes at other sites are immediately reflected at your local site.
- There are no conflicting updates from master to master if all sites are propagating synchronously.

Disadvantages

- If you experience network failures on any site to which you are propagating changes, you will not be able to perform local updates until the failure is resolved.
- If one master site cannot process a transaction for any reason, the transaction is rolled back at all master sites.
- Requires a stable environment to function smoothly:
 - Underlying network
 - All master sites well-tuned to avoid rollbacks

Functionality Overview Asynchronous Propagation



ORACLE

B-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Features of Asynchronous Propagation

Asynchronous data replication occurs when an application updates a local replica of a table, stores replication information in a local queue, and then forwards the replication information to other replication sites at a later time. Consequently, asynchronous replication is also called store-and-forward data replication.

Advantages

- Failure of any site containing a replicated table does not block the propagation of changes between other sites.
- Deferred transactions are propagated at intervals specified by the DBA.

Disadvantages

- Changes at other sites are not immediately reflected at your local site, resulting in temporary inconsistencies among replicas. However, the data propagation interval for asynchronous replication can be set to a very small interval in order to simulate real-time replication.
- Conflicting changes can be made at multiple sites. These conflicts are detected automatically at the receiving site and conflict resolution procedures can be enabled to automatically handle these conflicts.

Propagation of Deferred RPCs

- **Initiated by standard PL/SQL stored procedure calls**
- **Replication at time-based intervals using built-in job queue**
- **Propagation to each database can be controlled independently.**
- **Transactional consistency**
- **Transaction ordering**

ORACLE

B-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Features of Propagation of Deferred RPCs

- The propagation mechanism for deferred RPCs provides applications with the flexibility of control over the timing of propagation.
- You can use a standard PL/SQL stored procedure call to initiate the propagation of queued deferred RPC calls.
- The mechanism is designed for efficient time-based propagation at frequent intervals; for example, every few seconds. You can perform replication at time-based intervals using a built-in job queue facility.
- You can perform replication on demand.
- The propagation of the deferred RPCs can be controlled from various locations and with different priorities.
- Propagation to each target database can be controlled independently.
- The target destination link is specified when a deferred RPC is submitted to the queue, or when a deferred RPC is registered with the replication administration facility (discussed later).
- Each qualified destination database can have different propagation intervals.

Error Handling of Deferred RPCs

- **Propagation failure due to the unavailability of the target database results in deferred RPCs remaining in the queue.**
- **Other errors such as duplicate key violations result in rollback and an error message in the error log table at the destination.**
- **Procedure calls in the error log at the destination can be executed later.**

ORACLE

B-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Failures

If propagation fails because the target database is not available or accessible through the network, or the tablespace is full, the deferred RPCs for that target database remain in queue at the source site for later execution.

Other errors, such as duplicate key violation, that are encountered during execution on the target system, will cause:

- Any transaction to be rolled back
- The error message and the information of the deferred RPCs to the transaction to be stored in the error log table on the target database

A PL/SQL-stored procedure enables DBAs to easily re-execute procedure calls in the error log after fixing the problem that caused the error (discussed later in this course).

Recovery of Deferred RPCs

- **System or network failure requires only normal procedures for database recovery.**
- **A deferred RPC queue comprises normal database objects, which are recoverable by means of a redo log.**
- **Propagation of deferred RPC is protected as a two-phase commit transaction.**

ORACLE

B-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Recovery

The failure of source or target systems or a network requires only normal database and network procedures to recover the database to the current point in time.

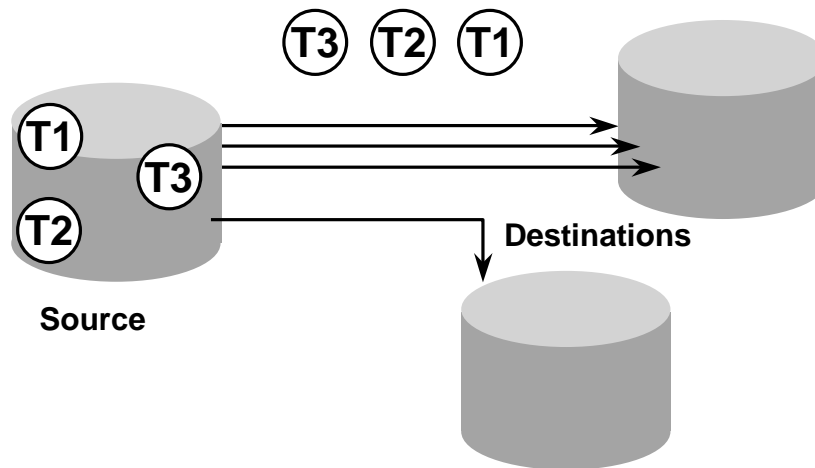
The deferred RPC queue and supporting facilities are implemented using database tables fully protected by the Oracle Server redo log.

Actual propagation and execution of a deferred RPC is performed as a distributed transaction protected by the distributed protocol.

In a global distributed database, good distributed database designs with fault-tolerant mechanisms are important.

Performance Optimization

- Streams-based execution
- Parallel propagation



ORACLE

B-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Streams-based Execution

Multiple deferred RPCs are executed as a transaction with one message exchange.

Parallel Execution

A deferred RPC transaction can be executed on multiple remote systems in parallel.

Parallel Propagation

This method asynchronously propagates replicated transactions, using multiple, parallel transit streams for higher throughput.

Performance Issue

Execution of transactions on remote nodes can potentially result in performance bottlenecks. Although multiple transactions are executed in parallel on one site, they may be propagated serially to another site. If a T1 transaction requires a disk access on the destination site, it could block subsequent transactions until T1 has completed its I/O. This may be truer in serial propagation than in parallel propagation.

Types of Replication Environments

Oracle supports the following types of replication environments:

- **Multimaster replication**
- **Single-master replication**
 - Read-only materialized views
 - Updatable materialized views
 - Oracle Lite materialized views
- **Multimaster and materialized view hybrid configurations**

ORACLE

B-12

Copyright © Oracle Corporation, 2001. All rights reserved.

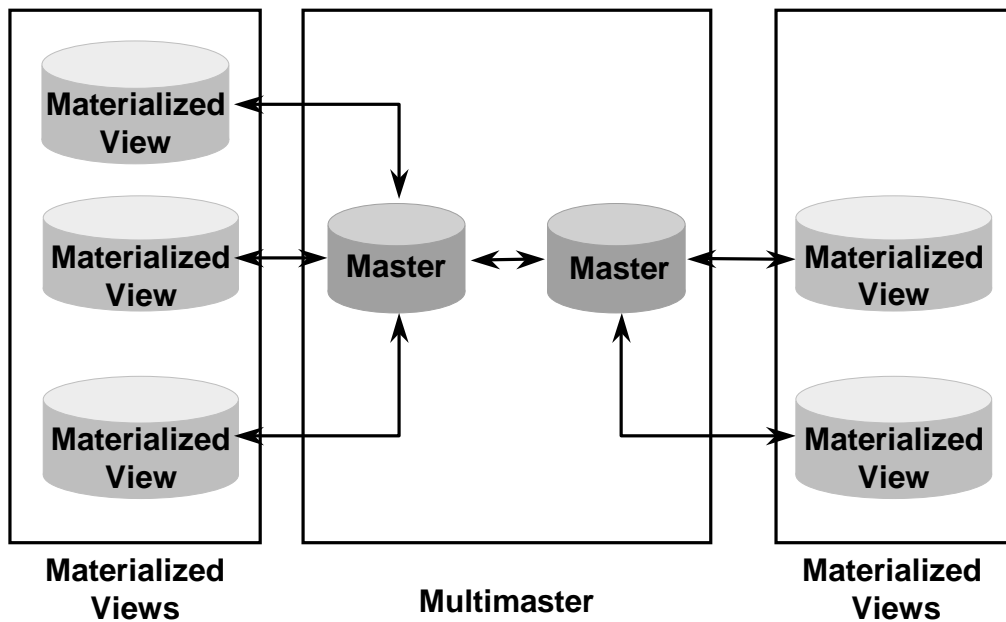
Types of Replication Environments

Oracle Replication supports both full-table replication and replication of defined subsets of tables.

There are several ways to implement replication:

- Multiple-master replication (peer-to-peer or n-way replication)
- Updatable materialized views
- Hybrid configurations

Hybrid Configuration



ORACLE

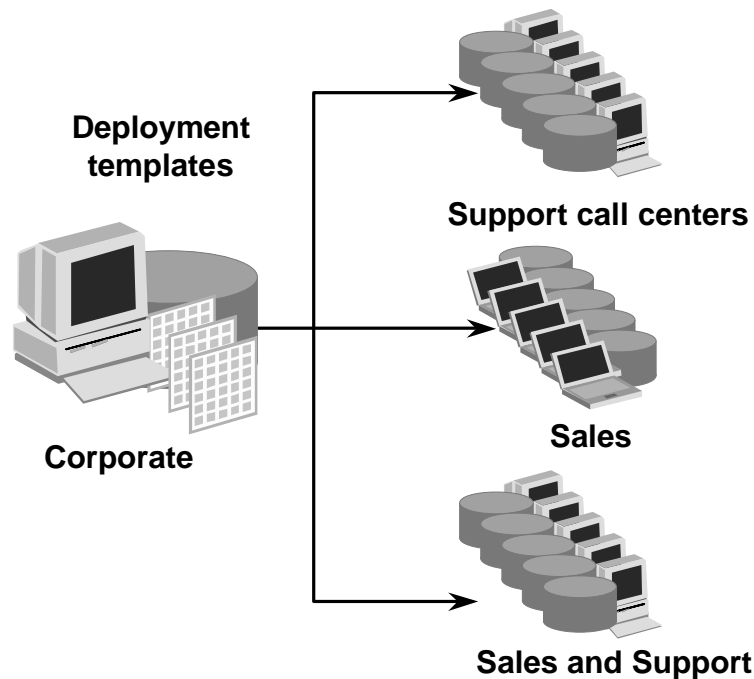
B-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Hybrid Configuration

- By combining the multimaster configuration with updatable materialized views, you can create sophisticated configurations (Enterprise Edition only).
- This configuration supports both full-table and table subset replication in a single replication system.
- Materialized views can be remastered if a master site fails, resulting in greater fault tolerance.

Oracle Deployment Templates



ORACLE

B-14

Copyright © Oracle Corporation, 2001. All rights reserved.

Deployment Templates

Oracle offers deployment templates to allow the DBA to package a materialized view environment for easy, custom, and secure deployment. Packaging a deployment template is the process of defining the materialized view environment that will be created by the deployment template. Packaging a deployment template prepares it for instantiation at the remote materialized view site. Instantiation creates the materialized view site objects and populates the materialized views with data.

In the illustration above, the corporate database might contain the following tables:

- Support
- Customer
- Order
- Product

The support call centers would need access only to the support, customer, and product tables. The sales force would need access to the order, product, and customer tables. A single deployment template could be set up using a parameter to determine which materialized views are created at the materialized view site. Every time new salespeople need to set up their laptops, they use the deployment template to create the replication objects and populate them with data on their computers. Then the salespeople would have access to customer information even when not connected to the network.

Data Distribution

- **Basic techniques**
 - Primary site ownership
 - Table level
 - Horizontal data subsets
 - Vertical data subsets
- **Advanced techniques**
 - Dynamic
 - Shared
 - Procedural

ORACLE

B-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Data Distribution

There are many ways to distribute data, depending on your needs for the availability and ownership of data.

Basic techniques are:

- Primary site ownership: Must be implemented through application code and cannot be configured within Oracle Replication.
 - Table level:
 - Horizontal partitioning of table data by using a WHERE clause
 - Vertical partitioning of table data by specifying a subset of columns
- Note:** Ownership of data subsets must be implemented through application code. This cannot be configured with Oracle replication.

Advanced techniques include:

- Dynamic ownership
- Shared ownership with update conflict detection and resolution
- Procedural-level replication

Advanced Techniques Replication Model

- **Update-anywhere model**
- **Automatic data change conflict detection**
- **Requires a methodology for consistency and convergence of data**
- **Dynamic**
- **Shared**

ORACLE

B-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Main Features of the Oracle Replication Model

- Supports an update-anywhere model, and is also known as Advanced Replication. Using Shared Ownership, you can update the same data on multiple sites.
- Can be implemented with either multimaster or updatable materialized views or a combination of both
- All sites can be updated and updates are propagated to all sites maintaining replicas
- Supports automatic data change conflict detection with application-level resolution capabilities for row-level replication
- Requires application developers to employ a methodology or a combination of methodologies to ensure consistency and convergence of data over time (conflict resolution)

Row-Level Replication

- **Default method in advanced replication**
- **Changes are propagated at the row level**
- **Can be optimized to minimize network traffic**
- **Performs conflict detection by default**
- **Supports conflict resolution procedures**
- **Easy to implement**

ORACLE

B-17

Copyright © Oracle Corporation, 2001. All rights reserved.

Features of Row-Level Replication

Typical transaction processing applications modify small numbers of rows per transaction. Such applications at work in an advanced replication environment usually depend on the row-level replication mechanism.

With row-level replication:

- Applications update the local replica of a table with standard data manipulation language (DML) statements.
- Internal triggers at the local site automatically queue the table changes as deferred RPCs, which are used to replicate the transactions to other sites in the system. These changes are then applied at the remote site using internal packages.
- At scheduled intervals, the originating site pushes deferred transactions in the queue to their destinations.
- At scheduled intervals, the originating site purges the transactions that have been successfully pushed to other destinations.

When application data is being changed by multiple sites at the same time with asynchronous row-level replication, data conflicts are possible. Oracle automatically detects uniqueness, update, and delete conflicts. Conflict resolution can be used to automatically resolve errors when they are detected.

Procedural Replication

- **Replicates only the call to a stored procedure rather than the individual DML operations**
- **Provides better performance for large transactions**
- **Is suitable for purging or archiving old data**
- **Should include conflict detection and resolution routines**
- **Can be used in conjunction with row-level replication**

ORACLE

B-18

Copyright © Oracle Corporation, 2001. All rights reserved.

Features of Procedural-Level Replication

Advantages

- In procedural-level replication, procedures that act on data are propagated instead of row-level changes, reducing network traffic.
- Procedural-level replication can offer better performance and more control.
- You can perform mixed procedural and row-level replication on the same data.

Disadvantages

- Procedures must be executed serially or be written so they can be serialized when run with other procedures.
- Replicated procedures must detect and resolve conflicts themselves.
- It is more complicated to propagate than row-level changes:
 - Data modification procedures must be packaged and replicated to all master sites.
 - A wrapper for each package must be generated at each site to guarantee consistency.
 - The call to the stored procedure can then be replicated to all sites.

Restrictions

- Only IN parameters are supported for replicated procedures.
- You can perform procedural replication only when *all* of the master groups on a master site are replicating data normally.

Administration Tools for Replication

- **Replication Manager**
- **Replication Management API**
- **Replication Catalog**

ORACLE

B-19

Copyright © Oracle Corporation, 2001. All rights reserved.

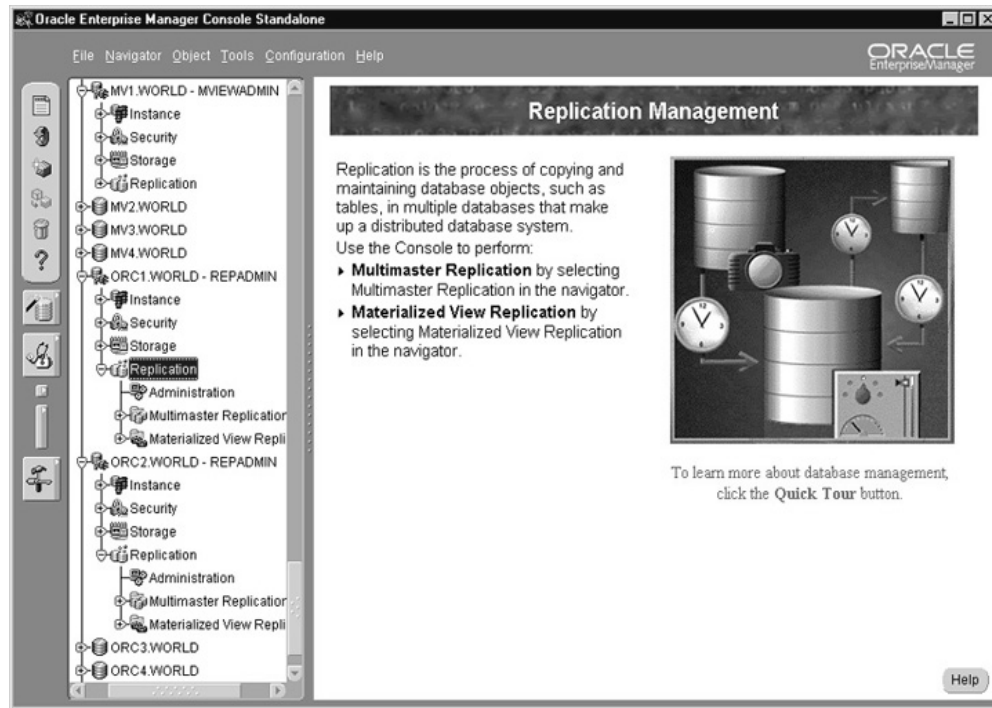
Administration Tools for Replication

Oracle's Enterprise Manager Console provides a powerful GUI interface to help manage your replication environment.

The replication management API is a set of PL/SQL packages that provide you with an application programming interface (API) to build customized scripts for replication administration. The Replication Management API is a command-line alternative to Enterprise Manager. The replication catalog is a distinct set of data dictionary tables and views that maintain administrative information about replication objects and groups at each site.

In a replication environment, all DDL statements must be issued either with Enterprise Manager or by using the DBMS_REPCAT package. When you use either of these two interfaces, the DDL is replicated to all sites within the group. DDL statements issued directly through SQL*Plus are not replicated to other sites and may cause failures in the replication system.

Replication Management GUI Tool



B-20

Copyright © Oracle Corporation, 2001. All rights reserved.

Replication Management GUI Tool

The Replication Management tool is part of Oracle Enterprise Manager and provides a graphical user interface (GUI) for setting up, managing, and monitoring a replication environment. The Replication Management tool includes wizards that guide you through many important operations. You can use the Replication Management tool to manage both multimaster and materialized view replication environments.

A GUI that supports advanced replication has existed for some time in the following applications:

- Replication Manager with Oracle Enterprise Manager (OEM) 1.6
 - Stand-alone product, multiple databases in the same window
- Replication Manager with Oracle Enterprise Manager 2.0
 - Stand-alone product, only one database accessible at a time
 - Shipped with Oracle 8.0
- Replication Manager with Oracle Enterprise Manager 2.1
 - Integrated with OEM console, needs an Oracle Management Server (OMS) to run
 - Shipped with Oracle8i
- DBA*Studio with Oracle Enterprise Manager 2.2
 - Stand-alone product or called from OEM console
 - Shipped with Oracle8i, release 8.1.7

Summary

In this lesson, you should have learned about:

- **Various types of configuration for replication**
- **Asynchronous and synchronous replication**
- **Basic components of a replication system**
- **Tools used to manage a replicated environment**

ORACLE



Content Management and Oracle9i

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

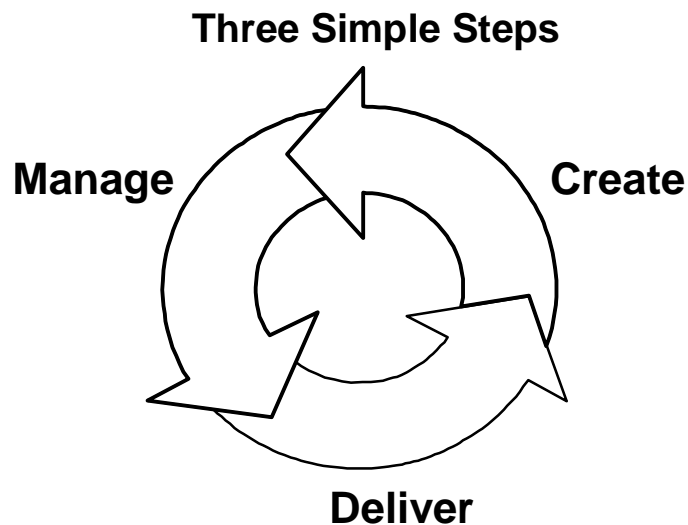
Objectives

After completing this lesson, you should have a basic knowledge of the following Oracle components:

- **Internet File System**
- **Portal**
- **Wireless Edition**
- **InterMedia**
- **Text**
- **Syndication and Dynamic Web Services**
- **Spatial / E-Location Services**
- **Workspaces**
- **Ultra Search**
- **XML Services**

ORACLE

What Is Content Management?



ORACLE

C-3

Copyright © Oracle Corporation, 2001. All rights reserved.

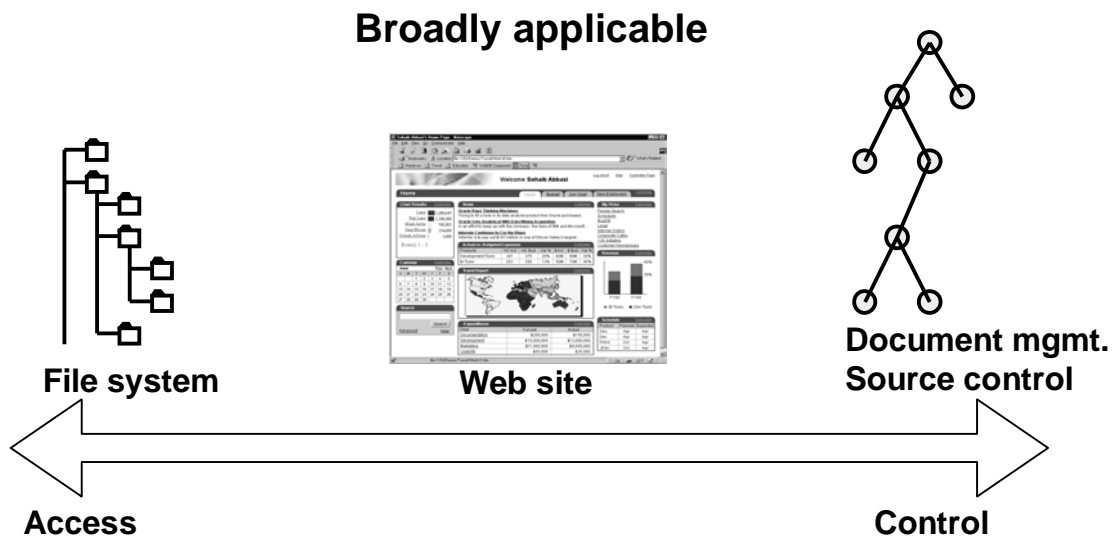
What Is Content Management?

Because every other content management presentation begins with the question, “What is content management?” let’s not ask that question here. Instead, let’s look at content management as a set of challenges centered around three areas of activity:

- Creating content. In other words, authoring documents, either individually or collaboratively, and storing them in some kind of repository
- Managing content. Taking the steps necessary to control who can access and edit content and how it is organized in the repository
- Delivering content. Packing and publishing content is the final step.

For devotees of content management, these processes are familiar and are often described under the rubric, *lifecycle management*. However, content management today requires much more than just automating or streamlining the process of creating, publishing, and retiring content.

Where Is Content Management Today?



ORACLE

C-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Where Is Content Management Today?

In all three phases of create, manage, and deliver, content management has requirements that should not conflict, but do. These conflicts are purely accidents of technology, not something inherent in how content providers work.

Between the two poles of access and functionality lie many repositories used for content management today. The most accessible repositories are file systems, optimized for ease of use and access and instantly compatible with hundreds of different popular authoring tools. However, file systems are notoriously unmanageable. They provide little in the way of collaborative features, forcing users to choose between primitive manual methods and expensive content management applications.

At the other end of this continuum of tradeoffs, are the most sophisticated content management applications, such as document management systems and source control systems. While these provide much control over who can read, edit, or dispose of content, as well as automating many content management tasks (such as staging content for publication), they sacrifice ease of use and access for this degree of control. In fact, these systems create a whole category of content managers whose primary job is to manage the flow of content from creators to consumers. While every journal may need an editor, not every internal Web site needs a Webmaster.

Where Is Content Management Today? (continued)

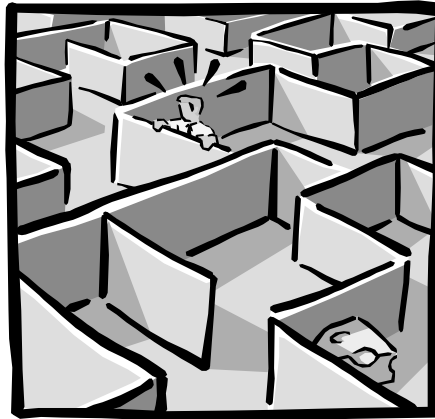
Web sites lie somewhere between these two extremes. While they are not as difficult to manage as many workgroup products or document management systems, they are not as easy as a file system for creators and consumers to access.

In short, there is no good resolution between the demands of access and control, unless you change the technology designed to meet these requirements. Platforms that are as easy to use as a file system, but at the same time capable of enforcing business rules about content in the way a document management system can, would go a long way toward resolving this dilemma.

Customer Challenge

Manage content using disparate applications:

Document
management
Knowledge
management
Configuration
management
Catalog
management



Web Content
management
Digital Asset
management
Media Asset
management
Site
management

Object DB
XML repositories
File systems

ORACLE

C-6

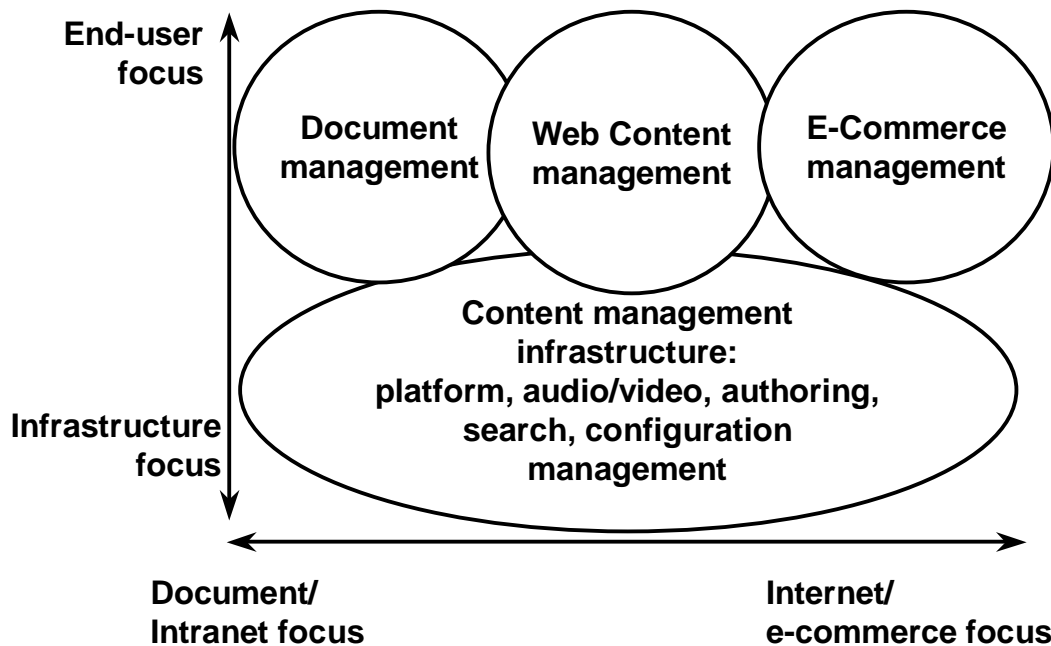
Copyright © Oracle Corporation, 2001. All rights reserved.

Customer Challenge

What makes this dilemma even more painful, and the resolution of it even more urgent, is the fact that many organizations have not implemented just one content management system, but many. Over time, as people within the organization have recognized new needs or vendors have stopped by to peddle new content management wares, the number of content management applications has proliferated. Many of these applications do essentially the same things: manage collaboration; make it easier to publish content; help to better describe, organize, and search for particular kinds of content; tailor information for particular audiences.

This proliferation of content management technologies has led many organizations to be justifiably skeptical about deploying new bits of technology advertised to solve their content management woes. Increasing IS overhead that forces users to learn and adopt to new systems, and committing to a relationship with an additional vendor are strong disincentives for organizations to take on new content management technologies.

Market Segmentation



ORACLE

C-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Market Segmentation

In short, content management is no longer a problem for application vendors to solve. Part of the problem is that, although some of the basic requirements cross the boundaries of content management applications, others do not. The companies who have made document management their focus do not necessarily have multimedia asset management as their core competence. Nor can they simply keep adding new features to ever more complex applications and proclaim that they have met the needs of a new market segment.

Even more significantly, some of the core content management requirements, such as, “How can I scale to terabytes of data?” or, “How can I scale to 1 million people browsing my web site?” cannot be solved by the application vendors alone. Instead, the platform vendors, the people who design the file systems and databases that power content management systems, have a critical role to play in addressing these requirements.

It’s not unreasonable, therefore, to push core content management capabilities down to the platform level first, so that applications can build on these capabilities. Application developers can then refine these capabilities to fit their specific markets. For example, nearly every content management application needs content-based searching and extensible metadata. The type of content to be searched, and the kinds of attributes to be added to these bits of content, will vary widely from application to application. And platform vendors can continue to play a constructive role providing scalability, reliability, security, and performance as needed.

In short, the ideal world looks something like this graphic.

Playing to Oracle's Strengths

- **Scalability**
- **Performance**
- **Reliability**
- **Searchability**
- **Security**
- **Manageability**

ORACLE

C-8

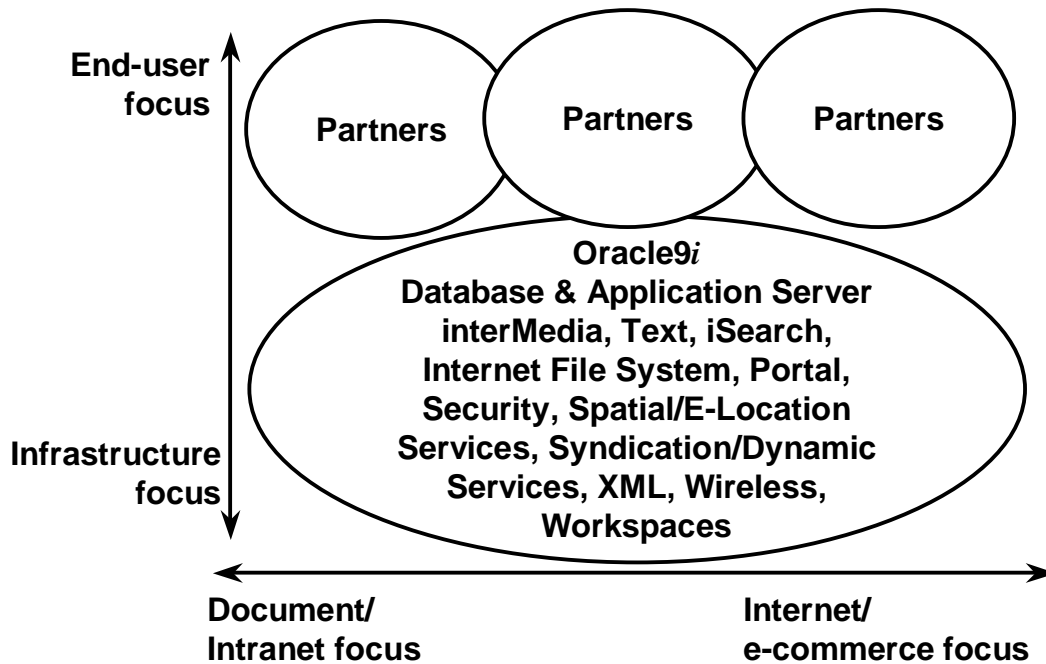
Copyright © Oracle Corporation, 2001. All rights reserved.

Playing to Oracle's Strengths

Many of the platform requirements for content management play to the strengths of the Oracle9i database and application server platforms. As content repositories grow both in the amount of content they store and the number of people accessing them, scalability, performance, and reliability grow in importance. Content needs to be easy to find, so features like search are critical. At the same time that you want to make content accessible, you also want to make sure that more sensitive information remains secure, and that your administrators can effectively manage the systems where you store your content.

These are exactly the capabilities that Oracle Corporation has been perfecting for the last 25 years. Some of these capabilities are fundamental to relational databases; others are content-specific features that Oracle has added to the database, such as the multimedia features provided by interMedia.

Oracle Content Management Positioning



ORACLE

C-9

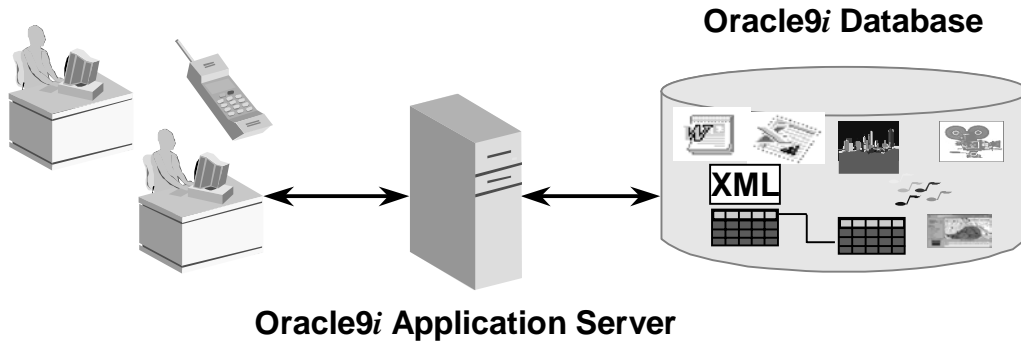
Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Content Management Positioning

It should not be surprising, then, that Oracle has created exactly the kind of relationship between itself as a platform vendor and partners as application vendors as described earlier. Oracle provides the core services for content management, and Oracle partners, such as ArborText, Macromedia, Bulldog, and others, build on this platform to address the specific needs of different content management market segments.

Content Management: An Oracle Focus

Single platform for creating, managing, and delivering personalized, rich content to any device



ORACLE

C-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Content Management: An Oracle Focus

Content management is therefore an important focus for Oracle Corporation, one that will receive a great deal of attention and investment in Oracle9i and beyond. The basic goal of Oracle's efforts is to provide the single platform for creating, managing, and delivering personalized, rich content to any device. There are many capabilities and features implied by that simple mission statement, and these will be discussed in this presentation.

Oracle9i Application Server: Content Management Features

- **Internet File System**
 - File system and content management features combined
- **Portal**
 - Create, manage, and deliver web content
- **Wireless Edition**
 - Deliver content to wireless devices

ORACLE

C-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i Application Server: Content Management Features

These features and capabilities can be summarized as follows:

- The Oracle Internet File System (9iFS) provides both an out-of-the-box file system for storing and managing content in the database, and a robust development platform for developing content management applications.
- Oracle Portal simplifies the process of delivering content to the intranet and Internet, and provides a framework for content providers to publish.
- The Wireless Edition of Oracle9i is designed to push content from the database into wireless devices.

Oracle9i Database: Content Management Features

- **interMedia**
 - Audio, video, image, wireless BMP, IVR
- **Text**
 - Powerful text search and indexing
- **Ultra Search**
 - Unified search for data from anywhere on the Web

ORACLE

C-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i Database: Content Management Features

These features and capabilities can be summarized as follows:

- Oracle interMedia extracts metadata from rich media files and lets you manipulate these files within the Oracle9i database.
- Oracle Text indexes textual content stored in the database and lets you perform sophisticated content-based queries on these indexes.
- Oracle Ultra Search builds on Oracle Text to provide a unified, searchable index of content stored in databases, file systems, and web sites.

Oracle9i Database: Content Management Features

- **Spatial and E-Location services**
 - Region modeling, proximity recall, and so on
- **Syndication server and Dynamic services**
 - Deliver dynamic content to particular subscribers
- **Workspaces**
 - Manage long-duration projects
- **XML services**
 - Parse and render content
 - Deliver content in different formats and styles

ORACLE

C-13

Copyright © Oracle Corporation, 2001. All rights reserved.

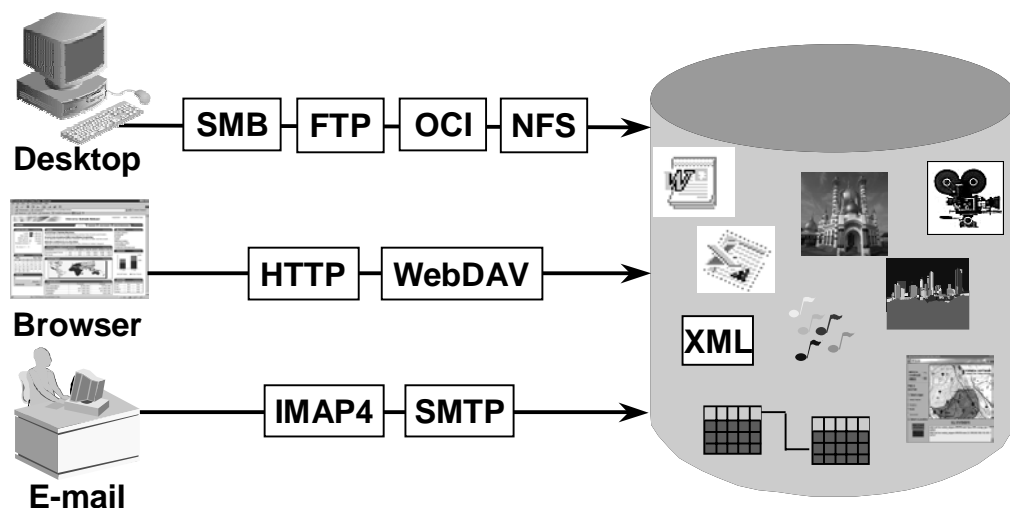
Oracle9i Database: Content Management Features (Continued)

Other features and capabilities are:

- Oracle Spatial/eLocation Services lets you add regional metadata to content and perform spatial searches.
- Dynamic Services and the Syndication Server make it easy to aggregate content and deliver it to subscribers.
- Workspaces help version content in the database.
- XML services like the Oracle XML parser help you parse and render XML content, making it possible to tailor XML-based content to different formats and audiences.

Access Content Using Any Protocol

- Store, parse, render, search, retrieve
- Text, multimedia, XML, object-relational data



ORACLE

C-14

Copyright © Oracle Corporation, 2001. All rights reserved.

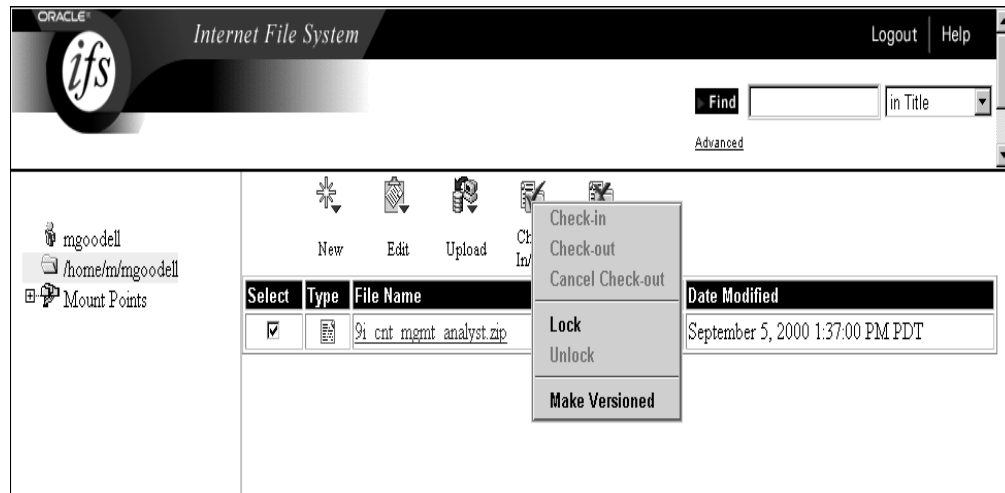
Access Content Using Any Protocol

The first step in authoring content is, of course, accessing the place where it will be stored. To put content into the Oracle9i database, you have a variety of protocols available. Aside from the traditional relational interfaces, Oracle9i is accessible through file-based network protocols like WebDAV, FTP, and IMAP4. These protocols let users access content from a variety of different platforms (Windows, UNIX, Macintosh, and network computers) and applications (for example, e-mail clients, productivity applications running on Windows, web authoring tools using WebDAV).

Whether the content is traditional row/column data, multimedia, XML, or text, you can store it and access it in Oracle9i.

Collaborate on Content Creation

- Checkin and checkout
- ACL-based security
- Version control
- Groups

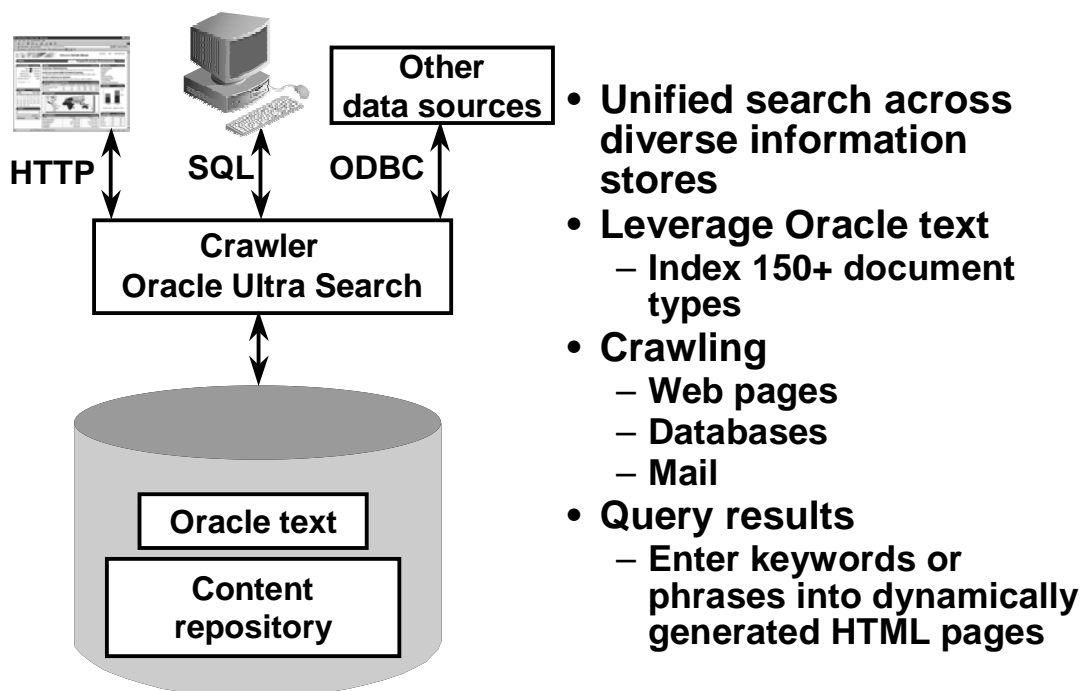


Collaborate on Content Creation

Collaboration is a key part of content management. In fact, it might be the most fundamental content management requirement. If different users cannot effectively collaborate on the same files, their content is not being well managed. Collaborative features are available in both the Oracle Internet File System and Oracle Portal. For example, checkin, checkout, and ACL-based security help to control read and write access to files.

Version control maintains earlier versions of files, in order to leave a history of changes and to make it possible to recover to earlier versions.

Index and Search Content



ORACLE

C-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Index and Search Content

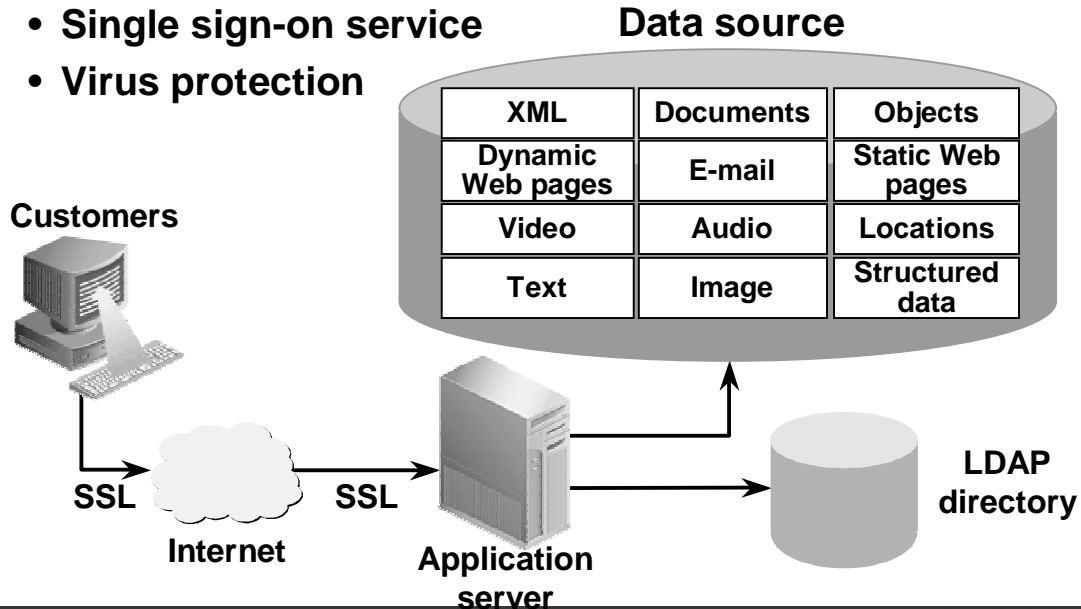
While you are creating content, the platform should be indexing it for later searches. Oracle Text, a long-established part of the Oracle database, can index textual content, either short text strings or files stored as LOBs, that is inserted in the database. Once indexed, content-based searches can be much faster than searches in a file system. Oracle Text lets you perform more sophisticated searches as well, such as synonym, theme, and fuzzy matching searches.

New in Oracle9i is Ultra Search, which extends the capabilities of Oracle Text to content stored in databases, file systems, and Web sites. Ultra Search can crawl these data stores and build the index entries needed to perform fast and powerful searches.

Similarly, interMedia AVI can extract encoded metadata from audio, video, and image files, such as play length and color depth, and then store these attributes with the file. interMedia AVI also lets you manipulate rich media as you access it, by changing formats, cropping images as you query them, and so on.

Store Content in a Secure Repository

- User and document access control
- Login and password protection
- Single sign-on service
- Virus protection



ORACLE

C-17

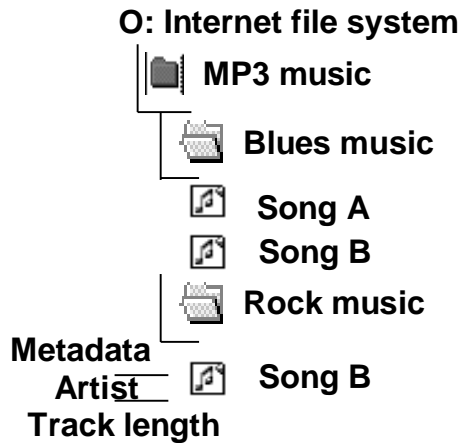
Copyright © Oracle Corporation, 2001. All rights reserved.

Store Content in a Secure Repository

After you create content, you must manage it. The most obvious connotation of the word management is security, and Oracle9i provides many high-security features. The Login Server and Oracle Internet Directory consolidate security management into a single sign-on for database applications and a single LDAP service for user and group management. ACL-based security in products like the Oracle Internet File System and Oracle Portal provide fine-grain security. Other services, like virus protection plug-ins provided by Oracle partners, keep content both safe and uncorrupted. Oracle's business depends on making customers comfortable putting their most sensitive data into the Oracle9i database, so over 25 years of security expertise are now at the disposal of content creators, consumers, and application developers.

Organize Content Efficiently

- **Multiple file folders**
- **Extensible metadata: Add or derive application specific information about the content**



ORACLE

C-18

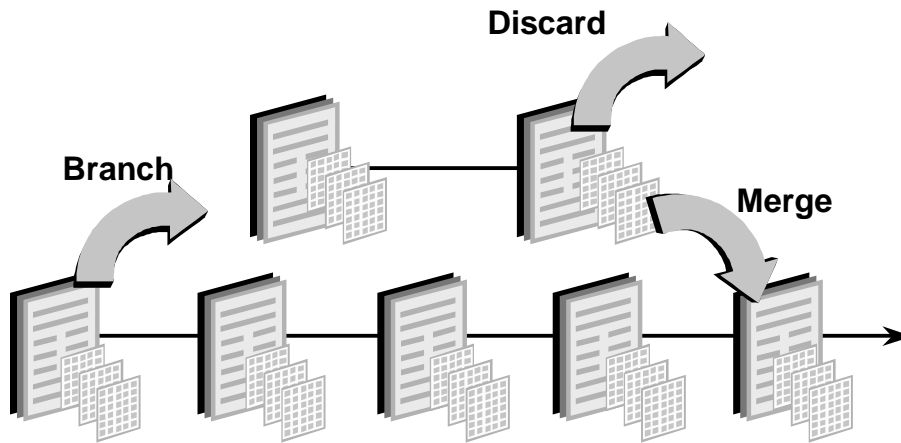
Copyright © Oracle Corporation, 2001. All rights reserved.

Organize Content Efficiently

Managing content also requires organizing it effectively. Storing the same file in multiple Oracle Internet File System folders, for example, is more manageable than copying the same file into multiple folders and then managing each copy. Because organization also depends on what kind of metadata you can assign to files, extensible metadata in products like the Oracle Internet File System and Oracle Portal make it easier to manage your content. Automatic metadata extraction from multimedia files has long been the strength of interMedia AVI, and is a capability that is now accessible through the Oracle Internet File System.

Manage Long-Duration Projects

Workspaces: Allow concurrent projects to coexist against different versions of content



ORACLE®

C-19

Copyright © Oracle Corporation, 2001. All rights reserved.

Manage Long-Duration Projects

Management also means organizing content into projects. Workspaces, a set of PL/SQL procedures designed to help organize database content into projects, provides a versioning interface to all content stored in Oracle9i. Workspaces lets you create nested projects, perform merges, and identify the currently published version of the project.

Develop Custom CM Applications

- Oracle Portal Developer kit
- Oracle *i*FS Developer kit
- Oracle Media Developer kit
- Oracle XML Developer kit
- Oracle Syndication Server framework
- Oracle interMedia APIs
- Oracle *i*Search API
- Oracle *j*Location APIs

ORACLE

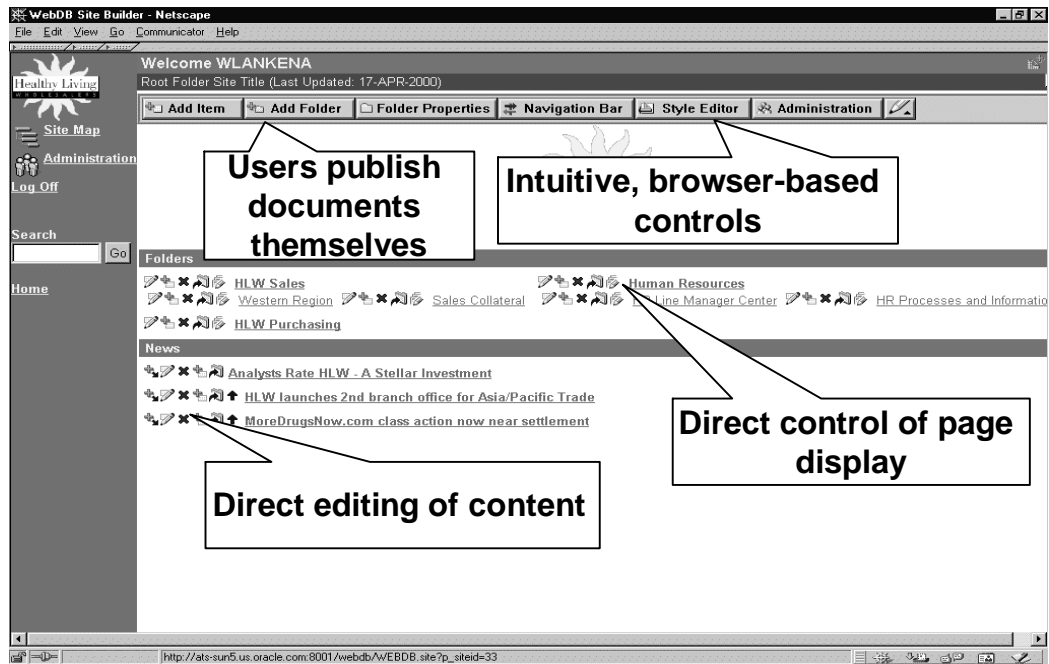
C-20

Copyright © Oracle Corporation, 2001. All rights reserved.

Develop Custom CM Applications

Of course, no content management system can provide everything that each organization needs. Every content management feature in Oracle9i has its own developer kit, providing PL/SQL, Java, and XML interfaces. Oracle partners, depending on these development environments, are building sophisticated, focused applications for their businesses on the Oracle9i content management platform.

Self-Service Publishing



ORACLE

C-21

Copyright © Oracle Corporation, 2001. All rights reserved.

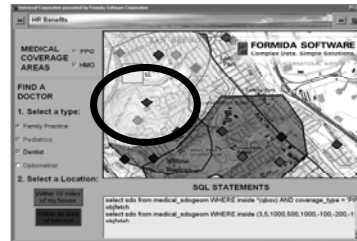
Self-Service Publishing

Finally, you need to deliver content. Oracle Portal provides an easy-to-use interface for publishing content, and you can define different groups and group administrators responsible for particular content areas.

Even more significant, however, is the portlet framework in Oracle Portal. This framework provides the hooks needed for content providers to register the different types of content available within the portal, to which users can then subscribe in their personalized portal. The portlet framework also lets you create reports based on content stored in the database, and portlets provide access to other Oracle9i components, such as the Oracle Internet File System, and Oracle applications. In short, Oracle Portal provides a web-based desktop that lets you both publish your content as well as subscribe to other content sources.

Location-Enabled Content

**Geocoding, region modeling,
and distance queries for
location information**



**Perform routing
computations
in the database**



**Mobile applications can
directly access location
services**

ORACLE

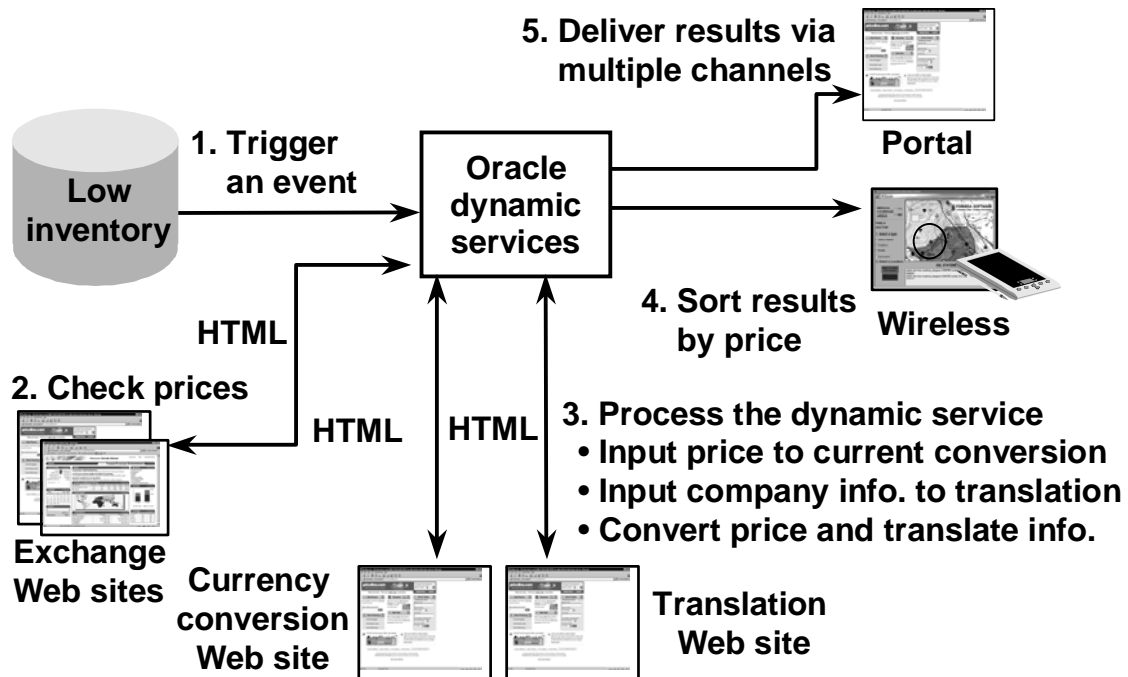
C-22

Copyright © Oracle Corporation, 2001. All rights reserved.

Location-Enabled Content

In the process of delivering content, you may want to indicate the geographical region to which it applies. Some content may appear on one regional version of your Web site, but not in others. An instructional video may be designed for one region, but not for others. Geographic encoding, or geocoding, of content in the database helps to provide this kind of location intelligence. Users can then perform geographic searches on content that applies to a particular area.

Aggregated Content



ORACLE

C-23

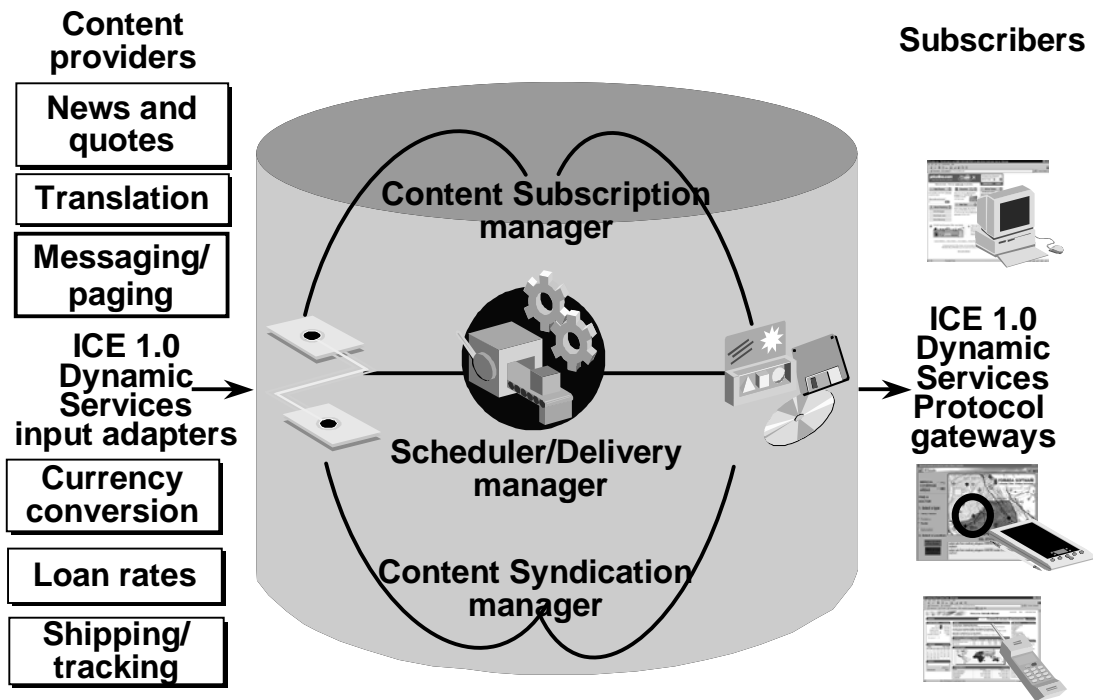
Copyright © Oracle Corporation, 2001. All rights reserved.

Aggregated Content

Oracle Dynamic Services is a programmatic framework for incorporating Internet services into portals, wireless, and B2B applications. Dynamic services let you use many standard protocols, such as SOAP and ICE, to help broadcast that this content is available and make it available for subscription aggregation.

Intelligent aggregation is one of Dynamic Services' important capabilities. For example, you can query a variety of separate sources to find the best price on laptops for sale in Paris. If you are traveling abroad, Dynamic Services can help aggregate retail, currency conversion, and consumer reviews into a single query.

Syndicated Content



ORACLE

C-24

Copyright © Oracle Corporation, 2001. All rights reserved.

Syndicated Content

Aside from these kinds of one-off queries, Dynamic Services provides the framework for subscribing to content sources. The Syndication Server, based on Dynamic Services, provides subscription services that can push content out to portals, wireless devices, and other platforms for delivering content.

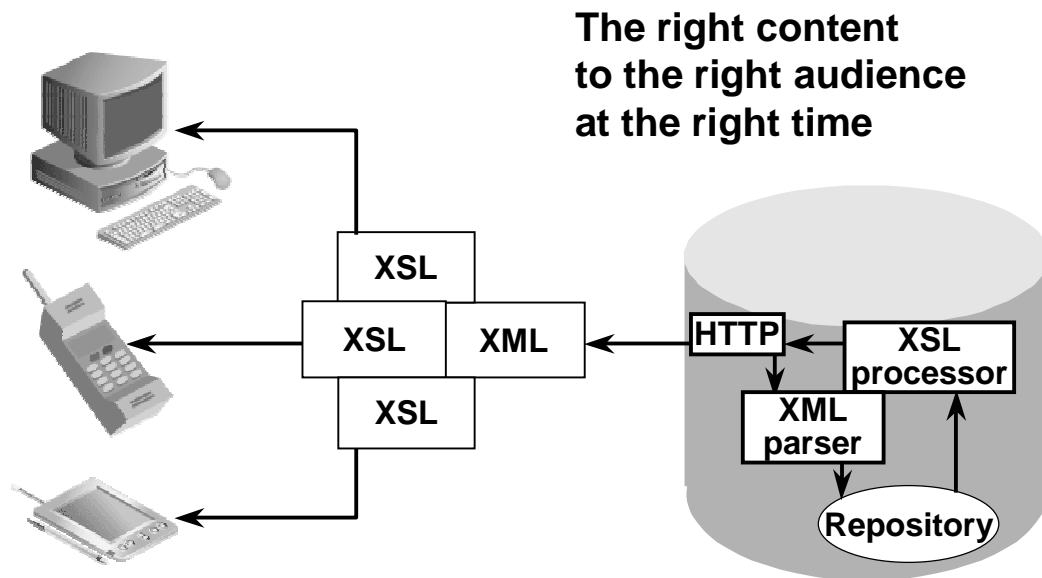
The Syndication Server includes:

- Subscription Manager to manage subscriptions for customers, perform content updates (partial or full), and track subscription activities
- Content Syndication Manager to coordinate syndication execution flow and manage affiliates profiles, delivery policies, targeted offers
- Syndication Affiliate Profile Manager used to create, modify, and remove user accounts and business-related information
- Syndication System Administrator to assist the administrator to supervise the system

It also provides a Performance Monitor to track down any performance-related issues, such as a network bottleneck due to a large amount of full content updates.

The Syndication Server capabilities in Oracle9i support the Internet Content Exchange (ICE) 1.0 protocol, and includes adapters to allow for subscription and delivery from non-ICE compliant providers and syndicators.

Any Content to Any Device with Oracle9i AS Wireless Edition



ORACLE

C-25

Copyright © Oracle Corporation, 2001. All rights reserved.

Any Content to Any Device With Oracle9i AS Wireless Edition

As the world becomes increasingly wireless, moving content to wireless devices, such as cellular phones and PDAs becomes increasingly important. Users are getting accustomed to searching for data, such as movie ticket availability, and performing transactions like buying tickets through these cellular devices. Wireless capabilities in Oracle9i help to publish content stored in Oracle9i to these wireless devices, and to provide the application platform needed to let users purchase tickets, read e-mail, or receive alerts when documents are updated.

Summary

In this lesson, you should have learned to explain:

- **How Oracle components fit together to deliver content over the Internet**
- **What function is performed by each Oracle component**

ORACLE