chapitre 11

Changes to a database can be made using different types of SQL statements:
Data Manipulation Language (DML),
Data Definition Language (DDL), (create /alter/drop/remove)
and Data Control Language (DCL). (grant revoke)
While DDL and DCL affect the data dictionary, DML (such as INSERT, DELETE, UPDATE, and MERGE) directly changes application data.

A transaction is a logical unit of work where all changes are either committed or rolled back together to maintain data consistency.
can consist of multiple SQL statements that must all be executed as a single unit. If any of the statements fail, the entire transaction is rollback
Transactions in a database management system (DBMS) must adhere to ACID properties:
Atomicity::Ensures that all operations within a transaction are completed; if not, the transaction is entirely rolled back

Consistency ensures that a transaction brings the database from one valid state to another.
This means that the data should always meet all the predefined rules (constraints) of the database.

Isolation ensures that the operations within a transaction are invisible to other transactions until the transaction is committed. This prevents other transactions from seeing intermediate states or partial updates, ensuring that they only see the data in a consistent state

Durability The changes made by the committed transactions are saved and the system must not lose the change, even if there is a system failure. The database must guarantee that a committed change is never lost.

oracle use undo and redo mechanism to ensures ACID properties

- A transaction begins with the first DML (Data Manipulation Language) statement in a session. It continues until the changes are either committed (saved) or rolled back (discarded) transaction start with DML and end with commit or rollback

- Changes made during a transaction are not immediately visible to other sessions. They become visible only after a COMMIT operation.

- undo segment:store the original data before changes are made.,Store the before-image of data to ensure consistency and allow rollbacks.,They provide a consistent view of the data to other sessions.They enable rollback of uncommitted transactions, restoring the original state.

- Log Buffer and Redo Logs: Changes are recorded in the log buffer and then written to online redo log files. This logging ensures that changes can be reapplied if necessary or rolled back during recovery if not committed.

- Buffer Cache: Holds data blocks for modification before they are written back to disk
- Checkpoint Process: Periodically writes dirty blocks to disk to ensure data durability.

Differentiating Undo and Redo

- Undo Data: Tracks the original values before any changes were made. It is used for: Rolling back a transaction Ensuring read consistency for queries and DML operations Instance recovery (undo data from the redo log is used for recovery) Flashback functionality to revert the database to a previous state

- Redo Data: Tracks the new values after changes are made. It is written to online redo log files and used for: Reapplying changes during instance recovery Archiving for point-in-time recovery Reading log files with log miner to reapply or undo changes

- **Key Differences**
  Storage: Undo space is dynamic, with segments allocated as needed,
  while the redo log buffer has a fixed size
  Purpose: Undo data "undoes" changes, while redo data "redoes" or re-applies changes for recovery purposes.
  Generation: All DML operations generate both undo and redo data. Even rolling back a change generates redo data.

both of them used for database recovery

Using Undo for Transaction Rollback

In a database, undoing changes made by DML commands (such as DELETE, INSERT, UPDATE, or MERGE) within a transaction is called rolling back the transaction. This process involves the following:

Rollback Information: The data needed to undo changes, stored in a special type of tablespace known as the undo tablespace.

Complete Transaction Rollback: When an entire transaction is rolled back, Oracle uses the rollback information to revert all changes made since the beginning of the transaction. This action also releases any locks on rows involved in the transaction and ends the transaction.

Failure Handling: If a client or network failure occurs, causing an abnormal termination of the user's connection, Oracle uses the undo information to automatically roll back the transaction, reverting all changes since the beginning. The undo mechanism ensures data integrity and consistency by allowing transactions to be safely reversed if necessary.

undo and redo for database recovery:

When the database is shut down abruptly (using SHUTDOWN ABORT) or crashes, instance recovery is required to bring the database back to a consistent state. This recovery process involves both undo and redo operations.

Oracle writes the redo log buffer to redo log files immediately after a commit. As such, any change in the database is captured in the log file,(no changes are lost)

Cache Recovery (Rolling Forward):

Purpose: Replay changes from redo log files since the last checkpoint

Process: Applies both committed and uncommitted changes to data files
Result: The database now contains all the committed changes and some uncommitted changes, leading to an inconsistent state.

Transaction Recovery (Rolling Back):
Purpose: Remove uncommitted changes to restore consistency.
Process: Uses undo segments from the undo tablespace to roll back uncommitted changes.
Result: The database is left with only committed changes, achieving a consistent state.

Achieving Read Consistency

Undo functionality in a database plays a crucial role in achieving read consistency, especially in scenarios where multiple users are accessing the same data simultaneously. Here's a breakdown of how undo achieves read consistency

Undo allows users who are querying rows involved in a DML transaction by another user or session to see a consistent view of the data. If one user begins to make changes to a table while another user is querying it, the querying user will not see the changes until their query completes and a new query is issued.

Undo segments are stored in an undo tablespace and are essential for maintaining data consistency. They store previous values of rows affected by DML commands within a transaction's undo segment.

Undo segments play a critical role in database operations by ensuring that users see a consistent view of the data, even when other users are making changes. They store previous versions of rows affected by transactions, enabling the database to provide read consistency to users querying the data. Understanding how undo segments work and their importance in maintaining data integrity is essential for effective database management and application design.

You can use **SET TRANSACTION READ ONLY**
you are telling the database that this transaction will not make any changes to the data. Instead, it will only read data.
uring a read-only transaction, all the queries you run will only see data that was committed before the transaction started. This means you won't see any changes made by other transactions that were committed after your read-only transaction began.

we can only do lock/select statement

Configuring and Monitoring Undo

Manual Undo Management (Not Recommended):
Allows DBAs to manually manage rollback segments for undo operations.
Uses `UNDO_MANAGEMENT=MANUAL` and requires setting `ROLLBACK_SEGMENTS`.
DBAs had to manually create, manage, and tune rollback segments

Automatic Undo Management (Recommended)
To configure automatic undo management, use the initialization parameters UNDO_MANAGEMENT, UNDO_TABLESPACE, and UNDO_RETENTION.

undo tablespace creation:Automatic Creation: If not explicitly specified, Oracle creates an undo tablespace named `SYS_UNDOTBS` when creating the database.

This command shows the current undo tablespace configured:
SQL> show parameter undo_tablespace;

This command switches the undo tablespace to `UNDO_BATCH`. After executing this command, the undo tablespace is changed
SQL> alter system set undo_tablespace=undo_batch;

UNDO_RETENTION Parameter:
UNDO_RETENTION is a setting in Oracle that controls how long (in seconds) undo information is kept after a transaction is committed.

Retention Time: Specifies how many seconds the undo information is kept.
Flexible: The database might overwrite the undo information earlier if it needs space for new transactions.

If you set UNDO_RETENTION to zero, Oracle will automatically adjust how long to keep undo information.
Oracle tries to keep at least 900 seconds (15 minutes) of undo information

Uncommitted Undo Information: Supports active transactions and is necessary for rollback or in case of transaction failure. This undo information is never overwritten.

Committed Undo Information (Unexpired Undo): No longer needed for active transactions but still required to satisfy the retention period specified by UNDO_RETENTION. This undo information can be overwritten if necessary.

Expired Undo Information: Undo information that is no longer needed for any transaction and can be overwritten to free up space..

lock mode in data base:
Lock modes in databases control how much and what kind of access other users have to the rows and tables you are using in your commands

Exclusive Lock Mode (X)
Purpose: Prevents the associated resource (table or row) from being shared.
Usage:
Acquired when a transaction modifies data (e.g., via INSERT, UPDATE, DELETE operations).
Ensures that only the transaction holding the exclusive lock can alter the resource until the lock is released.

Scope:
Exclusive locks on rows (RX mode) are acquired by DML statements (INSERT, UPDATE, DELETE) to prevent other transactions from modifying the same row concurrently.
Exclusive locks on tables (RX mode) prevent other transactions from running DDL (Data Definition Language) statements (e.g., ALTER TABLE) on the table while the lock is held.
Shared Lock Mode (S)
Purpose: Allows the associated resource to be shared among multiple users for reading purposes.
Usage:
Acquired by transactions that only read data (e.g., SELECT operations).
Allows multiple transactions to read the same resource concurrently.
Scope:
Multiple transactions can acquire shared locks on the same resource (table or row) simultaneously.
Shared locks are compatible with other shared locks but are not compatible with exclusive locks (an exclusive lock cannot be acquired while a shared lock is held).

lock mode
NOWAIT Mode
Using NOWAIT in a LOCK TABLE statement returns control to the user immediately if any locks already exist on the requested resource, as you can see in the following example:
SQL> lock table hr.employees
in share row exclusive mode
nowait;
lock table hr.employees
*
ERROR at line 1:
ORA-00054: resource busy and acquire with NOWAIT specified or timeout expired

WAIT Mode
You can tell Oracle to wait a specified number of seconds to acquire a DML lock
if no wait or wait exist oracle will wait indefinitely

chapitre12:
Oracle Net:
It is responsible for han-dling client-to-server and server-to-server communications,

server. Oracle Net manages the flow of information in the Oracle network infrastructure. First, it establishes the initial connection to the Oracle server, and then it acts as the messenger, passing requests from the client back to the server or passing them between two Oracle servers.

Oracle Net supports the use of middleware products(act as intermediaries between different software applications or services. They facilitate communication, data management, and operations between disparate systems)
such as Oracle Application Server and Oracle Connection Manager.These products allow n-tier architectures

ures and capabilities of Oracle Net

Connectivity

a client can connect to oracle by :PC-based application / a dumb terminal application, the client is connecting to the database via the Internet

Oracle Net can use many types of network protocols like TCP/IP and named pipes to connect different computers and operating systems.
allows Oracle Net to connect to a wide range of computers and a wide range of operating environments.

Oracle Net can run on many operating systems, from
Windows to all variants of Unix to large mainframe-based operating systems. Multiple Operating Systems

Java Database Connectivity (JDBC) allows Java applications to connect to Oracle databases using two types of drivers:
JDBC Oracle Call Interface (OCI) Driver:Installed on the client computer and used by Java applications on that computer. It works with Oracle Net to communicate with the database.
JDBC Thin Driver:Written in Java, so it works on any platform. It doesn't need to be installed on the client computer and directly interfaces with Oracle Net

Manageability

Oracle Net provides a variety of features that allow you to manage the components of an Oracle network

Scalability
Security
Accessibility

Understanding the Oracle Listener

The Oracle listener is a server-side Oracle networking component that facilitates connections between client computers and an Oracle database.

It acts like a big ear, listening for connection requests to Oracle services.

The Oracle listener is configured to handle connection requests for any number of databases that are configured on the server.

It can listen for requests coming through various network protocols, such as TCP/IP, named pipes

The client and the Oracle listener are on the same machine. The client connects to the database directly on the local machine.
The client and the Oracle listener are on different machines. The client connects to the database over the network

How Do Listeners Respond to Connection Requests?
The Oracle listener responds to client connection requests based on the server's network configuration and the type of connection requested.
The response can happen in one of two main ways
1. Dedicated Server Connections:
Each client connection is handled by its own server-side process, which is not shared with other client
connection Methods:
Direct: The listener directly connects the client to the server process.
Redirect: The listener initially handles the connection, then redirects the client to another process that takes over.
Requirement:
The listener process must be running on the same physical server as the database(s) for remote clients to use dedicated connections.

Direct Handoff Method:
Used when the client and the database are on the same server.
Connection Process:
Client Contact:
The client contacts the Oracle listener after resolving the service name.

Listener Action:The listener starts a dedicated process for the client.(just for that client)
The client inherits the network connection endpoint from the listener.(The new process created by the listener has its own network connection point (an endpoint) for communication.
The client's connection is transferred (or handed off) to this new process. Essentially, the client uses the same network connection details (endpoint) to communicate with the newly created process.)

Connection Established:The client is now connected to the dedicated server process.

Redirect Method:

Used when the client and the database server are on different machines.
Connection Process:

Client Contact:
The client contacts the Oracle listener after resolving the service name.

Listener Starts Dedicated Process:The listener starts a new dedicated process for the client on the database server.

Listener Sends Address:The listener sends a message back to the client with the address of this new dedicated process.

Client Connects to Dedicated Process:The client uses this address to establish a connection directly to the new dedicated process on the database server.

.

2. Shared Server Connections:

The listener can pass the connection request to a dispatcher in a Shared Server environment.

Connection Methods:**Direct**: The listener directly connects the client to a dispatcher.
Redirect: The listener initially handles the connection, then redirects the client to a dispatcher.

Database Resident Connection Pooling (DRCP):The listener initially handles the connection, then redirects the client to a dispatcher.
Enhances resource sharing and scales connection support

Oracle Shared Server: Direct Handoff Method

Direct Handoff in Oracle Shared Server:
Used when the client, listener, and database are all on the same machine.
Connection Process:

Client Contact:
The client contacts the Oracle listener after resolving the service name.

Listener Passes Request:The listener forwards the connection request to the dispatcher with the least load (least busy).

Connection Established:The client establishes a connection with this dispatcher.

LREG Updates Listener:The LREG process (process monitor) informs the listener about the number of active connections each dispatcher is handling

Redirect Method in Oracle Shared Server:

Used when the listener and the Oracle server are on different machines or the operating system does not support direct handoff.
Connection Process:

Client Contact:
The client contacts the Oracle listener after resolving the service name.

Listener Redirects Client:The listener sends the client information about the dispatcher port, then disconnects the initial connection.

Client Reconnects:The client uses the new information to connect to the dispatcher or server process.

Dispatcher Acknowledges:The dispatcher or server process acknowledges the connection to the client.

LREG Updates Listener:The LREG process (process monitor) updates the listener with information about the number of connections each dispatcher is handling, helping to balance the load.

managing the oracle listener:
During Oracle installation, a default listener is created with a basic configuration file (`listener.ora`).
Management Tools:

Oracle Net Manager: A graphical tool for configuring and managing Oracle network files.
Oracle Enterprise Manager Cloud Control: A comprehensive management tool for Oracle databases and their network configurations.
srvctl (Server Control Utility): A command-line tool for managing Oracle Real Application Clusters (RAC) and related services.
lsnrctl (Listener Control Utility): A command-line tool specifically for managing the Oracle listener.

Oracle Net Manager:

Provides a graphical interface for configuring network files.
Ensures files are created in a consistent format to reduce connection problems.
Network File Configurations:

Profiles: Configures sqlnet.ora.
Service Naming: Configures tnsnames.ora.
Listeners: Configures listener.ora.
Creating the Listener:

Default listener created during Oracle installation is called LISTENER.
Default settings in listener.ora:
Listener name: LISTENER
Port: 1521
Protocols: TCP/IP and IPC
Hostname: Default Host Name
SID name: Default Instance
In summary, Oracle Net Manager simplifies the configuration of network files and ensures consistency, while the default listener settings provide a standard starting point for Oracle network configuration.

manage the listener using the **lsnrctl**
lsnrctl Overview:

Tool: Command-line interface used to administer the Oracle listener.
Capabilities: Provides full configuration and administration capabilities.
Familiarity: Common tool used by Oracle administrators since early releases of Oracle.
Other Tools: Similar command-line tools exist for managing other Oracle network components, such as Connection Manager.
Key Points:

Usage: Used to start, stop, and manage listener services.
Configuration: Allows configuration of listener parameters such as ports, protocols, and service names.

Administration: Provides monitoring capabilities to view listener status and connections.
History: Long-standing tool in Oracle administration, familiar to Oracle database administrators.

starting the listener
To start an Oracle listener, you use the LSNRCTL (Listener Control) command
To start the default listener name LISTENER, just type start at the LSNRCTL> start
To start a different listener, type start followed by the listener's name start listener2
This process starts the listener and shows its configuration, network endpoints, services, and status.

Reloading the listener:
When you make changes to the listener.ora file, you need to reload the listener to apply these changes. This can be done without stopping the listener (manually,oracle net,enterprise manager).

Use the reload command in LSNRCTL. This command makes the listener reload the listener.ora file and apply the new settings without stopping the listener. (reload listener 2)

the listener continues running during the reload process.

status of the listener
The status command helps verify that the listener is active and provides important details such as log and trace file locations, uptime, and services handled by the listener.
How to Check the Status:
Simply type status at the LSNRCTL> status prompt to display the status of the default listener name LISTENER

service handel by the listener
The lsnrctl services command provides detailed information about the services managed by the listener. This includes information about the server processes, connection statistics, and the status of each service.

stop the listener
LSNRCTL>stop
Stopped: New incoming connections to the databases managed by the listener will be refused.
Stopping the listener does not affect currently connected sessions.

self registration
Oracle databases can automatically register their presence with a listener,means that the Oracle database can automatically tell the listener about its presence and the services it offers. You don't need to manually add these services to the listener.ora file.

LREG is a process that handles the registration of the database with the listener

listener configuration for self registration
For dynamic registration to work, the listener should either be the default listener or explicitly specified using the LOCAL_LISTENER parameter in the init.ora file.
Connection Description: Directly specify the connection details (host, port, etc.) of the listener.
TNS Alias: Use a TNS alias defined in the tnsnames.ora file, which maps to the connection details of the listener.

self registration step:
Automatic Registration Steps:

Instance Start: When the Oracle instance starts, the LREG process checks if the listener is running.

If the listener is already running, LREG sends the necessary information (like the instance name, service names, etc.) to the listener so that it can register the instance. This makes the instance available for client connections.

If the listener isn't running, LREG will periodically ep trying to contact the listener . Once the listener starts, LREG will send the registration information.

Manual Registration: You can force the instance to register with the listener using the command: ALTER SYSTEM REGISTER.

To enable dynamic registration in Oracle, you need to manually add the INSTANCE_NAME and SERVICE_NAMES parameters to the init.ora file (or set them in the SPFILE) and restart the database instance.

client-side name resolution option:
A name resolution method in the context of Oracle Net is a way for clients to resolve (or find) the network address of the Oracle server they need to connect to

User ID, Password, and Net Service Name: Clients need these three pieces of information to connect to an Oracle server.

Net Service Name: This provides the connect descriptor that details how to locate the Oracle service in the network

Names Resolution Methods:
Local Naming: Uses a file (tnsnames.ora) on each client to store the connection details.
Oracle Internet Directory (OID) Stores connection details in a central location useful for complex networks with many Oracle servers.
External Naming: Uses a non-Oracle system like Network Information Service (NIS) to manage service names.
Host Naming: Relies on DNS or local hosts files for name resolution.
Oracle Easy Connect: Simplifies the connection process. No configuration files needed.
    myserver:1521/orcl
Local Naming: Uses a local tnsnames.ora file to store service names and connect descriptors.