Cognizant
Passion for making a difference

Handout: Oracle 10g Architecture

Version: 10gARCHITECTURE/Handout/0308/1.0
Date: 31-03-08

Cognizant
500 Glen Pointe Center West
Teaneck, NJ 07666
Ph: 201-801-0233
www.cognizant.com

TABLE OF CONTENTS

Introduction

About this Module

The module provides an overview about the basics of Oracle 10g Architecture.

Target Audience

This module is designed for the entry level trainees.

Module Objectives

After completing this module, you will be able to:

- Explain RDBMS concept and Oracle Database
- Describe the architecture of Oracle Database and its Instance
- List Oracle Background Processes
- Define Concurrency and Read Consistency
- Describe Database Security and its Oracle Implementation
- List the use of Data Dictionary and Oracle Utilities

Pre-requisite

The trainees need to have an idea of the following:

- General Database Overview
- Structured Query Language
- Relational DBMS

Session 01: Introduction to RDBMS and Oracle

Learning Objectives

After completing this session, you will be able to:

- Understand the RDBMS concepts
- Install Oracle Database
- Start up the database
- Access the database
- Understand How Oracle Works

Understand the Database

Data base is a persistent repository of logically related data. The purpose of a database is to store and retrieve related information. Data base Provides platform for easy Data management, facilitate data sharing and Reduces data redundancy .

Understand the RDBMS Concepts

DBMS is a software system that enables users to define, create and maintain the database and provides controlled access to database. RDBMS stands for Relational Database Management System is a database which stores data in the form of tables and the relationship among the data is also stored in the form of tables. Relationship between the tables made the representation, loading and retrieval of the data easy.

RDBMS provides a set-oriented database language. For most RDBMS, this set-oriented database language is SQL. *Set oriented* means that SQL processes sets of data in groups.

Codd's Rule

Dr. E.F. Codd, an IBM researcher, first developed the relational data model in 1970. In 1985, Dr. Codd published a list of 12 rules that concisely define an ideal relational database, which have provided a guideline for the design of all relational database systems ever since

Rule 1: The Information Rule

All data should be presented to the user in table form.

Rule 2: Guaranteed Access Rule

All data should be accessible without ambiguity. This can be accomplished through a combination of the table name, primary key, and column name.

Rule 3: Systematic Treatment of Null Values

A field should be allowed to remain empty. This involves the support of a null value, which is distinct from an empty string or a number with a value of zero. Of course, this can't apply to primary keys. In addition, most database implementations support the concept of a nun- null field constraint that prevents null values in a specific table column.

Rule 4: Dynamic On-Line Catalog Based on the Relational Model

A relational database must provide access to its structure through the same tools that are used to access the data. This is usually accomplished by storing the structure definition within special system tables.

Rule 5: Comprehensive Data Sublanguage Rule

The database must support at least one clearly defined language that includes functionality for data definition, data manipulation, data integrity, and database transaction control. All commercial relational databases use forms of the standard SQL (Structured Query Language) as their supported comprehensive language.

Rule 6: View Updating Rule

Data can be presented to the user in different logical combinations, called views. Each view should support the same full range of data manipulation that direct-access to a table has available. In practice, providing update and delete access to logical views is difficult and is not fully supported by any current database.

Rule 7: High-level Insert, Update, and Delete

Data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

Rule 8: Physical Data Independence

The user is isolated from the physical method of storing and retrieving information from the database. Changes can be made to the underlying architecture (hardware, disk storage methods) without affecting how the user accesses it.

Rule 9: Logical Data Independence

How a user views data should not change when the logical structure (tables structure) of the database changes. This rule is particularly difficult to satisfy. Most databases rely on strong ties between the user view of the data and the actual structure of the underlying tables.

Rule 10: Integrity Independence

The database language (like SQL) should support constraints on user input that maintain database integrity. This rule is not fully implemented by most major vendors. At a minimum, all databases do preserve two constraints through SQL.

No component of a primary key can have a null value. (see rule 3)

If a foreign key is defined in one table, any value in it must exist as a primary key in another table.

Rule 11: Distribution Independence

A user should be totally unaware of whether or not the database is distributed (whether parts of the database exist in multiple locations). A variety of reasons make this rule difficult to implement; I will spend time addressing these reasons when we discuss distributed databases.

Rule 12: No subversion Rule

There should be no way to modify the database structure other than through the multiple row database language (like SQL). Most databases today support administrative tools that allow some direct manipulation of the data structure.

Oracle History

- 1977 Software Development Laboratories, the precursor to Oracle, is founded by Larry
- 1978 Oracle Version 1, written in assembly language
- 1979 Oracle Version 2, the first commercial SQL relational database management system,
- 1983 Oracle Version 3, built on the C programming language, is the first RDBMS
- 1985 Oracle Version 5, one of the first relational database systems to operate in client/server environments
- 1988 Oracle Version 6 debuts with several major advances
- 1992 Oracle Version 7 customer support as a database
- 1998 Oracle Version 8 Database and Oracle Applications
- 1999 Oracle 8i Database.
- 2001 Oracle 9i Database adds Oracle RAC
- 2003 Oracle Database 10g, grid computing
- 2007 Oracle Database 11g.

Basic Oracle Architecture

An Oracle database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management. In general, a server reliably manages a large amount of data in a multi-user environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

Oracle Database is the first database designed for enterprise grid computing, the most flexible and cost effective way to manage information and applications. Enterprise grid computing creates large pools of industry-standard, modular storage and servers. With this architecture, each new system can be rapidly provisioned from the pool of components. There is no need for peak workloads, because capacity can be easily added or reallocated from the resource pools as needed.

The database has logical structures and physical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting the access to logical storage structures.

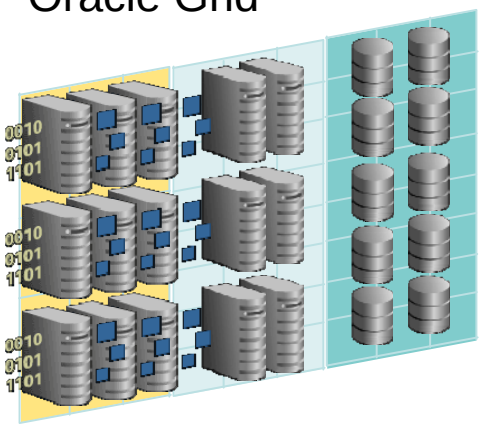
Oracle Grid Architecture

Grid computing is a new IT architecture that produces more resilient and lower cost enterprise information systems. With grid computing, groups of independent, modular hardware and software components can be connected and rejoined on demand to meet the changing needs of businesses. Grid computing also enables the use of smaller individual hardware components

Large Dedicated Server

- *Expensive costly components*
- *High incremental costs*
- *Single point of failure*

Oracle Grid



- *Low cost modular components*
- *Low incremental costs*
- *No single point of failure*
- *Enterprise service at low cost*

Capacity on Demand!

Figure: Oracle grid Architecture

Installing Oracle Database

Read the release notes with the platform-specific documentation.
http://www.oracle.com/technology/documentation

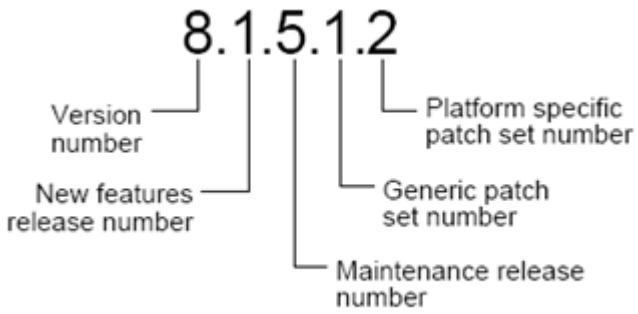


Figure: Example of Oracle Release Number

- Plan the installation; single of multiple, interactive or silent.
- Complete the pre-installation tasks.
- Install the software named, Oracle Universal Installer (OUI).
  - Use Oracle Universal Installer (OUI) to install Oracle Database and Automatic Storage Management (ASM), as well as how to clone an Oracle home.
  - Perform silent or non-interactive installations using response files, which you may want to use if you need to perform multiple installations of Oracle Database.
  - Install Java Access Bridge, which enables a screen reader with Oracle components.
  - Install and use Oracle components in different languages.
  - Remove Oracle Database.
- Complete post installation tasks.
- Get started using Oracle Database.
- To log in to Oracle Enterprise Manager 10g Database:
- Open your Web browser and enter the following URL
  - http://hostname:port/em
- In a default installation, the port number is 1158. If you are unsure of the correct port number to use, look for the following line in the `ORACLE_BASE\ORACLE_HOME\install\portlist.ini` file:
- Enterprise Manager Console HTTP Port (db_name) = port
- Log in to the database using the `SYSMAN` database user account. Enterprise Manager displays the Oracle Database home page.
- Use the password you specified for the `SYSMAN` account during the Oracle Database installation.

Oracle Database Installation Methods

There are two methods that you can use to install Oracle Database:

- **Basic:** Select this installation method if you want to quickly install Oracle Database. This installation method requires minimal user input. It installs the software and optionally creates a general-purpose database using the information that you specify on this window. It is the default installation method.
- **Advanced:** Select this installation method if you want to complete any of the following tasks:

Perform a custom software installation, in which you choose components individually, or choose a different database configuration

The Available Product Components installation window automatically selects the components most customers need in their Oracle Database installation. It also lists several components that are not selected by default, but which you may want to include. To find the listing of available components, select Advanced, and then in the Installation Type window, select Custom.

Starting Up the Database

A database administrator can perform these steps using the SQL*Plus STARTUP statement or Enterprise Manager.

When Oracle starts an instance, it reads the server parameter file (SPFILE) or initialization parameter file to determine the values of initialization parameters. After this, it allocates an SGA and creates background processes. When you start, the database instance comes into picture into system memory. Combination of the SGA and the Oracle processes is called an Oracle instance.

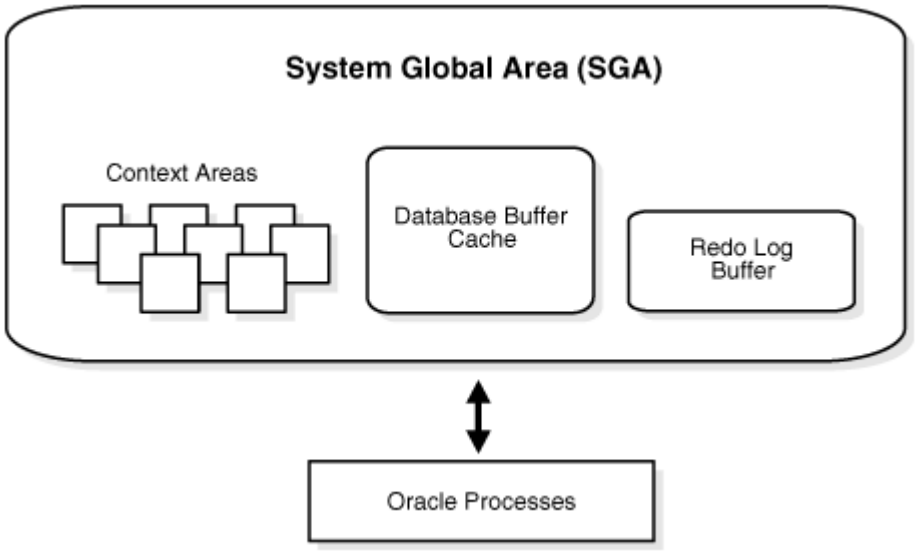


Figure: Oracle Instance

Connection with Administrator Privileges:

The user's operating system privileges allow him or her to connect, using administrator privileges. The user is granted the SYSDBA or SYSOPER privileges and the database uses password files to authenticate database administrators.

connect / as sysdba

You can use one of the following ways:

1. Use Oracle Enterprise Manager.
2. Use the SQL*Plus STARTUP statement.
3. On Windows, you can start the Oracle services.

Starting and Stopping the Database from the Microsoft Windows Services Utility

To start or stop the database:

- From the Start menu, select Programs, then Administrative Tools, and then Services.
- In the Services dialog box, locate the name of the database you want to start or stop.
- Right-click the name of the database, and from the menu, select either Start, Stop, or Pause.
- To set its start-up properties, right-click Properties, and in the dialog box, select Automatic, Manual, or Disabled.
- After logon to *http://hostname:port/em* , we can see the status of the database (see the following figure)

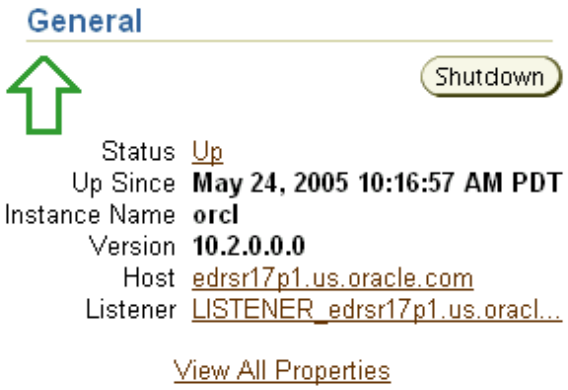


Figure: Database status through Oracle Enterprise Manager

Overview of Accessing the Database

There are several tools for interfacing with the database using SQL:

- Oracle SQL*Plus and iSQL*Plus
- Oracle Forms, Reports, and Discoverer
- Oracle Enterprise Manager
- Third-party tools

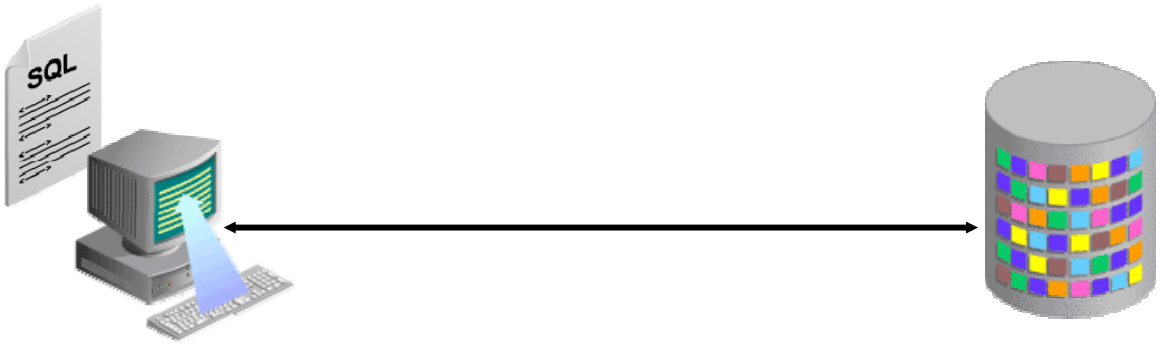


Figure: Accessing Database

Oracle Net, a component of Oracle Net Services, enables a network session from a client application to an Oracle database server. Once a network session is established, Oracle Net acts as the data courier for both the client application and the database server. It establishes and maintains the connection between the client application and database server, as well as exchanges messages between them. Oracle Net can perform these jobs because it is located on each computer in the network.

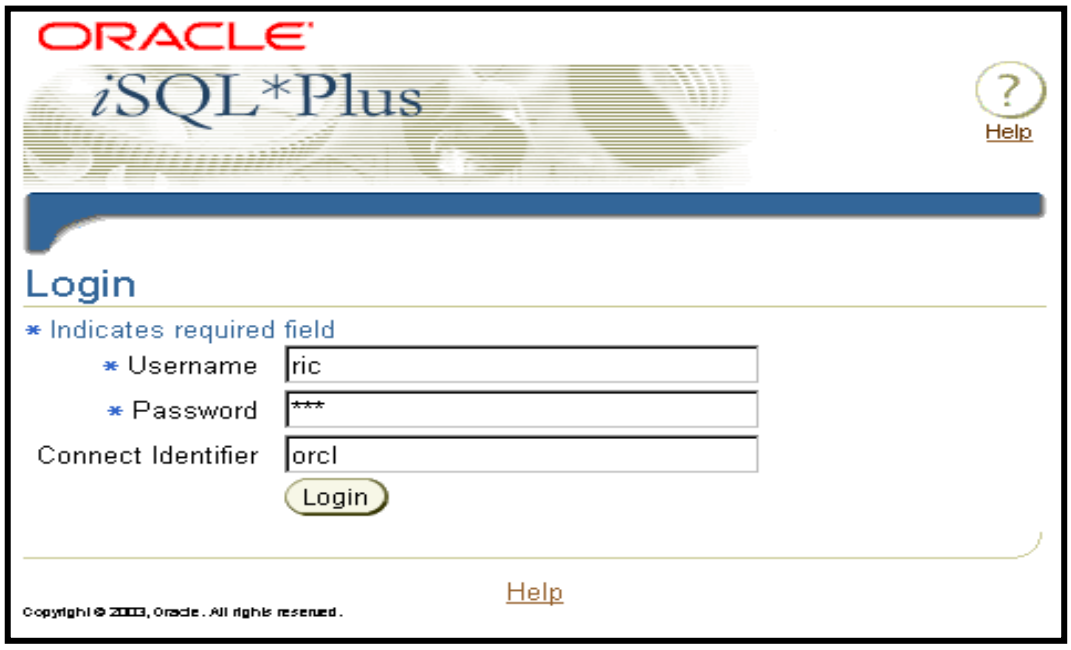


Figure: Accessing Oracle Database through iSQL*PLUS tool

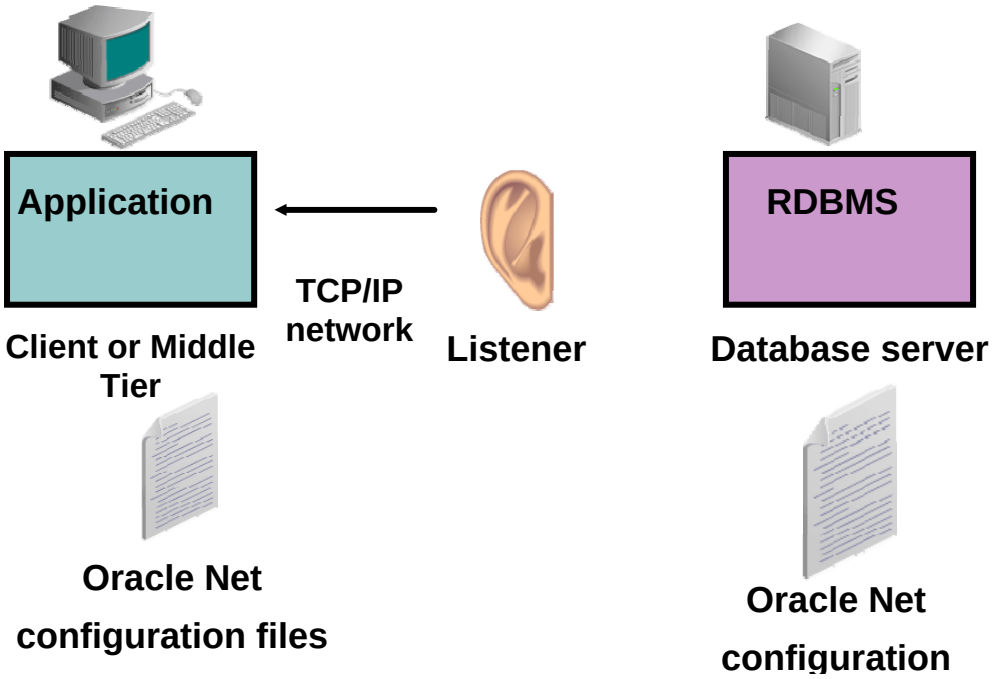


Figure: Client-Server network connectivity

How Oracle Works

The following example describes the most basic level of operations that Oracle performs. This illustrates an Oracle configuration where the user and associated server process are on separate computers (connected through a network).

1. An instance has started on the computer running Oracle (often called the host or database server).
2. A computer running an application (a local computer or client workstation) runs the application in a user process. The client application attempts to establish a connection to the server using the proper Oracle Net Services driver.
3. The server is running the proper Oracle Net Services driver. The server detects the connection request from the application and creates a dedicated server process on behalf of the user process.
4. The user runs a SQL statement and commits the transaction. For example, the user changes a name in a row of a table.
5. The server process receives the statement and checks the shared pool for any shared SQL area that contains a similar SQL statement. If a shared SQL area is found, then the server process checks the user's access privileges to the requested data, and the previously existing shared SQL area is used to process the statement.
    a) If not, then a new shared SQL area is allocated for the statement, so it can be parsed and processed.
6. The server process retrieves any necessary data values from the actual datafile (table) or those stored in the SGA.
7. The server process modifies data in the system global area. The DBWn process writes modified blocks permanently to disk when doing so is efficient. Because the transaction is committed, the LGWR process immediately records the transaction in the redo log file.

8. If the transaction is successful, then the server process sends a message across the network to the application. If it is not successful, then an error message is transmitted.
9. Throughout this entire procedure, the other background processes run, watching for conditions that require intervention. In addition, the database server manages other users' transactions and prevents contention between transactions that request the same data.

Summary

- A relational database-management system (RDBMS) consists of a collection of interrelated data and a set of programs to access those data.
- RDBMS acts as an interface between application programs and physical data files.
- An Oracle server is a relational database management system that provides an open, comprehensive, integrated approach to information management.
- Grid Computing in Oracle database can be scaled as per the demand.

Test Your Understanding

1. What is Grid Computing?
2. What are the structures involved in connecting a user to an Oracle Instance?
3. How Oracle access the data for you?
4. What are the steps to install Oracle Software?

Session 02: Oracle Database Architecture

Learning Objectives

After completing this session, you will be able to:

- Comprehend the procedure how Oracle Grid Architecture works
- Comprehend Application Architecture
- Identify the physical database structures
- Identify the logical database structures

Oracle Grid Architecture

Grid computing is a new IT architecture that produces more resilient and lower cost enterprise information systems. With grid computing, groups of independent, modular hardware and software components can be connected and rejoined on demand to meet the changing needs of businesses. Grid computing also enables the use of smaller individual hardware components

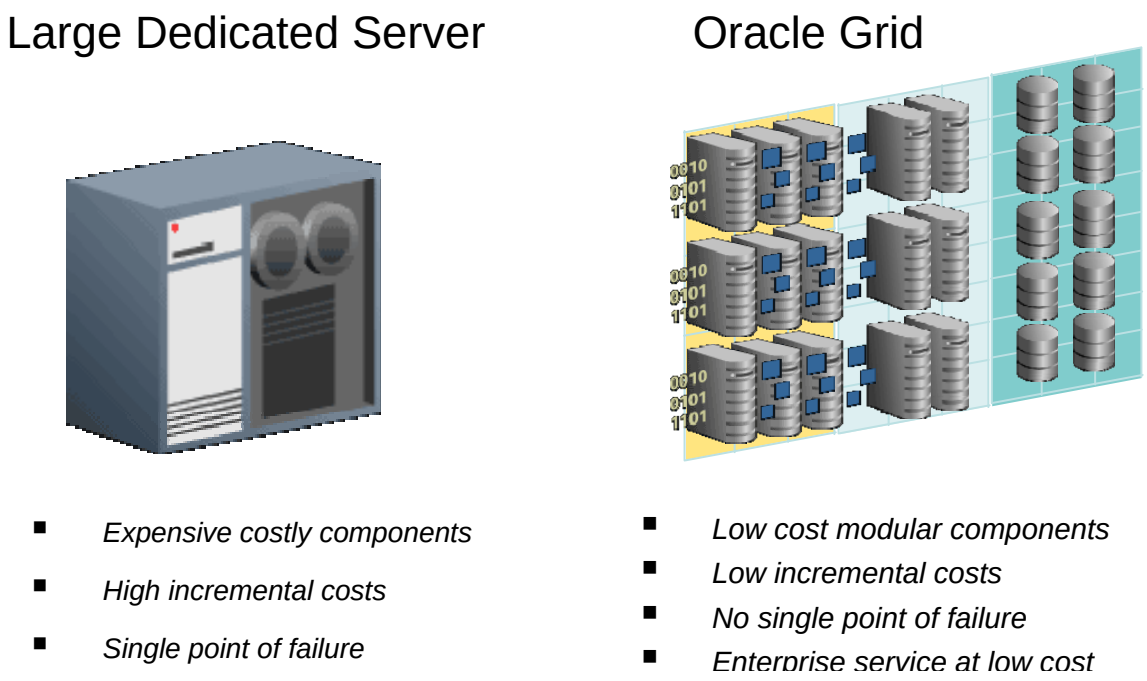


Figure: Oracle grid Architecture

Server Virtualization

Oracle Real Application Clusters 10g (RAC) enable a single database to run across multiple clustered nodes in a grid, pooling the processing resources of several standard machines.

Storage Virtualization

The Oracle Automatic Storage Management (ASM) feature of Oracle Database 10g provides a virtualization layer between the database and storage so that multiple disks can be treated as a single disk group and disks can be dynamically added or removed while keeping databases online.

Grid Management

The Grid Control feature of Oracle Enterprise Manager 10g provides a single console to manage multiple systems together as a logical group.

Oracle Grid Architecture

Client/Server Architecture: An Oracle database system can easily take advantage of distributed processing by using its client/server architecture. The client runs on a different computer than the database server, generally on a PC.

Multitier Architecture has the following components:

1. A client or initiator process that starts an operation
2. One or more application servers it can serve as an interface between clients and multiple database servers, including providing an additional level of security.
3. An end or database server that stores most of the data used in the operation

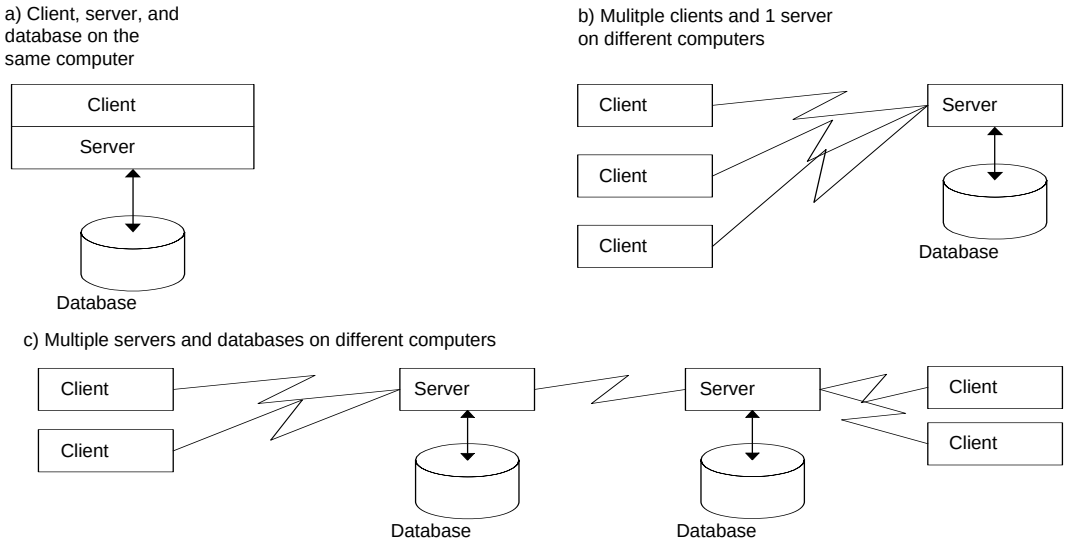


Figure: Client/Server Architecture

Logical Database Structures

The logical structures of an Oracle database include schema objects, data blocks, extents, segments, and table spaces

Schemas and Schema Objects

A schema is a collection of database objects. A schema is owned by a database user and has the same name as that user. Schema objects are the logical structures that directly refer to the database's data. Schema objects include structures like tables, views, and indexes

Some of the most common schema objects are defined in the following section.

Tables

Tables are the basic unit of data storage in an Oracle database. Database tables hold all user-accessible data. Each table has columns and rows. Oracle stores each row of a database table containing data for less than 256 columns as one or more row pieces. A table that has an employee database, for example, can have a column called employee number, and each row in that column is an employee's number.

Views

Views are customized presentations of data in one or more tables or other views. A view can also be considered a stored query. Views do not actually contain data. Rather, they derive their data from the tables on which they are based, referred to as the base tables of the views.

Like tables, views can be queried, updated, inserted into, and deleted from, with some restrictions. All operations performed on a view actually affect the base tables of the view.

Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table. They also hide data complexity and store complex queries.

Indexes

Indexes are optional structures associated with tables. Indexes can be created to increase the performance of data retrieval. Just as the index in this manual helps you quickly locate specific information, an Oracle index provides an access path to table data.

When processing a request, Oracle can use some or all of the available indexes to locate the requested rows efficiently. Indexes are useful when applications frequently query a table for a range of rows (for example, all employees with a salary greater than 1000 dollars) or a specific row.

Indexes are created on one or more columns of a table. After it is created, an index is automatically maintained and used by Oracle. Changes to table data (such as adding new rows, updating rows, or deleting rows) are automatically incorporated into all relevant indexes with complete transparency to the users.

Segments

The logical storage structures, including data blocks, extents, and segments, enable Oracle to have fine-grained control of disk space use.

Oracle Data Blocks

At the finest level of granularity, Oracle database data is stored in data blocks. One data block corresponds to a specific number of bytes of physical database space on disk. The standard block size is specified by the initialization parameter DB_BLOCK_SIZE. A database uses and allocates free database space in Oracle data blocks.

Extents

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

Segments

Above extents, the level of logical database storage is a segment. A segment is a set of extents allocated for a certain logical structure. The following table describes the different types of segments.

Segment	Description
Data segment	❑ Each table has a data segment. All table data is stored in the extents of the 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 segment	❑ Each index has an index segment that stores all of its data. ❑ For a partitioned index, each partition has an index segment.
Temporary segment	Temporary segments are created by Oracle when a SQL statement needs a temporary work area to complete execution. When the statement finishes execution, the extents in the temporary segment are returned to the system for future use.
Rollback segment	❑ The information in a rollback segment is used during database recovery: ❑ To generate read-consistent database information ❑ To roll back uncommitted transactions for users

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

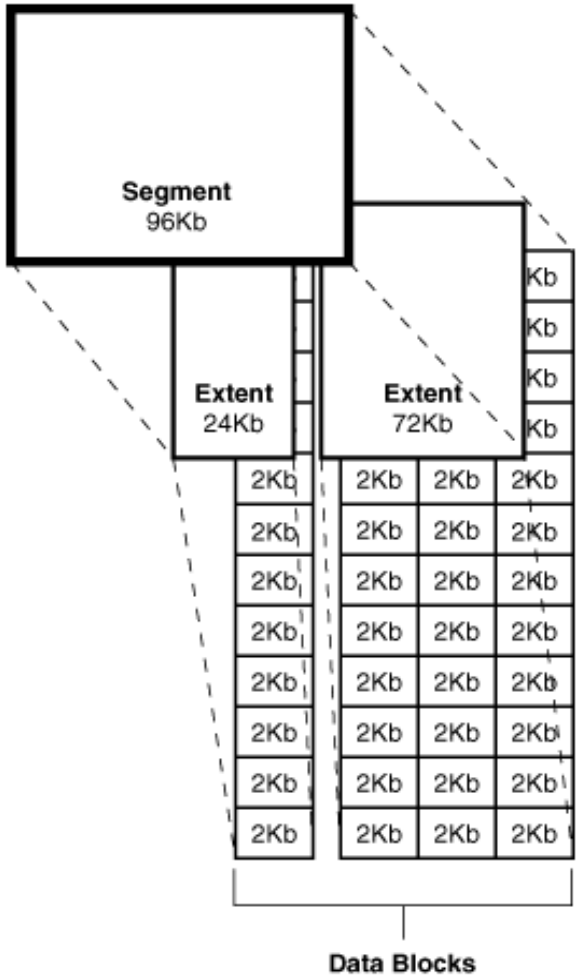


Figure: The Relationships among Segments, Extents, and Data Blocks

Table spaces

A database is divided into logical storage units called tablespaces, which group related logical structures together. For example, tablespaces commonly group together all application objects to simplify some administrative operations.

Databases, Tablespaces, and Datafiles

The relationship between databases, tablespaces, and datafiles (datafiles are described in the next section) is illustrated in following figure.

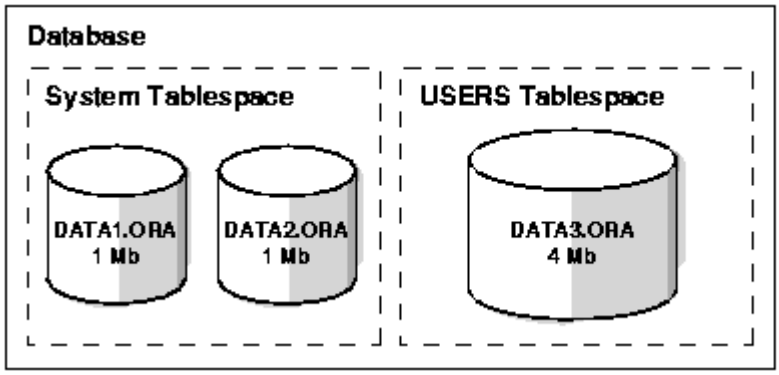


Figure: Database, Tablespaces, and Datafiles

This figure illustrates the following:

- Each database is logically divided into one or more tablespaces.
- One or more datafiles are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace.
- The combined size of the datafiles in a tablespace is the total storage capacity of the tablespace. (The SYSTEM tablespace has 2-megabyte (Mb) storage capacity, and USERS tablespace has 4 MB).
- The combined storage capacity of a database's tablespaces is the total storage capacity of the database (6 Mb).

Physical Database Structures

The following sections explain the physical database structures of an Oracle database, including Datafiles, Control Files, Redo Log Files, Archive Log Files, Parameter Files, Alert and Trace Log Files and Backup Files

Datafiles

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

The characteristics of datafiles are:

- A datafile can be associated with only one database.
- Datafiles can have certain characteristics set to let them automatically extend when the database runs out of space.
- One or more datafiles form a logical unit of database storage called a tablespace, as discussed earlier in this chapter.
- Data in a datafile is read, as needed, during normal database operation and stored in the memory cache of Oracle. For example, assume that a user wants to access some data in a table of a database. If the requested information is not already in the memory

cache for the database, then it is read from the appropriate datafiles and stored in memory.

- Modified or new data is not necessarily written to a datafile immediately. To reduce the amount of disk access and to increase performance, data is pooled in memory and written to the appropriate datafiles all at once, as determined by the database writer process (DBWn) background process.

Redo Log Files

Every Oracle database has a set of two or more redo log files. The set of redo log files is collectively known as the redo log for the database. A redo log is made up of redo entries (also called redo records).

The primary function of the redo log is to record all changes made to data. If a failure prevents modified data from being permanently written to the datafiles, then the changes can be obtained from the redo log, so work is never lost.

The information in a redo log file is used only to recover the database from a system or media failure that prevents database data from being written to the datafiles. For example, if an unexpected power outage terminates database operation, then data in memory cannot be written to the datafiles, and the data is lost. However, lost data can be recovered when the database is opened, after power is restored. By applying the information in the most recent redo log files to the database datafiles, Oracle restores the database to the time at which the power failure occurred.

The process of applying the redo log during a recovery operation is called rolling forward.

Control Files

Every Oracle database has a control file. A control file contains entries that specify the physical structure of the database. For example, it contains the following information:

- Database name
- Names and locations of datafiles and redo log files
- Time stamp of database creation
- Use of Control Files

Every time an instance of an Oracle database is started, its control file identifies the database and redoes log files that must be opened for database operation to proceed. If the physical makeup of the database is altered (for example, if a new data file or redo log file is created), then the control file is automatically modified by Oracle to reflect the change. A control file is also used in database recovery.

Summary

- You can explain How Grid computing Architecture Works.
- Client/Server Architecture
- Physical Database Structures
- Logical Database Structures

Test Your Understanding

1. What is tablespace in Oracle?
2. When we create an oracle object. Which space unit is allocated for the objects?
3. What is the role of Control files in Oracle?
4. Where table get stored and when it is created by user?

Session 03: Schemas and Common Schema Objects

Learning Objectives

After completing this session, you will be able to:

- Identify Oracle Users and Schema
- Comprehend the Tables in Oracle
- Comprehend Views in Oracle
- Comprehend Materialized Views in Oracle
- Comprehend Dimensions in Oracle
- Comprehend the Sequence Generator in Oracle
- Identify the Indexes in Oracle
- Comprehend Synonyms in Oracle

Oracle Users and Schema

Each Oracle database has a list of user names. To access a database, a user must use a database User and Password. A schema is a collection of database objects. A schema is owned by a database user and has the same name as the user himself. Schema objects are the logical structures that directly refer to the database's data. Schema objects include structures like tables, views, and indexes.

```
CREATE SCHEMA AUTHORIZATION scott
CREATE TABLE new_product
(color VARCHAR2(10)  PRIMARY KEY, quantity NUMBER)
CREATE VIEW new_product_view
AS SELECT color, quantity FROM new_product WHERE color = 'RED'
GRANT select ON new_product_view TO hr1;
```

This statement does not actually create a schema. Oracle Database automatically creates a schema when you create a user. This populates your schema with tables and views. This also grants privileges on those objects without having issued multiple SQL statements in multiple transactions.

Schema Objects, Tablespaces, and Datafiles

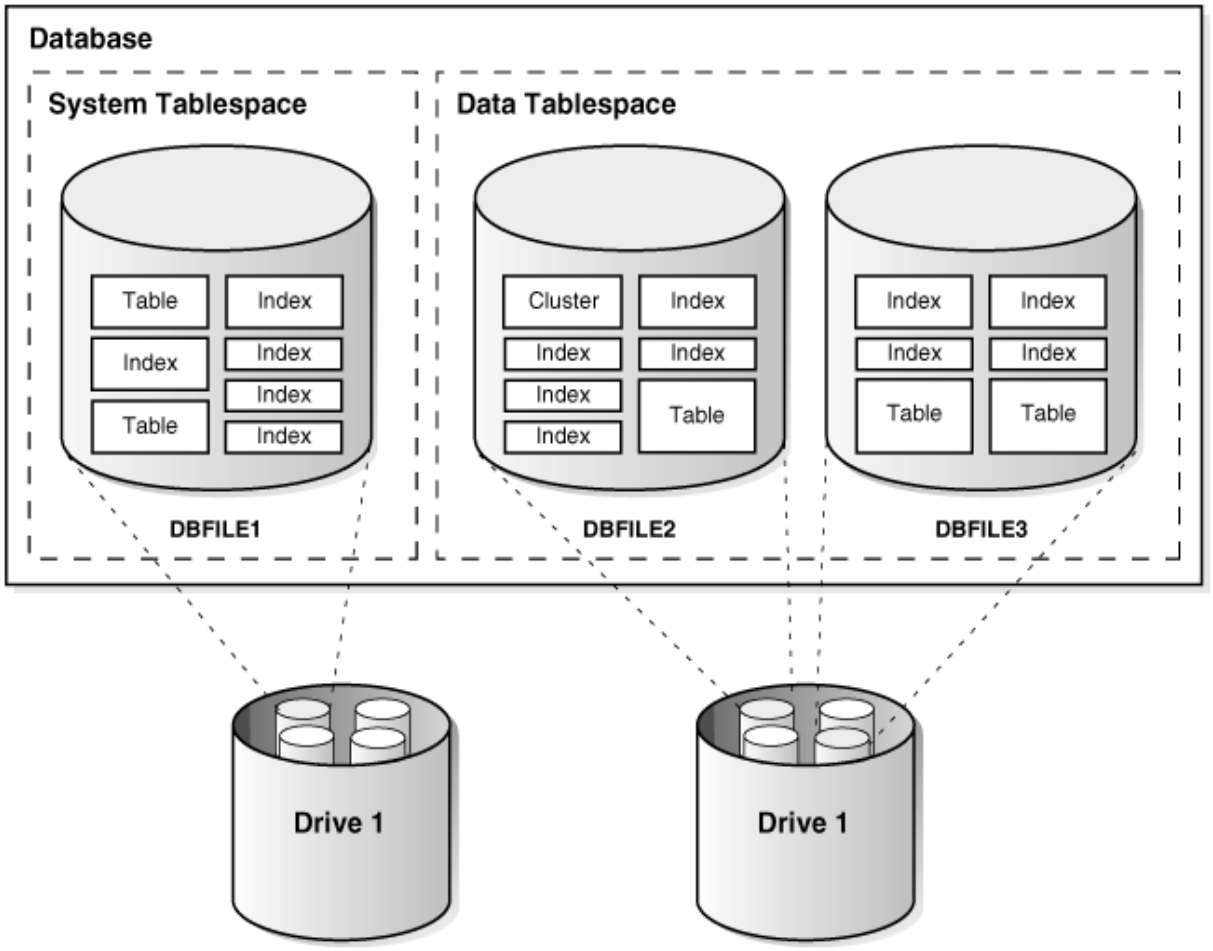


Figure: Relationship among Objects, Tablespaces and Datafiles

Overview of Tables

- 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 (such as employees) and set of columns. You give each column a column name (such as employee_id, last_name, and job_id), a datatype (such as VARCHAR2, DATE, or NUMBER), and a width. The width can be predetermined by the datatype, as in DATE. If columns are of the NUMBER datatype, define precision and scale instead of width. A row is a collection of column information corresponding to a single record.
- You can specify rules for each column of a table. These rules are called integrity constraints. One example is a NOT NULL integrity constraint. This constraint forces the column to contain a value in every row. You can also specify table columns for which data is encrypted before being stored in the datafile. Encryption prevents users from circumventing database access control mechanisms by looking inside datafiles directly with operating system tools.
- After you create a table, insert rows of data using SQL statements. Table data can then be queried, deleted, or updated using SQL.

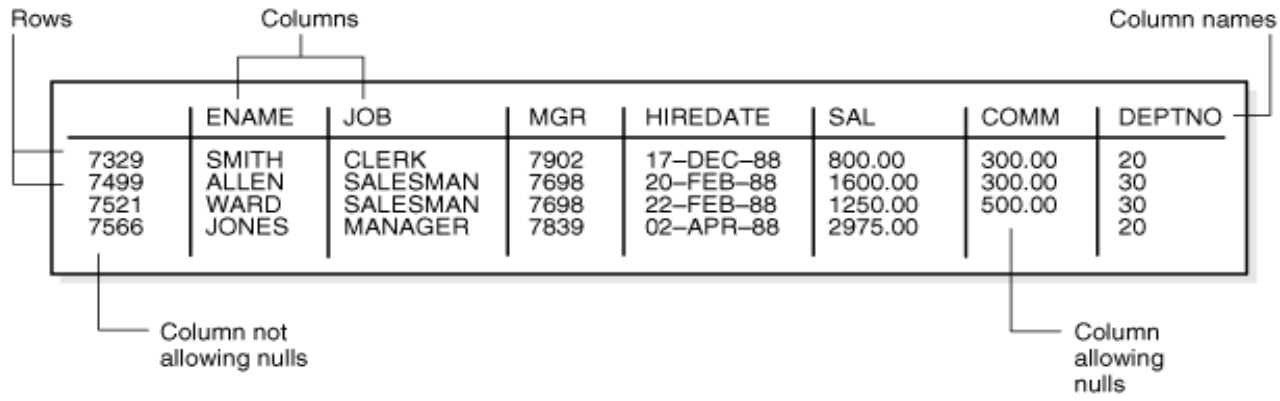


Figure: Table in Oracle

Overview of Views

A view is a tailored presentation of the data contained in one or more tables or other views. 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. 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 base tables that the view references.

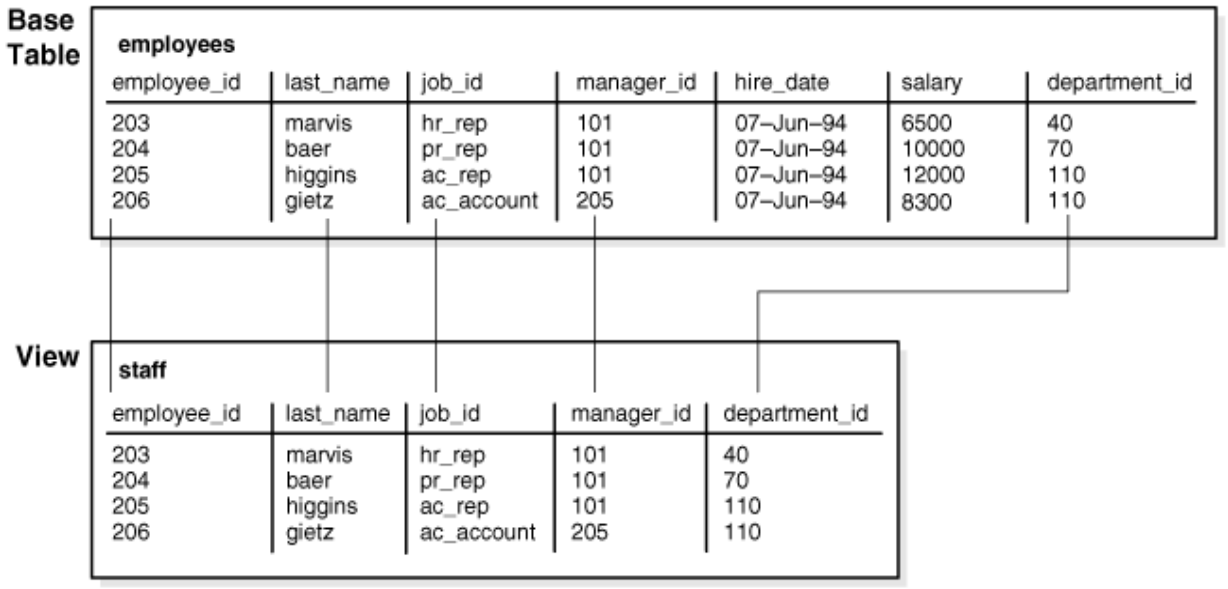


Figure: An Example of a View

Views provide a means to present a different representation of the data that resides within the base tables. Views are very powerful because they let you tailor the presentation of data to different types of users.

Views are often used to:

- Provide an additional level of table security by restricting access to a predetermined set of rows or columns of a table
- Hide data complexity
  - For example, a single view can be defined with a join, which is a collection of related columns or rows in multiple tables. However, the view hides the fact that this information actually originates from several tables.
- Simplify statements for the user
  - For example, views allow users to select information from multiple tables without actually knowing how to perform a join.
- Present the data in a different perspective from that of the base table
  - For example, the columns of a view can be renamed without affecting the tables on which the view is based.
- Isolate applications from changes in definitions of base tables
  - For example, if a view's defining query references three columns of a four column table, and a fifth column is added to the table, then the view's definition is not affected, and all applications using the view are not affected.
- Express a query that cannot be expressed without using a view
  - For example, a view can be defined that joins a GROUP BY view with a table, or a view can be defined that joins a UNION view with a table.
- Save complex queries
  - For example, a query can perform extensive calculations with table information. By saving this query as a view, you can perform the calculations each time the view is queried.

Overview of Materialized Views

Materialized views are schema objects that can be used to summarize, compute, replicate, and distribute data. They are suitable in various computing environments such as data warehousing, decision support, and distributed or mobile computing:

- In data warehouses, materialized views are used to compute and store aggregated data such as sums and averages. Materialized views in these environments are typically referred to as summaries because they store summarized data. They can also be used to compute joins with or without aggregations. If compatibility is set to Oracle9i or higher, then materialized views can be used for queries that include filter selections.
- The optimizer can use materialized views to improve query performance by automatically recognizing when a materialized view can and should be used to satisfy a request. The optimizer transparently rewrites the request to use the materialized view. Queries are then directed to the materialized view and not to the underlying detail tables or views.
- In distributed environments, materialized views are used to replicate data at distributed sites and synchronize updates done at several sites with conflict resolution methods. The materialized views as replicas provide local access to data that otherwise have to be accessed from remote sites.
- In mobile computing environments, materialized views are used to download a subset of data from central servers to mobile clients, with periodic refreshes from the central servers and propagation of updates by clients back to the central servers.

Materialized views are similar to indexes in several ways:

- They consume storage space.
- They must be refreshed when the data in their master tables changes.
- They improve the performance of SQL execution when they are used for query rewrites.
- Their existence is transparent to SQL applications and users.
  - Unlike indexes, materialized views can be accessed directly using a `SELECT` statement. Depending on the types of refresh that are required, they can also be accessed directly in an `INSERT`, `UPDATE`, or `DELETE` statement.
  - A materialized view can be partitioned. You can define a materialized view on a partitioned table and one or more indexes on the materialized view.

Overview of Dimensions

A dimension defines hierarchical (parent/child) relationships between pairs of columns or column sets. Each value at the child level is associated with one and only one value at the parent level. A hierarchical relationship is a functional dependency from one level of a hierarchy to the next level in the hierarchy.

A dimension is a container of logical relationships between columns, and it does not have any data storage assigned to it.

The columns in a dimension can come either from the same table (denormalized) or from multiple tables (fully or partially normalized). To define a dimension over columns from multiple tables, connect the tables using the `JOIN` clause of the `HIERARCHY` clause.

For example, a normalized time dimension can include a date table, a month table, and a year table, with join conditions that connect each date row to a month row, and each month row to a year row. In a fully denormalized time dimension, the date, month, and year columns are all in the same table. Whether normalized or denormalized, the hierarchical relationships among the columns need to be specified in the `CREATE DIMENSION` statement.

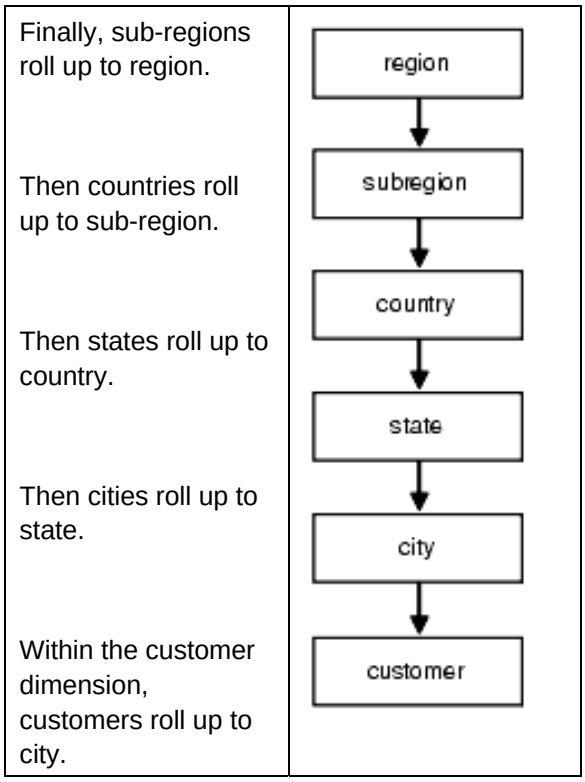


Figure: An example of dimension

Overview of the Sequence Generator

The sequence generator provides a sequential series of numbers. The sequence generator is especially useful in multiuser environments for generating unique sequential numbers without the overhead of disk I/O or transaction locking.

For example, assume two users are simultaneously inserting new employee rows into the employees table. By using a sequence to generate unique employee numbers for the employee_id column, neither user has to wait for the other to enter the next available employee number. The sequence automatically generates the correct values for each user.

Therefore, the sequence generator reduces serialization where the statements of two transactions must generate sequential numbers at the same time. By avoiding the serialization that results when multiple users wait for each other to generate and use a sequence number, the sequence generator improves transaction throughput, and user's wait is considerably shorter.

Sequence numbers are Oracle integers of up to 38 digits defined in the database. A sequence definition indicates general information, such as the following:

- The name of the sequence
- Whether the sequence ascends or descends
- The interval between numbers
- Whether Oracle should cache sets of generated sequence numbers in memory

Oracle stores the definitions of all sequences for a particular database as rows in a single data dictionary table in the SYSTEM tablespace. Therefore, all sequence definitions are always available, because the SYSTEM tablespace is always online.

Overview of 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 the index in this 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.

You can create many indexes for a table as long as the combination of columns differs for each index. You can create more than one index using the same columns if you specify distinctly different combinations of the columns. For example, the following statements specify valid combinations:

```
CREATE INDEX employees_idx1 ON employees (last_name, job_id);

CREATE INDEX employees_idx2 ON employees (job_id, last_name);
```

Oracle provides several indexing schemes, which provide complementary performance functionality:

- B-tree indexes
- B-tree cluster indexes
- Hash cluster indexes
- Reverse key indexes
- Bitmap indexes
- Bitmap join 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. An index is merely a fast access path to the data.

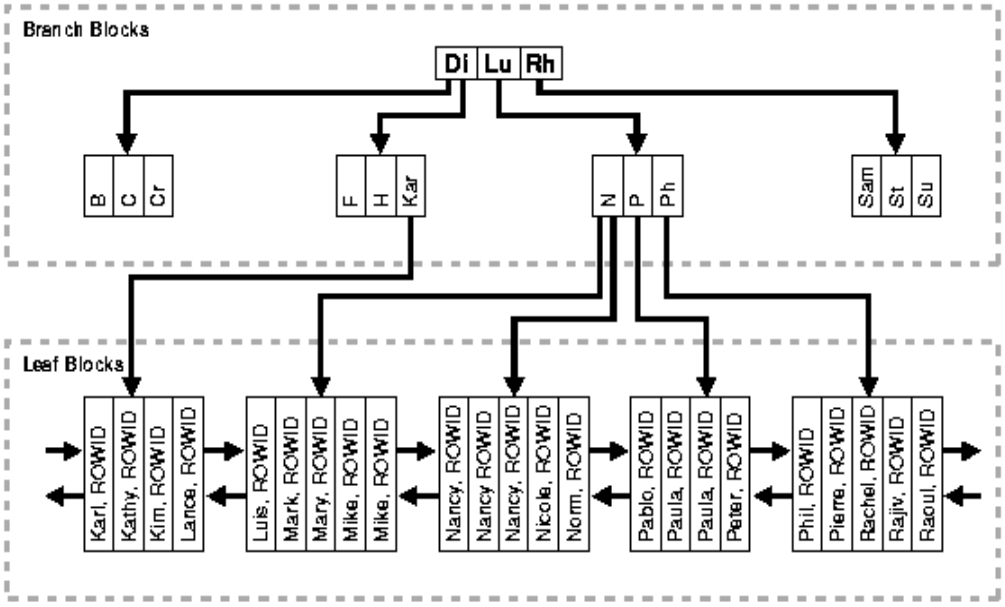


Figure: Internal structure of a B-tree Index

Overview of Synonyms

A synonym is an alias for any table, view, materialized view, sequence, procedure, function, package, type, Java class schema object, user-defined object type, or another synonym. Because a synonym is simply an alias, it requires no storage other than its definition in the data dictionary.

Synonyms are often used for security and convenience. For example, they can do the following:

- Mask the name and owner of an object
- Provide location transparency for remote objects of a distributed database
- Simplify SQL statements for database users
- Enable restricted access similar to specialized views when exercising fine-grained access control

You can create both public and private synonyms. A public synonym is owned by the special user group named PUBLIC and every user in a database can access it. A private synonym is in the schema of a specific user who has control over its availability to others.

Synonyms are very useful in both distributed and non-distributed database environments because they hide the identity of the underlying object, including its location in a distributed system. This is advantageous because if the underlying object must be renamed or moved, then only the synonym needs to be redefined.

Applications based on the synonym continue to function without modification. Synonyms can also simplify SQL statements for users in a distributed database system.

The following example shows how and why public synonyms are often created by a database administrator to hide the identity of a base table and reduce the complexity of SQL statements. Assume the following:

- A table called SALES_DATA is in the schema owned by the user JWARD.
- The SELECT privilege for the SALES_DATA table is granted to PUBLIC.

At this point, you have to query the table SALES_DATA with a SQL statement similar to the following:

```
SELECT * FROM jward.sales_data;
```

Notice how you must include both the schema that contains the table along with the table name to perform the query.

Assume that the database administrator creates a public synonym with the following

```
CREATE PUBLIC SYNONYM sales FOR jward.sales_data;
```

After the public synonym is created, you can query the table SALES_DATA with a simple SQL statement:

```
SELECT * FROM sales;
```

Notice that the public synonym SALES hides the name of the table SALES_DATA and the name of the schema that contains the table.

Summary

- Each Oracle database has a list of user names. To access a database, a user must use a database User and Password.
- Tables are the basic unit of data storage in an Oracle database.
- A view is a tailored presentation of the data contained in one or more tables or other views.
- Materialized views are schema objects that can be used to summarize, compute, replicate, and distribute data.
- A dimension defines hierarchical (parent/child) relationships between pairs of columns or column sets.
- The sequence generator provides a sequential series of numbers.
- Indexes are optional structures associated with tables and clusters.

Test your understanding

1. What is the difference between User and Schema in Oracle?
2. How do you create a Schema?
3. What is the use of Index?
4. State the difference between Public and Private Synonym.
5. What is use of view?

Session 04: Oracle Instance

Learning Objectives

After completing this session, you will be able to:

- Comprehend the Instance Memory Structures
- Comprehend the System Global Area (SGA)
- Comprehend the database buffers
- Comprehend the redo log buffer
- Define the shared pool
- Describe the Program Global Area (PGA)
- Identify the Dedicated and Shared Servers.

Instance Memory Structures

An Oracle database server consists of an Oracle database and an Oracle instance. Every time a database is started, a system global area (SGA) is allocated and Oracle background processes are started. The combination of the background processes and memory buffers is called an Oracle instance.

Real Application Clusters: Multiple Instance Systems

Some hardware architectures (for example, shared disk systems) enable multiple computers to share access to data, software, or peripheral devices. Real Application Clusters (RAC) takes advantage of such architecture by running multiple instances that share a single physical database. In most applications, RAC enables access to a single database by users on multiple computers with increased performance.

An Oracle database server uses memory structures and processes to manage and access the database. All memory structures exist in the main memory of the computers that constitute the database system. Processes are jobs that work in the memory of these computers.

Two basic memory structures are associated with Oracle, the system global area (SGA), and the program global area (PGA).

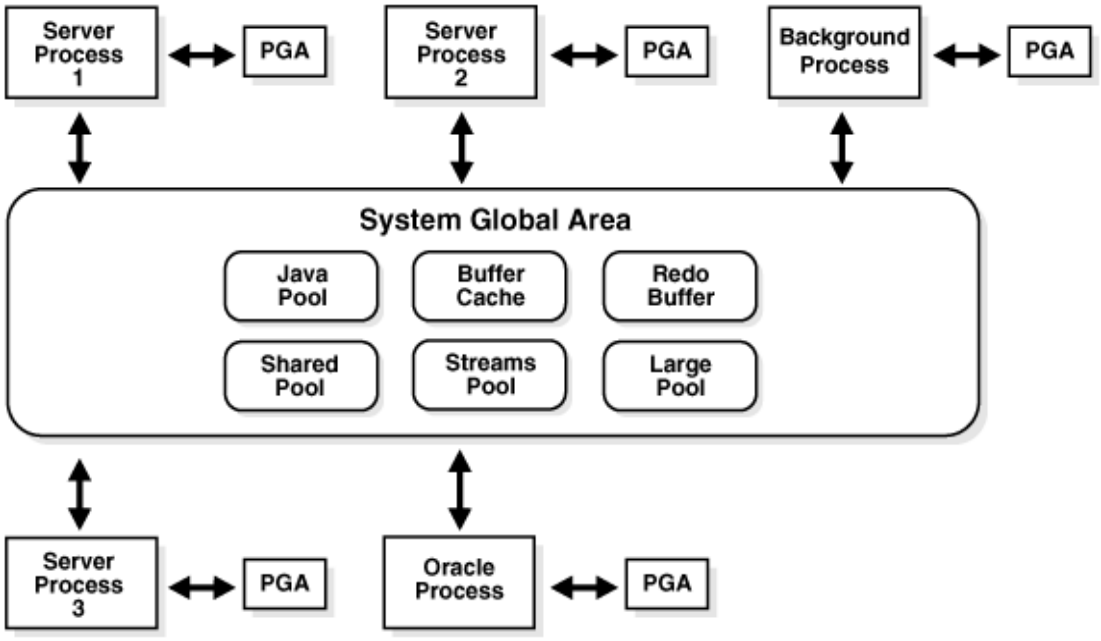


Figure: Oracle Memory Structures

System Global Area (SGA)

The System Global Area (SGA) is a shared memory region that contains data and control information for one Oracle instance. Oracle allocates the SGA when an instance starts and deallocates it when the instance shuts down. Each instance has its own SGA.

Users currently connected to an Oracle database share the data in the SGA. For optimal performance, the entire SGA should be as large as possible (while still fitting in real memory) to store as much data in memory as possible and to minimize disk I/O.

The information stored in the SGA is divided into several types of memory structures, including the database buffer; redo log buffer, and the shared pool.

The SGA contains the following data structures:

- Database buffer cache
- Redo log buffer
- Shared pool
- Java pool
- Large pool (optional)
- Streams pool
- Data dictionary cache
- Other miscellaneous information

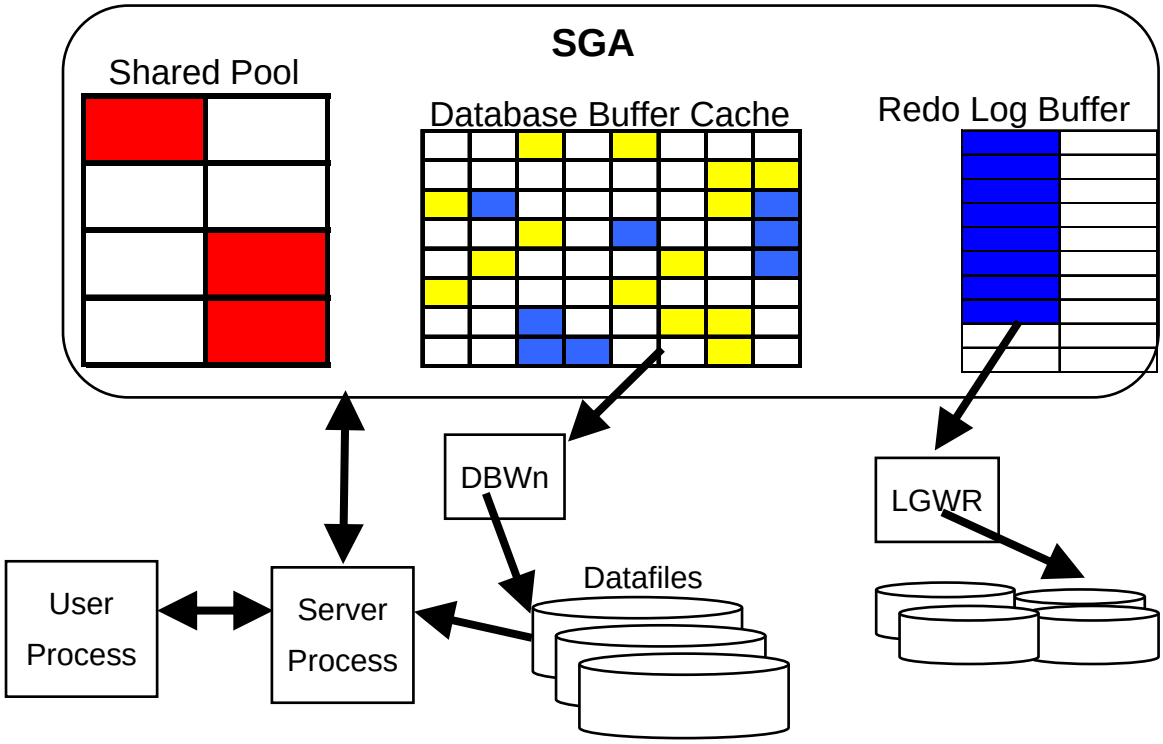


Figure: DML Changes and Redo Log Buffer uses

Database Buffers

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

Redo Log Buffer

Redo Log Buffer of the SGA The redo log buffer stores redo entries—a log of changes made to the database. The redo entries stored in the redo log buffers are written to an online redo log, which is used if database recovery is necessary. The size of the redo log is static.

Shared Pool

Shared Pool of the SGA The shared pool contains shared memory constructs, such as shared SQL areas. A shared SQL area is required to process every unique SQL statement submitted to a database. A shared SQL area contains information such as the parse tree and execution plan for the corresponding statement. A single shared SQL area is used by multiple applications that issue the same statement, leaving more shared memory for other uses.

Program Global Area (PGA)

A program global area (PGA) is a memory region that contains data and control information for a server process. It is a non-shared memory created by Oracle when a server process is started. Access to it is exclusive to that server process and is read and written only by Oracle code acting on behalf of it. The total PGA memory allocated by each server process attached to an Oracle instance is also referred to as the aggregated PGA memory allocated by the instance.

Content of the PGA:

The content of the PGA memory varies, depending on whether the instance is running the shared server option. But generally speaking, the PGA memory can be classified in the following manner.

- **Private SQL Area:**
  - A private SQL area contains data such as bind information and runtime memory structures.
  - The location of a private SQL area depends on the type of connection established for a session. If a session is connected through a dedicated server, private SQL areas are located in the server process's PGA. However, if a session is connected through a shared server, part of the private SQL area is kept in the SGA.
- **Cursors and SQL Areas:**
  - Session memory is the memory allocated to hold a session's variables (logon information) and other information related to the session. For a shared server, the session memory is shared and not private.
- **SQL Work Areas:**
  - For complex queries (for example, decision-support queries), a big portion of the runtime area is dedicated to work areas allocated by memory-intensive operators.

Dedicated and Shared Servers

Server Configurations:

- Dedicated server process
- Shared server process

Dedicated Server Process: The user and server processes are separate, distinct processes. The separate server process created on behalf of each user process is called a dedicated server process (or shadow process), because this server process acts only on behalf of the associated user process.

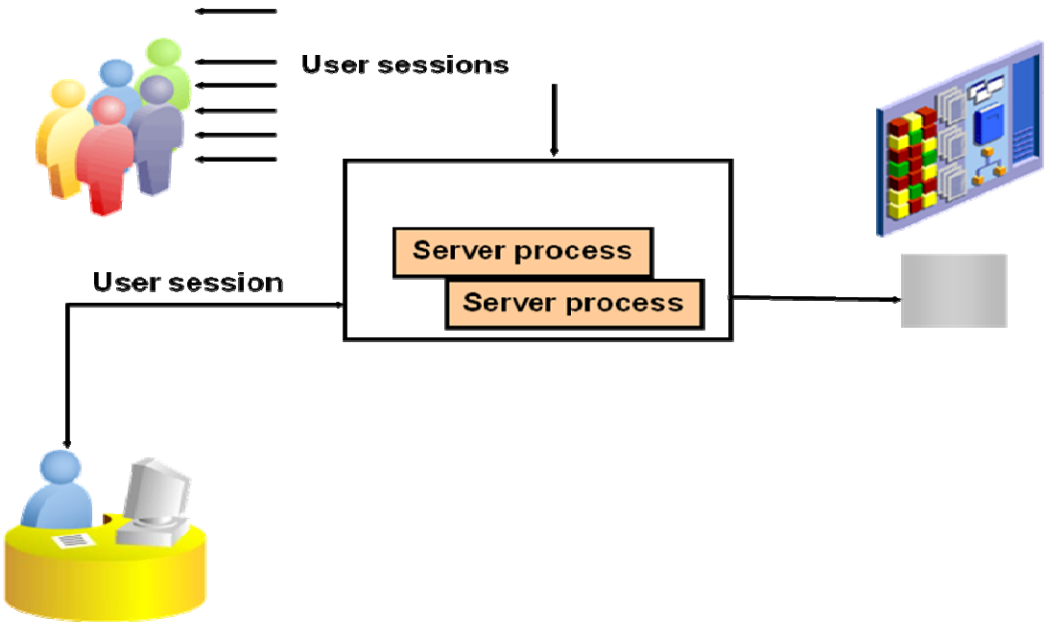


Figure: Dedicated Server Architecture

Shared Server Architecture: Shared server architecture eliminates the need for a dedicated server process for each connection. A dispatcher directs multiple incoming network session requests to a pool of shared server processes. An idle shared server process from a shared pool of server processes picks up a request from a common queue, which means a small number of shared servers can perform the same amount of processing as many dedicated servers

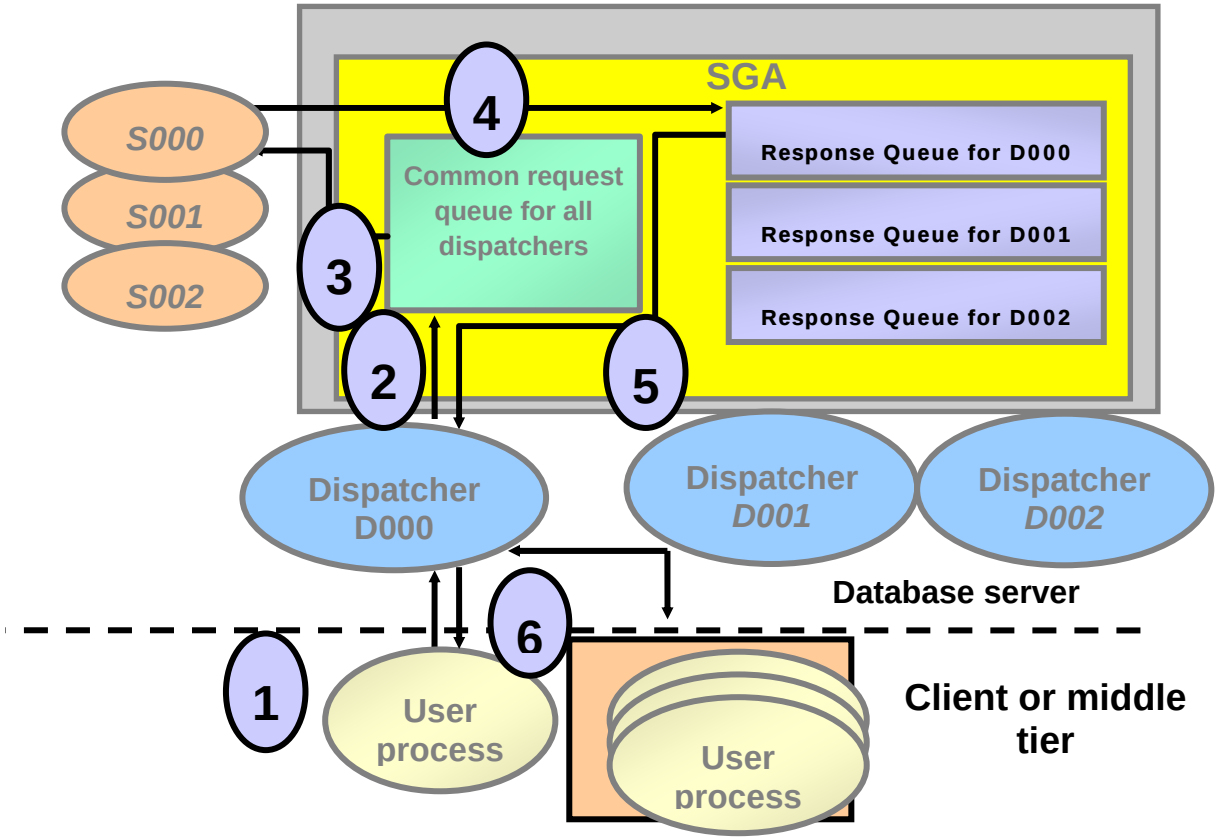


Figure: Shared Server Architecture

Summary

- When Oracle database starts, SGA comes into picture in Memory.
- Two main parts of the Instance Memory, SGA and PGA
- Three main areas of SGA; Database Buffers, Redo Log Buffer, Shared Pool
- A program global area (PGA) is a memory region that contains data and control information for a server process.
- You can use one of the two architectures; Dedicated and Shared Servers

Test your Understanding

1. What is the difference between Database and Instance?
2. What are the different components of SGA?
3. What is PGA and it's Uses?
4. What is difference between Dedicated and Shared Servers?

Session 05: Oracle Processes

Learning Objectives

After completing this session, you will be able to:

- Comprehend server processes
- Comprehend background processes
- Comprehend shared server architecture

Server Processes

Oracle creates server processes to handle the requests of user processes, connected to the instance. In some situations, when the application and Oracle operate on the same computer, it is possible to combine the user process and corresponding server process into a single process to reduce system overhead. However, when the application and Oracle operate on different computers, a user process always communicates with Oracle through a separate server process.

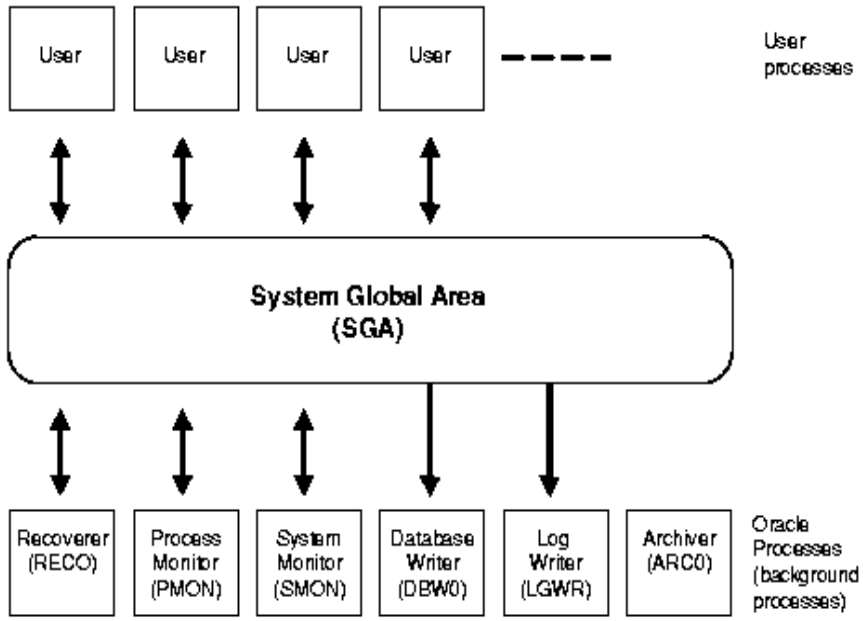


Figure: Oracle Process Architecture and User Processes

Server processes created on behalf of each user's application can perform one or more of the following:

- Parse and run SQL statements issued through the application.
- Read necessary data blocks from datafiles on disk into the shared database buffers of the SGA, if the blocks are not already present in the SGA.

Return results in such a way that the application can process the information.

Background Processes

To maximize performance and accommodate many users, a multi-process Oracle system uses some additional Oracle processes called background processes.

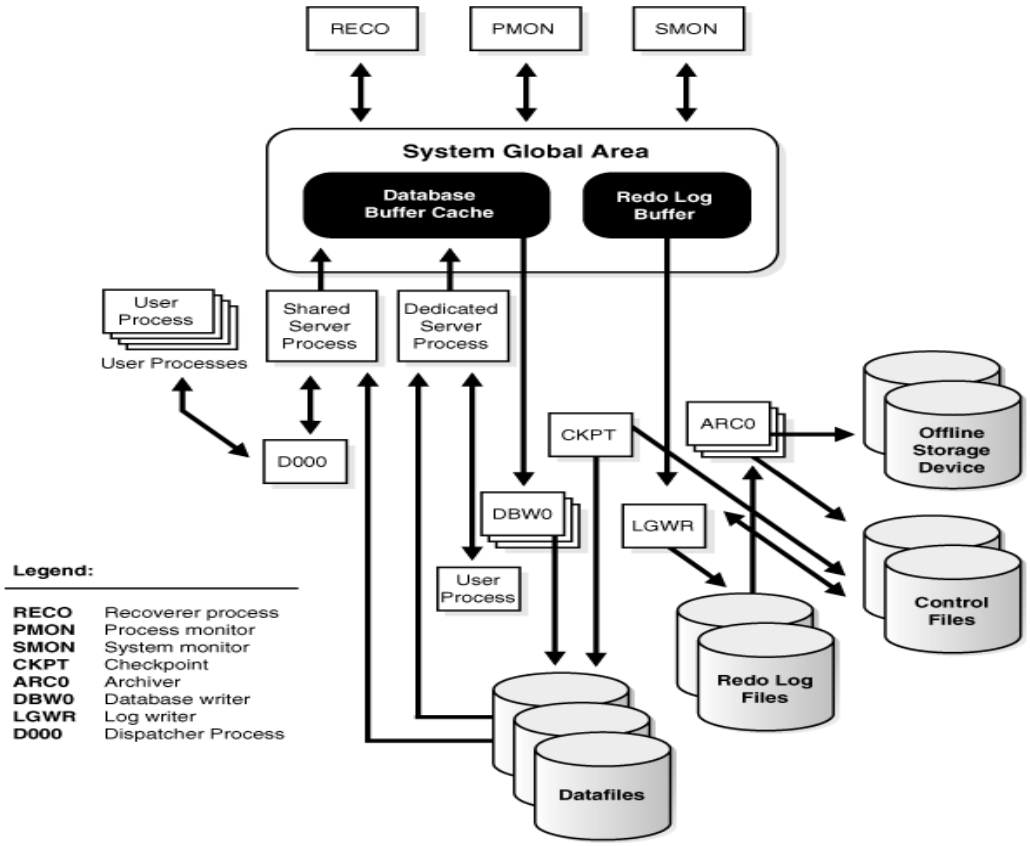


Figure: Oracle Background Processes and their functions

The background processes in an Oracle instance can include the following:

- Database Writer Process (DBWn)
- Log Writer Process (LGWR)
- Checkpoint Process (CKPT)
- System Monitor Process (SMON)
- Process Monitor Process (PMON)
- Re-coverer Process (RECO)
- Job Queue Processes
- Achiever Processes (ARCn)
- Queue Monitor Processes (QMNn)

Database Writer Process (DBWn)

The database writer process (DBWn) writes the contents of buffers to datafiles. The DBWn processes are responsible for writing modified (dirty) buffers in the database buffer cache to disk. Although one database writer process (DBW0) is adequate for most systems, you can configure additional processes (DBW1 through DBW9 and DBWa through DBWj) to improve write performance if your system modifies data heavily. These additional DBWn processes are not useful on uniprocessor systems.

When a buffer in the database buffer cache is modified, it is marked dirty. A cold buffer is a buffer that has not been recently used according to the least recently used (LRU) algorithm. The DBWn process writes cold, dirty buffers to disk so that user processes are able to find cold, clean buffers that can be used to read new blocks into the cache. As buffers are dirtied by user processes, the number of free buffers diminishes. If the number of free buffers drops too low, user processes that must read blocks from disk into the cache are not able to find free buffers. DBWn manages the buffer cache so that user processes can always find free buffers.

The DBWn process writes dirty buffers to disk under the following conditions:

- When a server process cannot find a clean reusable buffer after scanning a threshold number of buffers, it signals DBWn to write. DBWn writes dirty buffers to disk asynchronously while performing other processing.
- DBWn periodically writes buffers to advance the checkpoint, which is the position in the redo thread (log) from which instance recovery begins. This log position is determined by the oldest dirty buffer in the buffer cache.

Log Writer Process (LGWR)

The log writer process (LGWR) is responsible for redo log buffer management—writing the redo log buffer to a redo log file on disk. LGWR writes all redo entries that have been copied into the buffer since the last time it wrote.

The redo log buffer is a circular buffer. When LGWR writes redo entries from the redo log buffer to a redo log file, server processes can then copy new entries over the entries in the redo log buffer that have been written to disk. LGWR normally writes fast enough to ensure that space is always available in the buffer for new entries, even when access to the redo log is heavy.

LGWR writes one contiguous portion of the buffer to disk. LGWR writes:

- A commit record when a user process commits a transaction
- Redo log buffers
  - Every three seconds
  - When the redo log buffer is one-third full
  - When a DBWn process writes modified buffers to disk, if necessary

When a user commits a transaction, the transaction is assigned a system change number (SCN), which Oracle records along with the transaction's redo entries in the redo log. SCNs are recorded in the redo log so that recovery operations can besynchronized in Real Application Clusters and distributed databases

Checkpoint Process (CKPT)

When a checkpoint occurs, Oracle must update the headers of all datafiles to record the details of the checkpoint. This is done by the CKPT process. The CKPT process does not write blocks to disk; DBWn always performs that work.

The statistic DBWR checkpoints displayed by the System_Statistics monitor in Enterprise Manager indicates the number of checkpoint requests completed.

System Monitor Process (SMON)

The system monitor process (SMON) performs recovery, if necessary, at instance startup. SMON is also responsible for cleaning up temporary segments that are no longer in use and for coalescing contiguous free extents within dictionary managed tablespaces. If any terminated transactions were skipped during instance recovery because of file-read or offline errors, SMON recovers them when the tablespace or file is brought back online. SMON checks regularly to see whether it is needed. Other processes can call SMON if they detect a need for it.

With Real Application Clusters, the SMON process of one instance can perform instance recovery for a failed CPU or instance.

Process Monitor Process (PMON)

The process monitor (PMON) performs process recovery when a user process fails.

PMON is responsible for cleaning up the database buffer cache and freeing resources that the user process was using. For example, it resets the status of the active transaction table, releases locks, and removes the process ID from the list of active processes.

PMON periodically checks the status of dispatcher and server processes, and restarts any that have stopped running (but not any that Oracle has terminated intentionally). PMON also registers information about the instance and dispatcher processes with the network listener.

Like SMON, PMON checks regularly to see whether it is needed and can be called if another process detects the need for it.

Recoverer Process (RECO)

The recoverer process (RECO) is a background process used with the distributed database configuration that automatically resolves failures involving distributed transactions. The RECO process of a node automatically connects to other databases involved in an in-doubt distributed transaction. When the RECO process reestablishes a connection between involved database servers, it automatically resolves all in-doubt transactions, removing from each database's pending transaction table any rows that correspond to the resolved in-doubt transactions.

If the RECO process fails to connect with a remote server, RECO automatically tries to connect again after a timed interval. However, RECO waits an increasing amount of time (growing exponentially) before it attempts another connection. The RECO process is present only if the instance permits distributed transactions. The number of concurrent distributed transactions is not limited.

Job Queue Processes

Job queue processes are used for batch processing. They run user jobs. They can be viewed as a scheduler service that can be used to schedule jobs as PL/SQL statements or procedures on an Oracle instance. Given a start date and an interval, the job queue processes try to run the job at the next occurrence of the interval.

Job queue processes are managed dynamically. This allows job queue clients to use more job queue processes when required. The resources used by the new processes are released when they are idle.

Dynamic job queue processes can run a large number of jobs concurrently at a given interval.

Archiver Processes (ARCn)

The archiver process (ARCn) copies redo log files to a designated storage device after a log switch has occurred. ARCn processes are present only when the database is in ARCHIVELOG mode, and automatic archiving is enabled.

An Oracle instance can have up to 10 ARCn processes (ARC0 to ARC9). The LGWR process starts a new ARCn process whenever the current number of ARCn processes is insufficient to handle the workload. The alert log keeps a record of when LGWR starts a new ARCn process.

If you anticipate a heavy workload for archiving, such as during bulk loading of data, you can specify multiple archiver processes with the initialization parameter `LOG_ARCHIVE_MAX_PROCESSES`. The `ALTER SYSTEM` statement can change the value of this parameter dynamically to increase or decrease the number of ARCn processes. However, you do not need to change this parameter from its default value of 1, because the system determines how many ARCn processes are needed, and LGWR automatically starts up more ARCn processes when the database workload requires more.

Shared Server Architecture

Shared server architecture eliminates the need for a dedicated server process for each connection. A dispatcher directs multiple incoming network session requests to a pool of shared server processes. An idle shared server process from a shared pool of server processes picks up a request from a common queue, which means a small number of shared servers can perform the same amount of processing as many dedicated servers. It is because of the amount of memory required for each user is relatively small, less memory and process management are required, and more users can be supported.

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.

Some user processes cannot communicate with the dispatcher, so the network listener process cannot connect them to a dispatcher. In this case, or if the user process requests a dedicated server, the listener creates a dedicated server and establishes an appropriate connection.

Oracle's shared server architecture increases the scalability of applications and the number of clients simultaneously connected to the database. It can enable existing applications to scale up without making any changes to the application itself.

Summary

- Oracle creates server processes to handle the requests of user processes, connected to the instance.
- The database writer process (DBWn) writes the contents of buffers to data files.
- The log writer process (LGWR) is responsible for writing the redo log buffer to a redo log file on disk.
- The system monitor process (SMON) performs recovery, if necessary, at instance startup.
- The process monitor (PMON) performs process recovery when a user process fails.

Test Your Understanding

1. What is function for the Server Processes?
2. Which process write data from data buffer cache to Data files?
3. Which process is responsible for cleanup and process recovery?
4. Which server architecture is better when number of users is very high?
5. Which process reads the data from data files to Data buffer cache?

Session 06: Scalability and Performance Features

Learning Objectives

After completing this session, you will be able to:

- Comprehend Concurrency
- Comprehend Read Consistency
- Comprehend Locking Mechanisms
- Comprehend Portability

Concurrency

A primary concern of a multiuser database management system is how to control concurrency, which is the simultaneous access of the same data by many users. Without adequate concurrency controls, data could be updated or changed improperly, compromising data integrity.

One way to manage data concurrency is to make each user wait for a turn. The goal of a database management system is to reduce that wait so it is either nonexistent or negligible to each user. All data manipulation language statements should proceed with as little interference as possible, and destructive interactions between concurrent transactions must be prevented. Destructive interaction is any interaction that incorrectly updates data or incorrectly alters underlying data structures. Neither performance nor data integrity can be sacrificed.

Oracle resolves such issues by using various types of locks and a multi-version consistency model. These features are based on the concept of a transaction. It is the application designer's responsibility to ensure that transactions fully exploit these concurrency and consistency features.

Maximizes concurrency:

- Updates are never blocked by queries.
- Queries are never blocked by updates or other queries.

During an insert, update or delete, undo information is written to an undo segment:

- Allows the user to roll back the transaction if necessary instead of committing.
- Also enables Oracle to reconstruct an image of the data in the database looked like at a time in the past.

Read Consistency

Read consistency, as supported by Oracle, does the following:

- Guarantees that the set of data seen by a statement is consistent with respect to a single point in time and does not change during statement execution (statement-level read consistency)
- Ensures that readers of database data do not wait for writers or other readers of the same data

- Ensures that writers of database data do not wait for readers of the same data
- Ensures that writers only wait for other writers if they attempt to update identical rows in concurrent transactions

The simplest way to think of Oracle's implementation of read consistency is to imagine each user operating a private copy of the database, hence the multi-version consistency model.

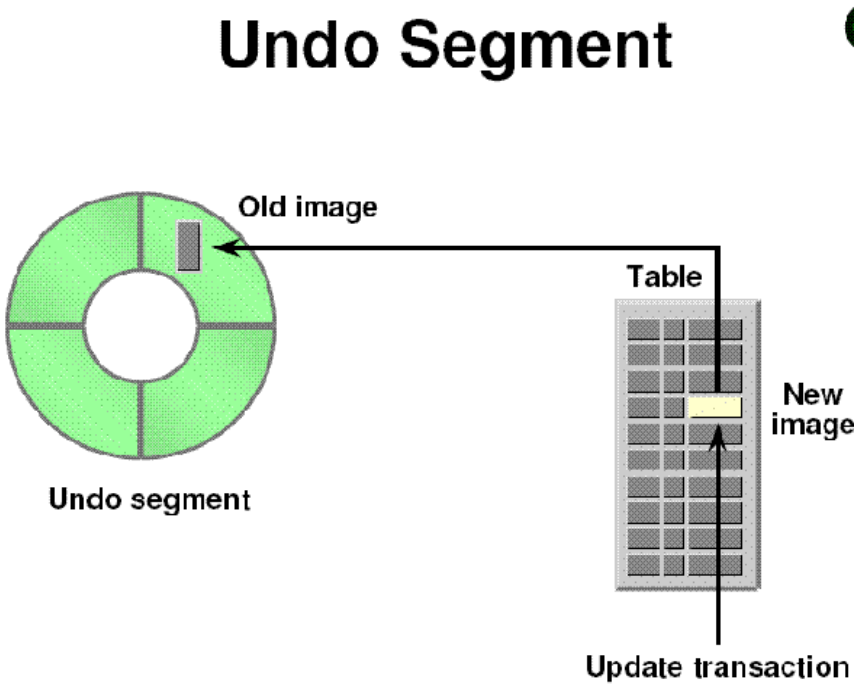


Figure: Undo Management in Oracle

To manage the multi-version consistency model, Oracle must create a read-consistent set of data when a table is queried (read) and simultaneously updated (written). When an update occurs, the original data values changed by the update are recorded in the database undo records. As long as this update remains part of an uncommitted transaction, any user that later queries the modified data views the original data values. Oracle uses current information in the system global area and information in the undo records to construct a read-consistent view of a table's data for a query.

Only when a transaction is committed are the changes of the transaction made permanent. Statements that start after the user's transaction is committed only see the changes made by the committed transaction.

Locking Mechanisms

Oracle also uses locks to control concurrent access to data. When updating information, the data server holds that information with a lock until the update is submitted or committed. Until that happens, no one else can make changes to the locked information. This ensures the data integrity of the system.

Oracle provides unique non-escalating row-level locking. Unlike other data servers that "escalate" locks to cover entire groups of rows or even the entire table, Oracle always locks only the row of information being updated. Because Oracle includes the locking information with the actual rows themselves, Oracle can lock an unlimited number of rows so users can work concurrently without unnecessary delays.

Automatic Locking:

- Oracle's lock manager automatically locks the table data at the row level.
- By locking table data at the row level, contention for the same data is minimized.

Manual Locking:

- Under some circumstances, a user might want to override default locking.
- Oracle allows manual override of automatic locking features at both the row level and the table level.

Portability

Oracle provides unique portability across all major platforms and ensures that your applications run without modification after changing platforms. This is because the Oracle code base is identical across platforms, so you have identical feature functionality across all platforms, for complete application transparency. Because of this portability, you can easily upgrade to a more powerful server as your requirements change.

Summary

- Concurrency is the simultaneous access of the same data by many users.
- Read Consistency ensures that readers of database data do not wait for writers or other readers of the same data.
- Oracle uses locks to control concurrent access to data.
- Oracle provides unique portability across all major platforms.

Test your Understanding

1. What is Concurrency and why it is needed?
2. How Oracle implements Read Consistency?
3. What is the default level of locking?
4. Can you read the table which is getting updated?

Session 07: Database Security

Learning Objectives

After completing this session, you will be able to:

- Comprehend database users and schemas
- Comprehend storage settings and quotas
- Comprehend authentication, authorization, and auditing
- Identify authentication methods

Introduction to Database Security

Database security entails allowing or disallowing user actions on the database and the objects within it. Oracle uses schemas and security domains to control access to data and to restrict the use of various database resources.

Oracle provides comprehensive discretionary access control. Discretionary access control regulates all user access to named objects through privileges. A privilege is permission to access a named object in a prescribed manner; for example, permission to query a table. Privileges are granted to users at the discretion of other users.

Database Users and Schemas

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

Security Domain:

Each user has a security domain, a set of properties that determine such things as:

- The actions (privileges and roles) available to the user
- The tablespace quotas (available disk space) for the user
- The system resource limits (for example, CPU processing time) for the user

Each property that contributes to a user's security domain is discussed in the following sections.

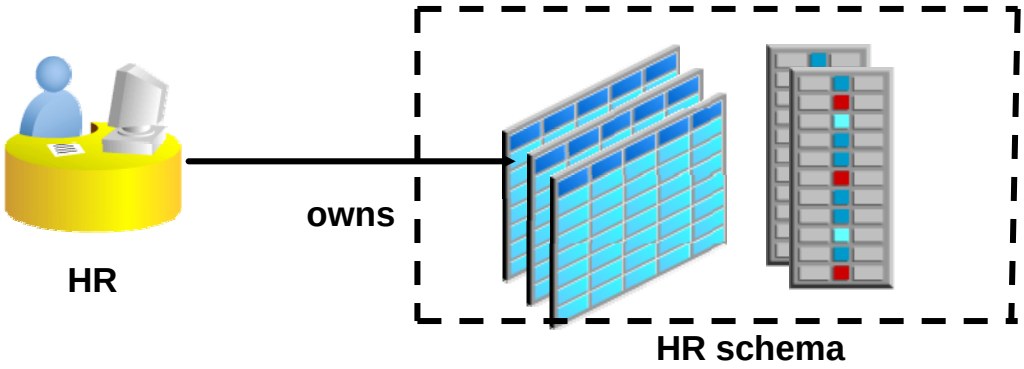


Figure: Relationship between User and Schema

Database Users, Roles, and Privileges

A privilege is a right to run a particular type of SQL statement or to access another user's object.

Grant privileges to users so that they can accomplish tasks required for their job. Grant privileges only to users who absolutely require them. Excessive granting of unnecessary privileges can compromise security. A user can receive a privilege in two different ways:

- You can grant privileges to users explicitly. For example, you can explicitly grant the privilege to insert records into the employees table to the user SCOTT.
- You can grant privileges to a role (a named group of privileges), and then grant the role to one or more users. For example, you can grant the privileges to select, insert, update, and delete records from the employees table to the role named clerk, which in turn you can grant to the users scott and brian.

As roles allow for easier and better management of privileges, you should generally grant privileges to roles and not to specific users.

There are two distinct categories of privileges:

- System Privileges
- Schema Object Privileges

System Privileges: A system privilege is the right to perform a particular action, or to perform an action on any schema objects of a particular type. For example, the privileges to create tablespaces and to delete the rows of any table in a database are system privileges.

There are over 100 distinct system privileges.

Schema Object Privileges: A schema object privilege is a privilege or right to perform a particular action on a specific schema object:

Different object privileges are available for different types of schema objects. For example, the privilege to delete rows from the departments table is an object privilege.

Granting object privileges on a table, view, sequence, procedure, function, or package to a synonym for the object has the same effect as if no synonym were used. When a synonym is dropped, all grants for the underlying schema object remain in effect, even if the privileges were granted by specifying the dropped synonym.

Roles:

Managing and controlling privileges is made easier by using roles, which are named groups of related privileges that you grant, as a group, to users or other roles. Within a database, each role name must be unique, different from all user names and all other role names. Unlike schema objects, roles are not contained in any schema. Therefore, a user who creates a role can be dropped with no effect on the role.

Roles ease the administration of end-user system and schema object privileges. However, roles are not meant to be used by application developers, because the privileges to access schema objects within stored programmatic constructs must be granted directly.

Oracle provides easy and controlled privilege management through roles.

Roles are named groups of related privileges that you grant to users or other roles.

- Easier privilege management
- Dynamic privilege management
- Selective availability of privileges
- Can be granted through the operating system

Storage Settings and Quotas

You can direct and limit the use of disk space allocated to the database for each user, including default and temporary tablespaces and tablespace quotas.

Default Tablespace: Each user is associated with a default tablespace. When a user creates a table, index, or cluster and no tablespace is specified to physically contain the schema object, the user's default tablespace is used if the user has the privilege to create the schema object and a quota in the specified default tablespace. The default tablespace provides Oracle with information to direct space use in situations where schema objects's location is not specified.

Temporary Tablespace: Each user has a temporary tablespace. When a user runs a SQL statement that requires the creation of temporary segments (such as the creation of an index), the user's temporary tablespace is used. By directing all users' temporary segments to a separate tablespace, the temporary tablespace can reduce I/O contention among temporary segments and other types of segments.

Tablespace Quotas: Oracle can limit the collective amount of disk space available to the objects in a schema. Quotas (space limits) can be set for each tablespace available to a user. This permits selective control over the amount of disk space that can be consumed by the objects of specific schemas.

Profiles and Resource Limits: Each user is assigned a profile that specifies limitations on several system resources available to the user, including the following:

- Number of concurrent sessions the user can establish
- CPU processing time available for the user's session and a single call to Oracle made by a SQL statement
- Amount of logical I/O available for the user's session and a single call to Oracle made by a SQL statement
- Amount of idle time available for the user's session
- Amount of connect time available for the user's session
- Password restrictions:
  - Account locking after multiple unsuccessful login attempts
  - Password expiration and grace period
  - Password reuse and complexity restrictions

Authentication, Authorization, Auditing

The Oracle database provides security in the form of authentication, authorization, and auditing.

Authentication ensures that only legitimate users gain access to the system.

Authorization ensures that those users only have access to resources they are permitted to access.

Auditing ensures accountability when users access protected resources.
Although these security mechanisms effectively protect data in the database, they do not prevent access to the operating system files where the data is stored.

Authentication means verifying the identity of a user, who wants to use data, resources, or applications.

You can refer to any combination of the methods described in the following sections:

- Authentication by the Operating System
- Authentication by the Network
- Authentication by the Oracle Database
- Multitier Authentication and Authorization
- Authentication by the Secure Socket Layer Protocol
- Authentication of Database Administrators

Summary

- To access a database, a user must use a database application and attempt a connection with a valid user name of the database. Each user name has an associated password to prevent unauthorized use.
- Roles are named groups of related privileges that you grant to users or other roles.
- You can direct and limit the use of disk space allocated to the database for each user, including default and temporary tablespaces and tablespace quotas.
- The Oracle database provides security in the form of authentication, authorization, and auditing.

Test Your Understanding

1. State the difference between User and Schema.
2. How can you create a user without create objects privileges?
3. What is the use of Temporary Tablespace?
4. What are the benefits of Oracle Roles?

Session 08: Oracle Utilities

Learning Objectives

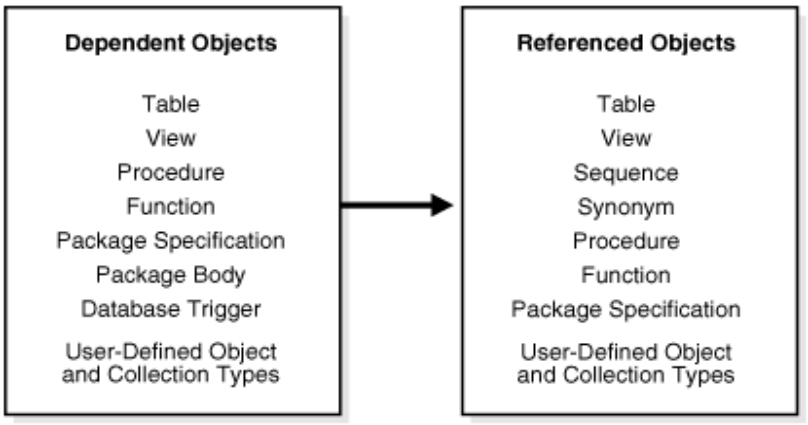
After completing this session, you will be able to:

- Comprehend dependencies among Schema Objects
- Comprehend Data Dictionary
- Identify Oracle Utilities
- Comprehend the overview of SQL

Dependencies among Schema Objects

The definitions of some objects, including views, procedures, reference other objects, such as tables are referred to here.

As a result, the objects are defined and dependent on the objects referenced in their definitions.



If you alter the definition of a referenced object, the dependent objects may or may not continue to function without error, depending on the type of alteration. For example, if you drop a table, no view based on the dropped table is usable.

Oracle automatically records dependencies among objects to alleviate the complex job of dependency management for the database administrator and users. For example, if you alter a table on which several stored procedures depend, Oracle automatically recompiles the dependent procedures the next time the procedures are referenced.

When a schema object is referenced directly in a SQL statement or indirectly through a reference to a dependent object, Oracle checks the status of the object explicitly, which is specified in the SQL statement and any referenced objects, as necessary.

Oracle's action depends on the status of the objects that are directly and indirectly referenced in a SQL statement.

If every referenced object is valid, then Oracle runs the SQL statement immediately without any additional work. If any referenced view or PL/SQL program unit (procedure, function, or package) is invalid, then Oracle automatically attempts to compile the object.

1. If all invalid referenced objects can be compiled successfully, then they are compiled and Oracle runs the SQL statement.
2. If an invalid object cannot be compiled successfully, then it remains invalid.
3. Oracle returns an error and rolls back the failing SQL statement.
4. The rest of the transaction is unaltered and can be committed or rolled back by the user.

Data Dictionary

An Oracle data dictionary is a set of tables and views that are used as a read-only reference about the database. A data dictionary stores information about both the logical and physical structure of the database.

A data dictionary also stores the following information:

- The valid users of an Oracle database
- Information about integrity constraints defined for tables in the database
- The amount of space allocated for a schema object and how much of it is in use

A data dictionary is created when a database is created. To accurately reflect the status of the database at all times, the data dictionary is automatically updated by Oracle in response to specific actions, such as when the structure of the database is altered. The database relies on the data dictionary to record, verify, and conduct ongoing work. For example, during database operation, Oracle reads the data dictionary to verify that schema objects exist and that users have proper access to them.

How to use the Data Dictionary

The data dictionary is always available when the database is open. It resides in the SYSTEM tablespace, which is always online. The data dictionary consists of sets of views.

Table: Data Dictionary

Prefix	Scope
USER	User's view (what is in the user's schema)
ALL	Expanded user's view (what the user can access)
DBA	Database administrator's view (what is in all user's schemas)

Oracle Utilities

Oracle's database utilities let you perform the following tasks:

- High-speed movement of data and metadata from one database to another using Data Pump Export and Import
- Extract and manipulate complete representations of the metadata for database objects, using the Metadata API
- Move all or part of the data and metadata for a site from one database to another, using the Data Pump API
- Load data into Oracle tables from operating system files using SQL*Loader or from external sources using external tables
- Query redo log files through a SQL interface with LogMiner
- Perform physical data structure integrity checks on an offline (for example, backup) database or datafile with `DBVERIFY`.
- Maintain the internal database identifier (`DBID`) and the database name (`DBNAME`) for an operational database, using the `DBNEWID` utility

Oracle Data Pump Technology

Oracle Data Pump technology enables very high-speed movement of data and metadata from one database to another.

This technology is the basis for Oracle's data movement utilities, Data Pump Export and Data Pump Import. The Data Pump API provides a high-speed mechanism to move all or part of the data and metadata for a site from one database to another.

To use the Data Pump API, you use the procedures provided in the `DBMS_DATAPUMP PL/SQL` package.

Oracle LogMiner

Oracle LogMiner enables you to query redo log files through a SQL interface.

All changes made to user data or to the database dictionary are recorded in the Oracle redo log files. Therefore, redo log files contain all the necessary information to perform recovery operations.

LogMiner functionality is available through a command-line interface or through the Oracle LogMiner Viewer graphical user interface (GUI). The LogMiner Viewer is a part of Oracle Enterprise Manager.

The following are some of the potential uses for data contained in redo log files:

- Pinpointing when a logical corruption to a database, such as errors made at the application level, may have begun. This enables you to restore the database to the state it was in just before corruption.
- Detecting and whenever possible, correcting user error, which is a more likely scenario than logical corruption. User errors include deleting the wrong rows because of incorrect values in a `WHERE` clause, updating rows with incorrect values, dropping the wrong index, and so forth.

- Determining what actions you would have to take to perform fine-grained recovery at the transaction level. If you fully understand and take into account existing dependencies, it may be possible to perform a table-based undo operation to roll back a set of changes.
- Performance tuning and capacity planning through trend analysis. You can determine which tables get the most updates and inserts. That information provides a historical perspective on disk access statistics, which can be used for tuning purposes.
- Performing post-auditing. The redo log files contain all the information necessary to track any DML and DDL statements run on the database, the order in which they were run, and who executed them.

External Tables

External tables access data in external sources, as if it were in a table in the database.

You can connect to the database and create metadata for the external table using DDL.

The DDL for an external table consists of two parts:

- One part that describes the Oracle column types.
- Second part that describes the mapping of the external data to the Oracle data columns.

To use the external tables feature, you must have some knowledge of the file format and record format of the datafiles on your platform. You must also know enough about SQL to be able to create an external table and perform queries against it.

SQL*Loader

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

You can use SQL*Loader to do the following:

- Load data from multiple datafiles during the same load session.
- Load data into multiple tables during the same load session.
- Specify the character set of the data.
- Selectively load data (you can load records based on the records' values).
- Manipulate the data before loading it, using SQL functions.
- Generate unique sequential key values in specified columns.
- Load data from disk, tape, or named pipe.

A typical SQL*Loader session takes as input a control file, which controls the behavior of SQL*Loader, and one or more datafiles. The output of SQL*Loader is an Oracle database (where the data is loaded), a log file, a bad file, and potentially, a discard file.

Overview of SQL

SQL is a database access, nonprocedural language.

Users describe in SQL what they want done, and the SQL language compiler automatically generates a procedure to navigate the database and perform the desired task.

Oracle Database is broadly compatible with the SQL-99 Core specification:

- Data Manipulation Language (DML) Statements
- Data Definition Language (DDL) Statements
- Transaction Control Statements
- Session Control Statements
- System Control Statements
- Embedded SQL Statements

Summary

- If you alter the definition of a referenced object, then the dependent objects may or may not continue to function without error.
- An Oracle data dictionary is a set of tables and views that are used as a read-only reference about the database.
- Oracle Data Pump technology enables very high-speed movement of data and metadata from one database to another.
- Oracle LogMiner enables you to query redo log files through a SQL interface.

Test Your Understanding

1. What happened to a view when its underlying table is altered?
2. How can you see all the tables owned by me?
3. What are the different options (utilities) available in Oracle to load data from Datafiles to Database?
4. Can you read a text file from Oracle?
5. What are the different prerequisite to load a data from SQL*LOADER?

References

Websites

- http://otn.oracle.com
- http://www.docnmail.com/learn/Oracle.htm

Books

- Oracle Database Concepts Guide Release 2 (10.2)

STUDENT NOTES: