Database
2.3. Introduction database management systems

Lourdes Moreno López
Paloma Martínez Fernández
José Luis Martínez Fernández
Rodrigo Alarcón García

HULAT Human Language & Accessibility Technologies

Content

Introduction

Oracle Database Architecture

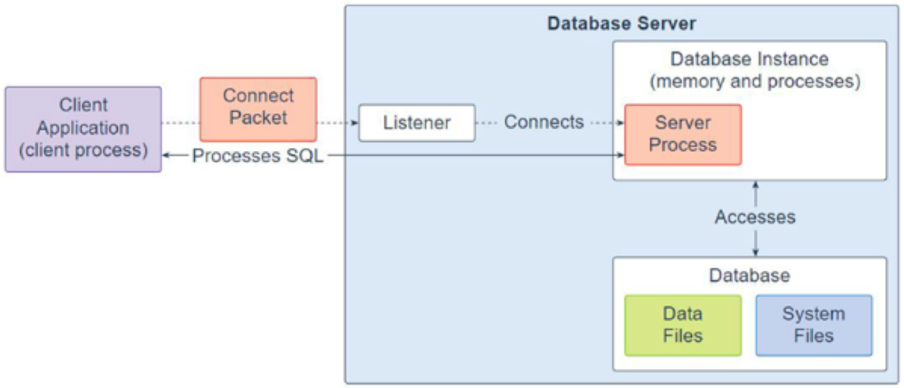
Database administrator's responsibilities

- Storage management: tablespace
- Administering User Accounts.
- Query Optimizer. Managing table Indexes.
- Transaction. Database Backup and Recovery

Introduction. Oracle Database Architecture

Oracle Database

- ORACLE, is a database server which manages a large amount of data in a multiuser environment so that users can concurrently access the same data.
- A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

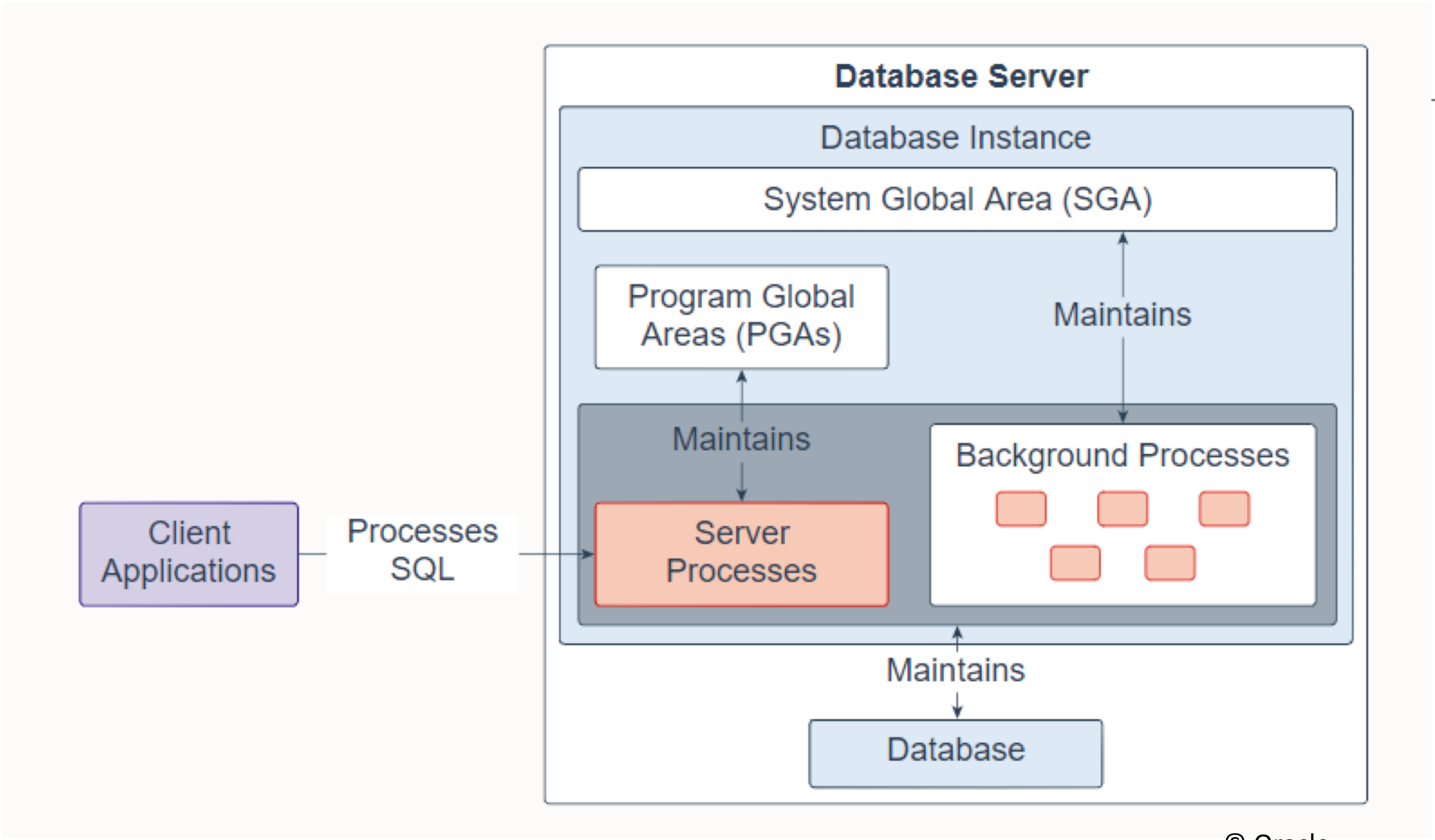


© Oracle

Database and Instance

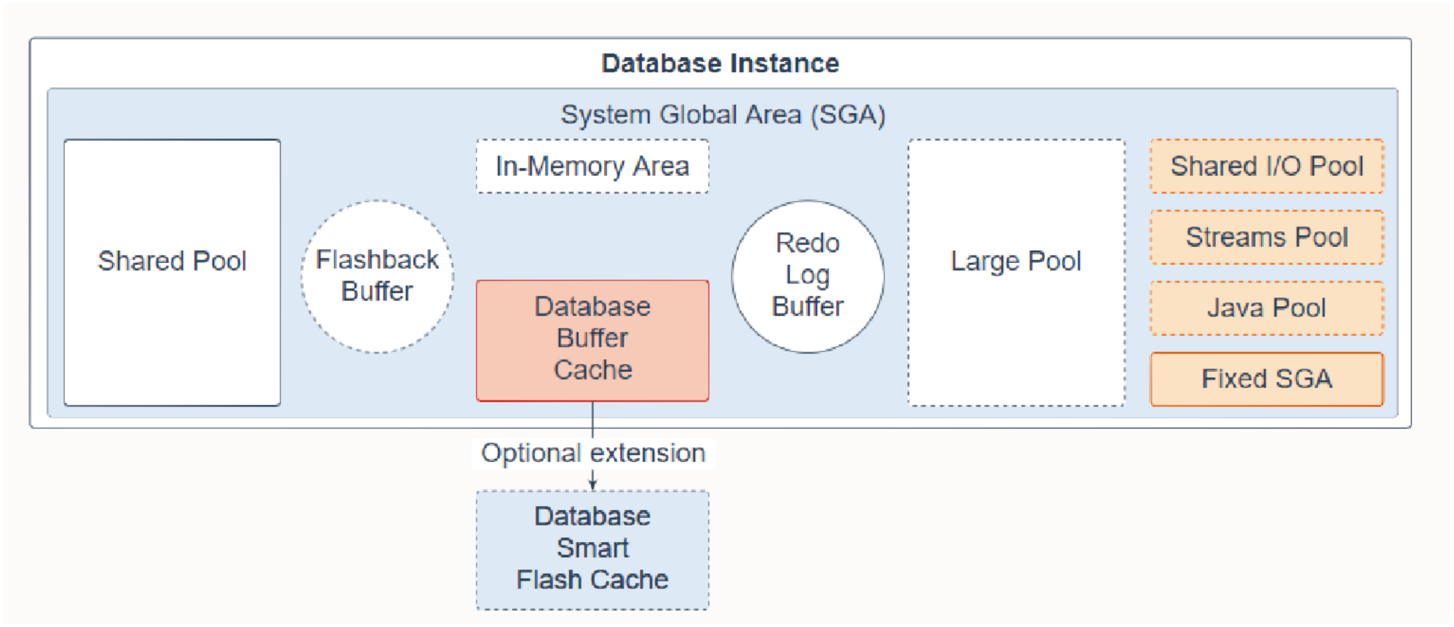
- An Oracle database server consists of a database and at least one database instance
  - **Database:** A database is a set of files, located on disk, that store data. These files can exist independently of a database instance.
  - **Database instance:** An instance is a set of memory structures that manage database files.
    - The instance consists of a shared memory area, called the system global area (SGA), and a set of background processes.
    - For each user connection to the instance, a client process runs the application. Each client process is associated with its own server process. The server process has its own private session memory, known as the program global area (PGA).

Database and Instance



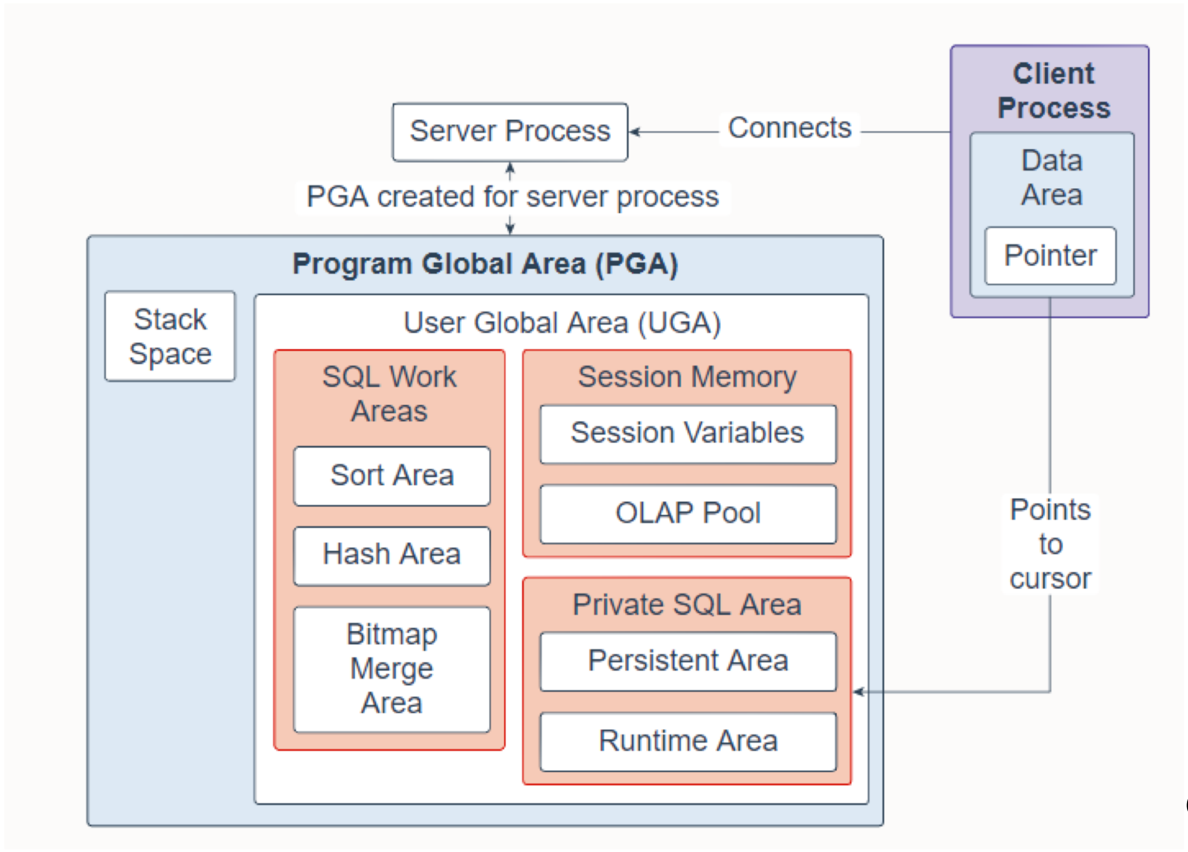
© Oracle

System Global Area (SGA)



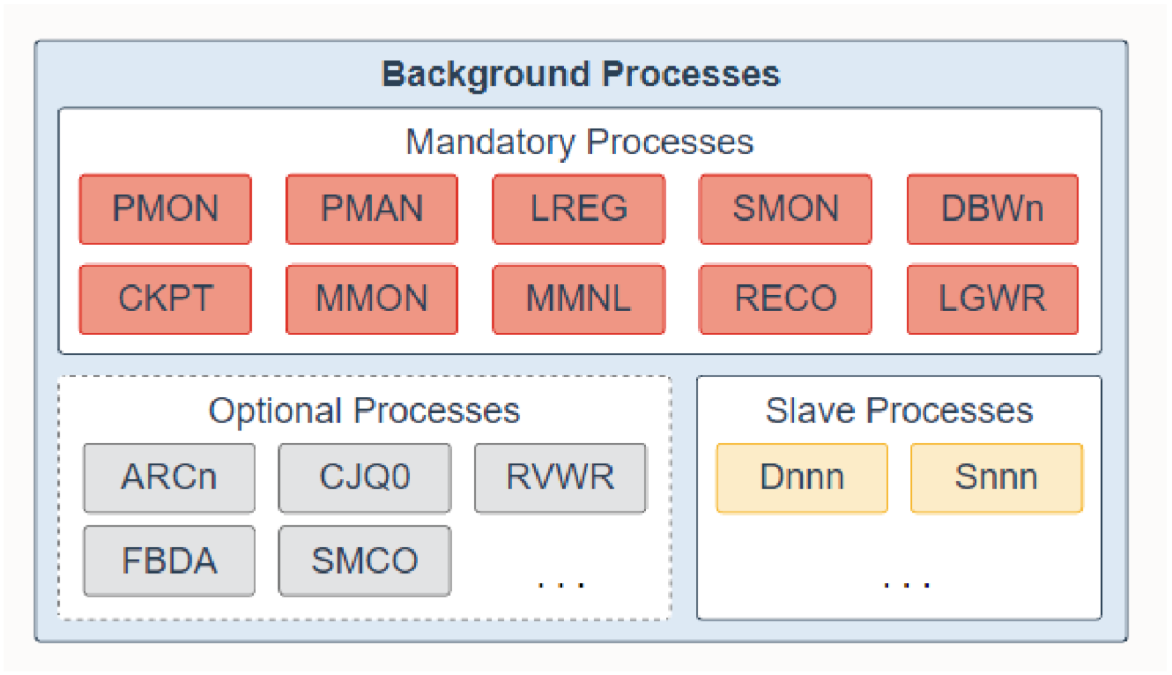
© Oracle

Program Global Area (PGA)



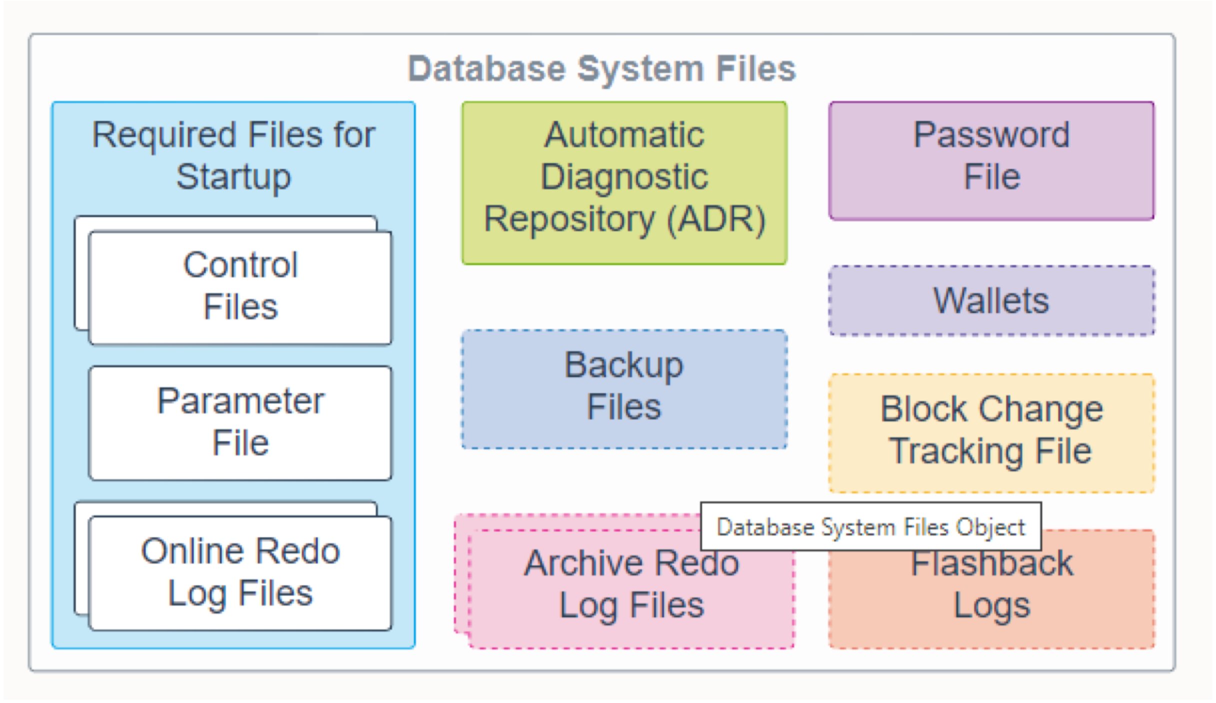
© Oracle

Background processes



© Oracle

Database system files



© Oracle

References

- Understanding the Oracle Database 12cR2 Technical Architecture https://www.oracle.com/webfolder/technetwork/tutorials/architecture-diagrams/12.2/technical-architecture/database-technical-architecture.html

Dictionary

- An important part of an Oracle database is its data dictionary, which **is a read-only set of tables that provides administrative metadata about the database.**
- A data dictionary contains information such as the following:
  - The definitions of every schema object in the database, including default values for columns and integrity constraint information
  - The amount of space allocated for and currently used by the schema objects
  - The names of Oracle Database users, privileges and roles granted to users, and auditing information related to users

Dictionary

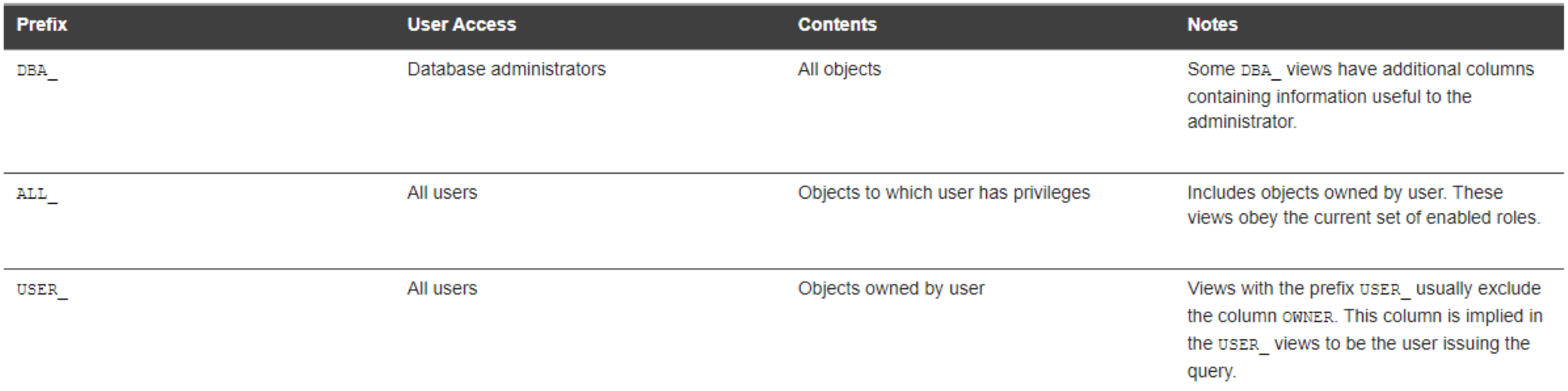
- The **data dictionary** is a central part of data management for every Oracle database
  - For example, the database performs the following actions:
    - Access to the data dictionary to find information about users, schema objects, and storage structures
    - Modify the data dictionary every time that a DDL statement is issued

Dictionary

- The data dictionary consists of base tables and views.
  - **Base tables:** These store information about the database. Only Oracle Database should write to and read these tables.
  - **Views:** The views contain the names and descriptions of all objects in the data dictionary. Some views are accessible to all database users, whereas others are intended for administrators only.

Dictionary

- Data Dictionary View Sets



Prefix	User Access	Contents	Notes
`DBA_`	Database administrators	All objects	Some `DBA_` views have additional columns containing information useful to the administrator.
`ALL_`	All users	Objects to which user has privileges	Includes objects owned by user. These views obey the current set of enabled roles.
`USER_`	All users	Objects owned by user	Views with the prefix `USER_` usually exclude the column `OWNER`. This column is implied in the `USER_` views to be the user issuing the query.

- Catalog Views / Data Dictionary Views: https://docs.oracle.com/cd/B28359_01/nav/catalog_views.htm
- System Tables and Views : https://docs.oracle.com/database/timesten-18.1/TTSYS/systemtables.htm#TTSYS346

Dictionary

- **Examples:**
  - SELECT * FROM dictionary
  - SELECT * FROM user_tables
  - SELECT * FROM user_catalog

  - DESCRIBE user_objects;

```
SELECT object_name, object_type, created, status
FROM  user_objects
ORDER BY object_type;
```

Dictionary

- **Examples:**
  - DESCRIBE user_tables;

    SELECT table_name
    FROM user_tables;

  - SELECT column_name, data_type, data_default, data_precision, data_scale, nullable
    FROM user_tab_columns
    WHERE table_name = 'EMPLOYEES';

Dictionary

- **Examples:**
  - SELECT CC.constraint_name, C.constraint_type, CC.table_name,
      CC.column_name, C.delete_rule, C.status, C.last_change
    FROM user_cons_columns CC, user_constraints C
    WHERE CC.constraint_name = C.constraint_name AND
CC.table_name in ('EMPLOYEE');

Introduction. Database administrator's responsibilities

Database administrator's responsibilities

- Coordinate all the activities of the database system
  - Administrator has a good understanding of the enterprise's information resources and needs.
- Database administrator's duties include:
  - Defining storage structure and access method
  - Modifying schema and physical organization
  - Granting users authority to access the database
  - Backing up data
  - Monitoring performance and responding to changes

Content

- Introduction
  - Oracle Database Architecture
  - Database administrator's responsibilities

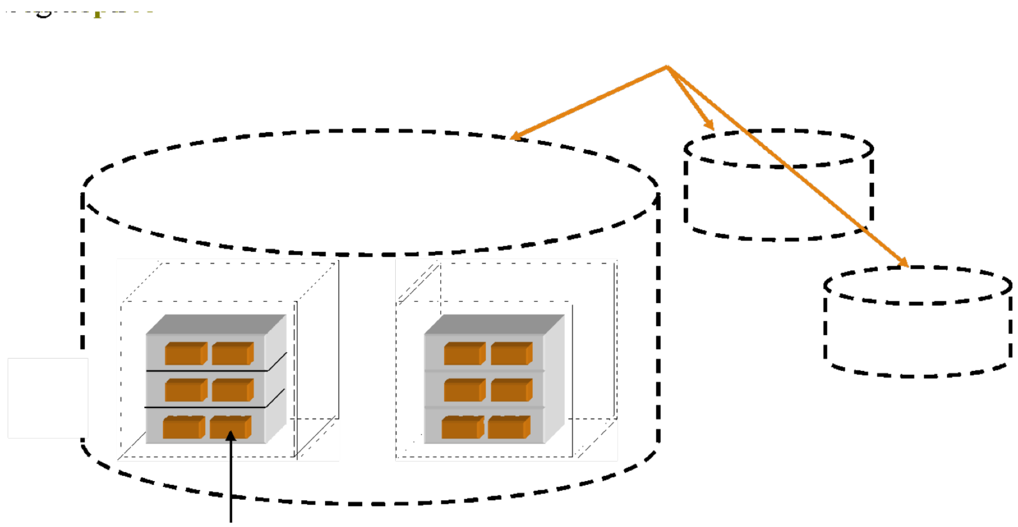
  Storage management: tablespace
- Administering User Accounts.
- Query Optimizer. Managing table Indexes.
- Transaction. Database Backup and Recovery

Storage management: tablespace

DATA BASE, BACHELOR IN DATA SCIENCE AND ENGINEERING, UC3M

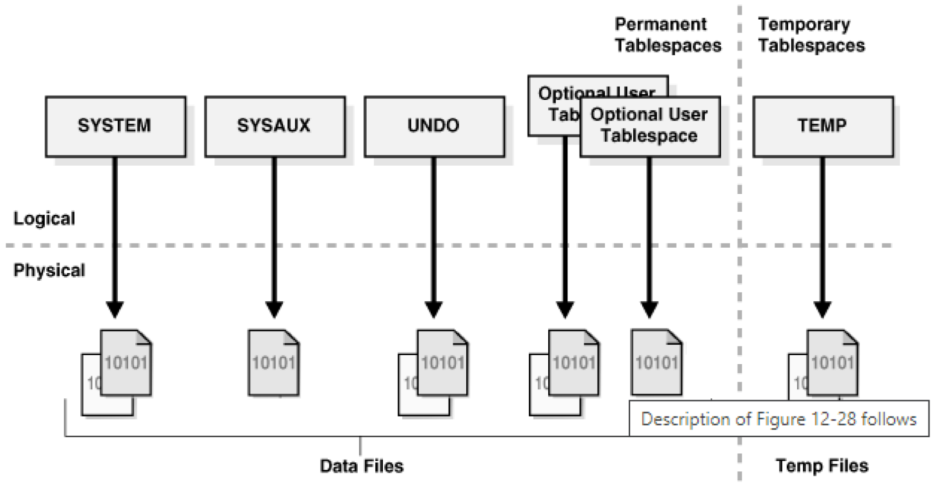
Logical structures

- The data files contain the data from the database. These files are logically grouped into a structure called **tablespace.**
- A tablespace is a logical unit of storage, it is composed of one or more physical files.
- Administration operations are performed by working with tablespaces.



Tablespace

- Permanent Tablespaces
  - SYSTEM
  - SYSAUX
  - UNDO
- Temporary Tablespaces
  - TEMP



© Oracle

SYSTEM Tablespace

- It is necessary to include a SYSTEM tablespace when a database is created.
- Oracle Database uses SYSTEM to manage the database.
- The SYSTEM tablespace includes the following information, all owned by the SYS user:
  - The data dictionary
  - Tables and views that contain administrative information about the database
  - Compiled stored objects such as triggers, procedures, and packages

SYSAUX Tablespace

- It is an auxiliary tablespace to the SYSTEM tablespace.
- SYSAUX is the default tablespace for many Oracle Database features and products that previously required their own tablespaces.
  - It reduces the number of tablespaces required by the database.
  - It also reduces the load on the SYSTEM tablespace.
- Database creation or upgrade automatically creates the SYSAUX tablespace.

UNDO Tablespace

- An UNDO tablespace is a locally managed tablespace reserved for system-managed undo data.
- UNDO tablespaces contain data files.

TEMP Tablespace

- It is a temporary tablespace contains transient data that persists only for the duration of a session.
- No permanent schema objects can reside in a temporary tablespace.

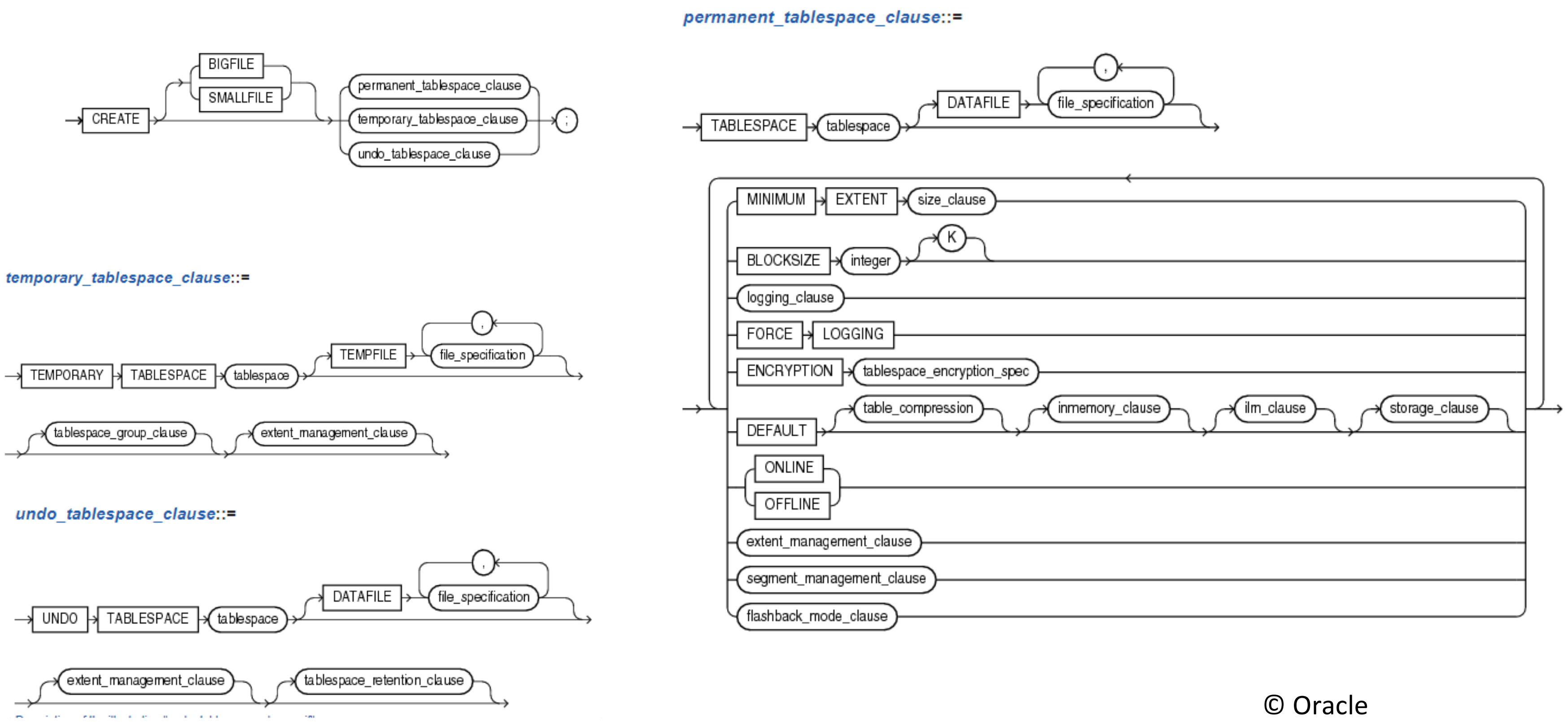
Online and Offline Tablespaces

- States of a tablespace:
  - Online: available to applications and DBs
  - Offline: data is not available, although the DB is running.

Tablespace File Size

- A tablespace is either a **bigfile** tablespace or a **smallfile** tablespace.
  - A smallfile tablespace can contain multiple data files or temp files, but the files cannot be as large as in a bigfile tablespace. This is the default tablespace type.
  - A bigfile tablespace contains one very large data file or temp file. This type of tablespace can do the following:
    - Increase the storage capacity of a DB
    - Reduce the burden of managing many data files and temp files
    - Perform operations on tablespaces rather than individual files

Syntax
CREATE TABLESPACE



© Oracle

https://docs.oracle.com/database/121/SQLRF/statements_7003.htm#SQLRF01403

Syntax CREATE TABLESPACE

Examples:

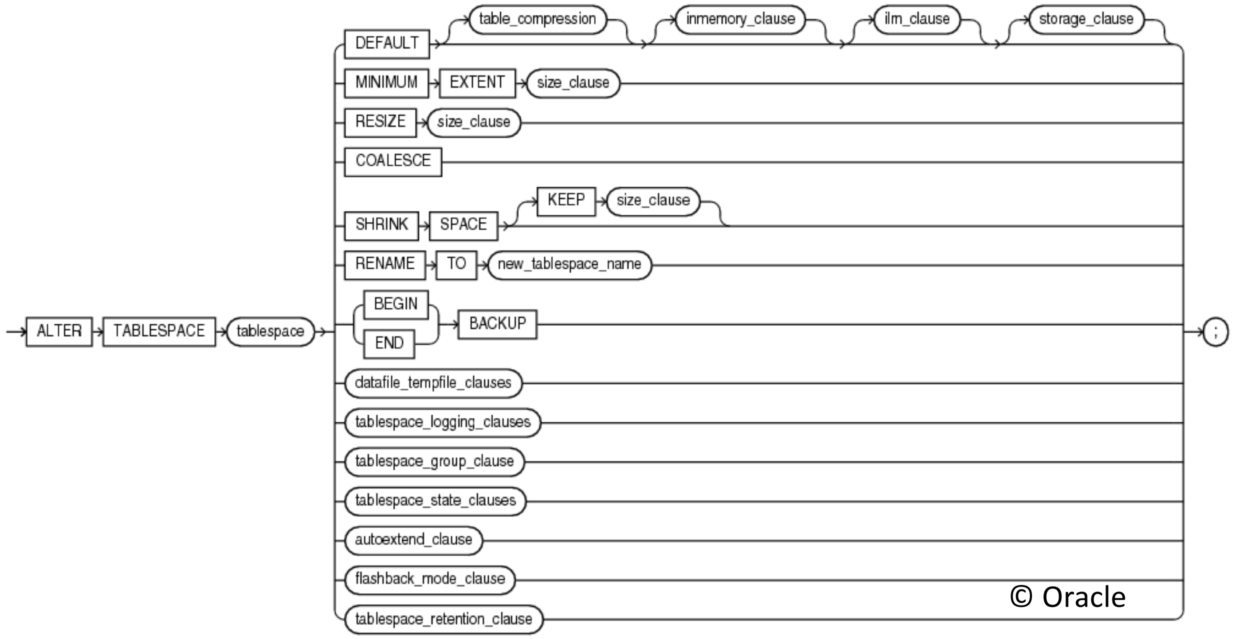
```
CREATE TEMPORARY TABLESPACE temp_demo
TEMPFILE 'temp01.dbf' SIZE 5M AUTOEXTEND ON;

CREATE TABLESPACE tbs_02
DATAFILE 'diskb:tbs_f5.dbf' SIZE 500K REUSE
AUTOEXTEND ON NEXT 500K MAXSIZE 100M;

CREATE UNDO TABLESPACE undots1
DATAFILE 'undotbs_1a.dbf'
SIZE 10M AUTOEXTEND ON
RETENTION GUARANTEE
```

Syntax ALTER TABLESPACE

- Use the ALTER TABLESPACE statement to alter an existing tablespace or one or more of its data files or temp files.

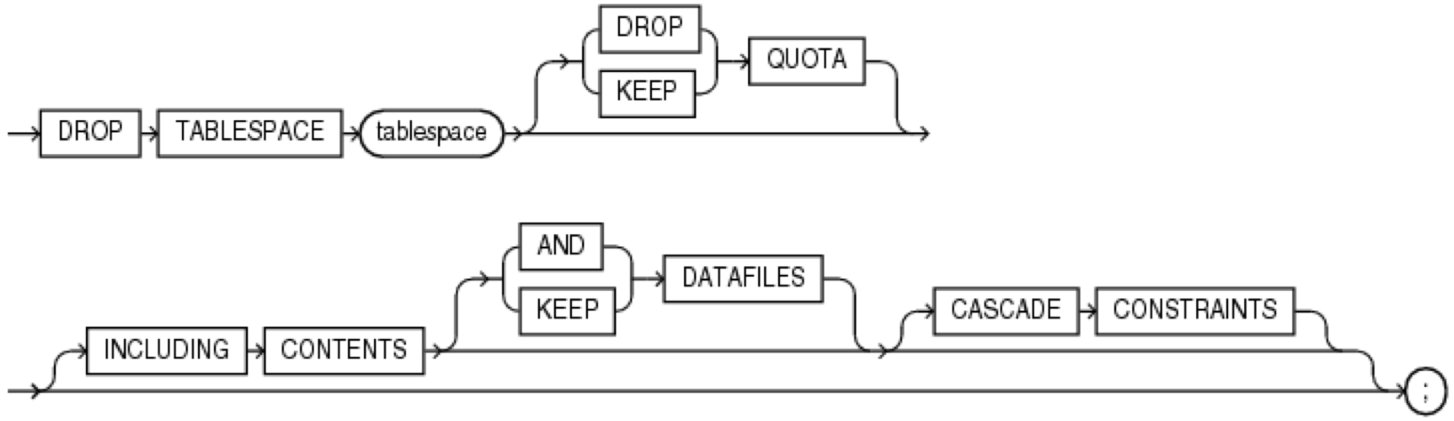


© Oracle

https://docs.oracle.com/database/121/SQLRF/statements_3002.htm#SQLRF01002

Syntax DROP TABLESPACE

- Use the DROP TABLESPACE statement to remove a tablespace from the DB



© Oracle

https://docs.oracle.com/database/121/SQLRF/statements_9004.htm#SQLRF01807

References

- Oracle 12. Overview of Tablespaces https://docs.oracle.com/database/121/CNCPT/logical.htm#CNCPT402

Content

Administering User Accounts

Introduction

- The term security refers to the protection of the DB against unauthorized access, either intentional or accidental.
- DB security refers to the mechanisms that protect the DB against intentional or accidental threats.
- We focus on the following computer-based security controls for a multi-user environment: authorization, access controls, backup and recovery

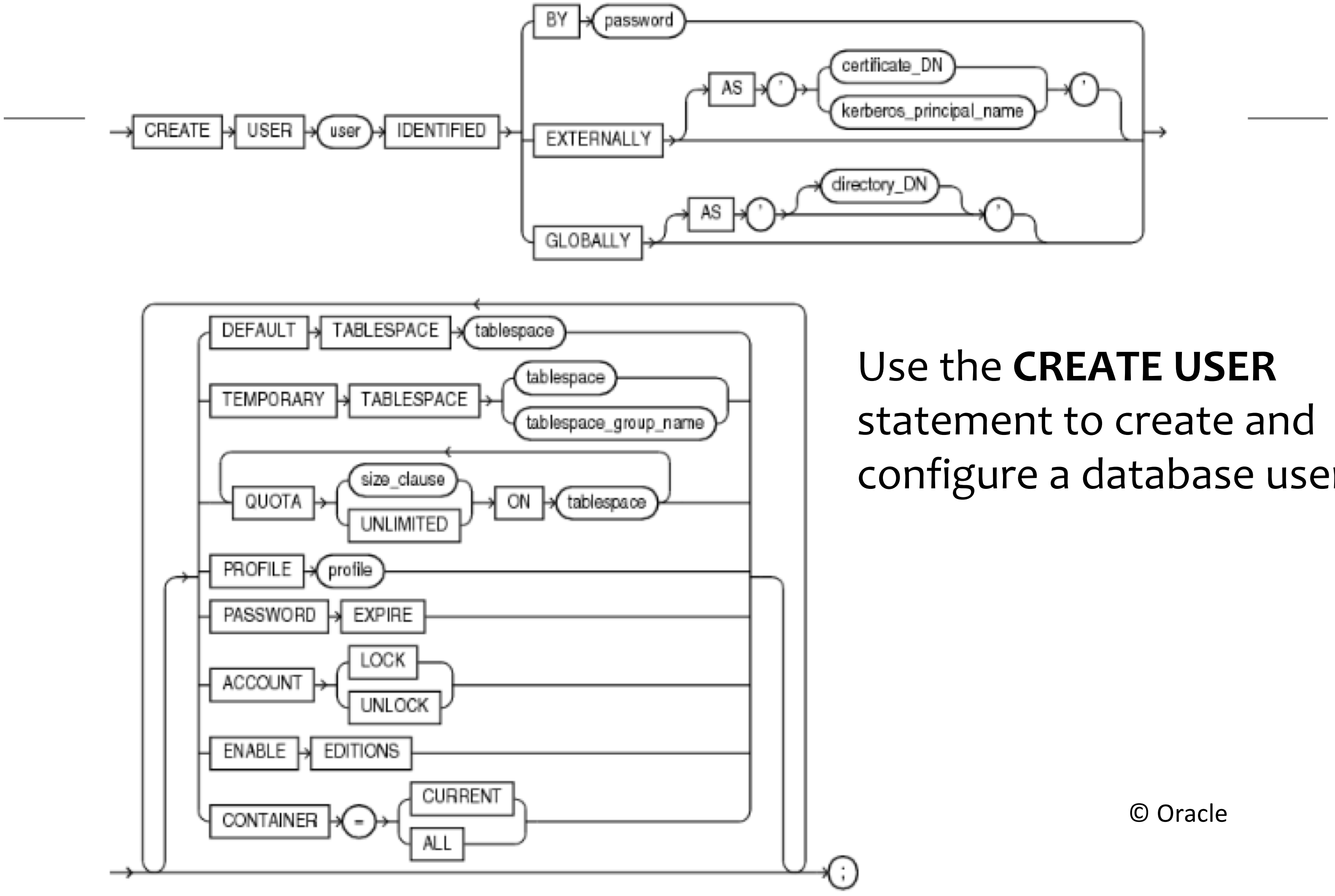
Introduction

- Security and confidentiality mechanisms
  - User accounts
  - Privileges and Roles
  - Profiles

User accounts

- For users to access your database, you must create user accounts and grant appropriate database access privileges to those accounts. A user account is identified by a user name and defines the attributes of the user, including the following:
  - Authentication method
  - Password for database authentication
  - Default tablespaces for permanent and temporary data storage
  - Tablespace quotas
  - Account status (locked or unlocked)
  - Password status (expired or not)

User accounts
CREATE USER



Use the **CREATE USER** statement to create and configure a database user,

© Oracle

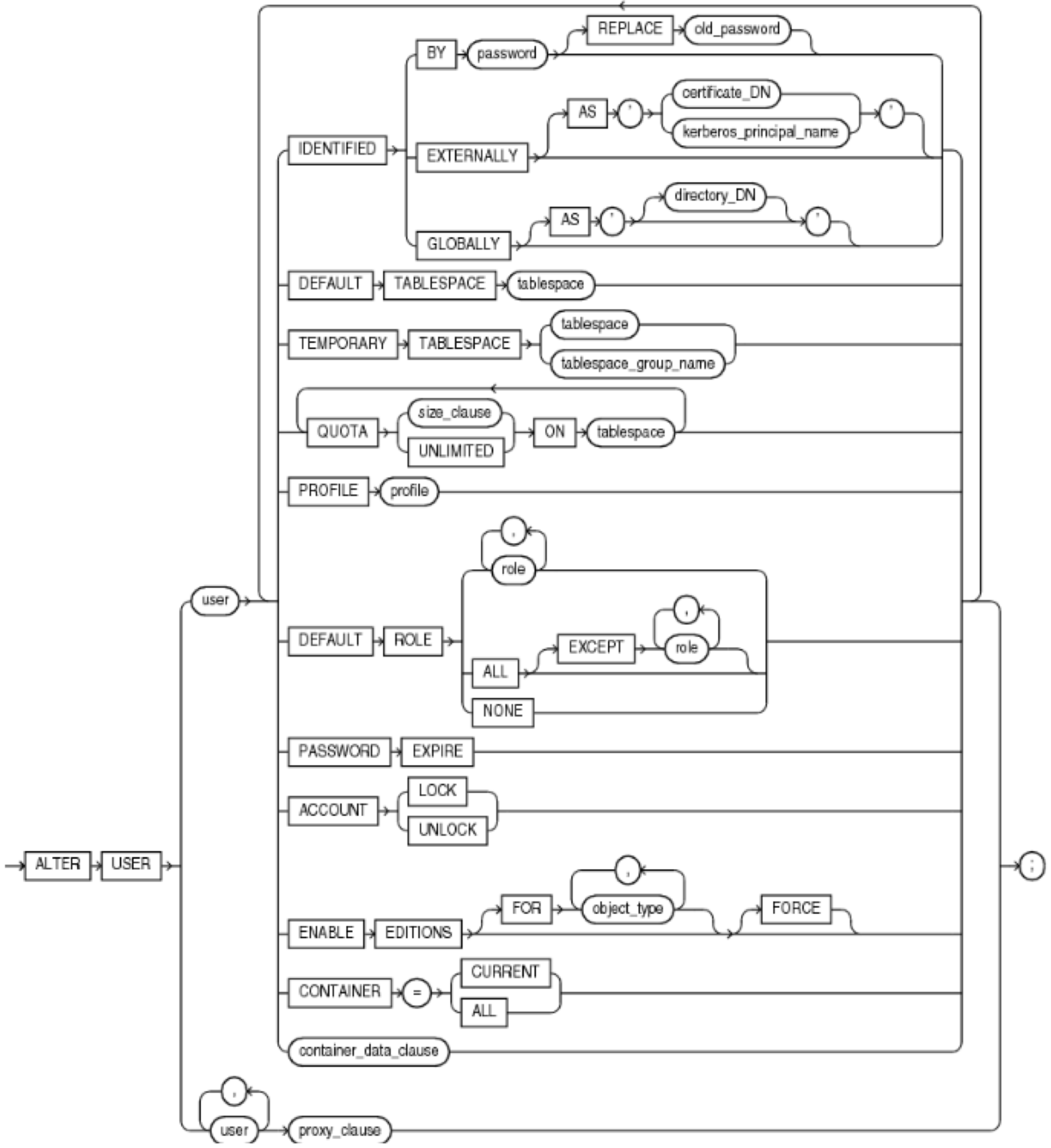
User accounts
CREATE USER

Example:

```
CREATE USER sidney
IDENTIFIED BY out_standing1
DEFAULT TABLESPACE example
QUOTA 10M ON example
TEMPORARY TABLESPACE temp
QUOTA 5M ON system
PASSWORD EXPIRE;
```

https://docs.oracle.com/database/121/SQLRF/statements_8003.htm#SQLRF01503

User accounts ALTER USER



Use the **ALTER USER** statement to change the authentication or database resource characteristics of a database user

© Oracle

User accounts ALTER USER

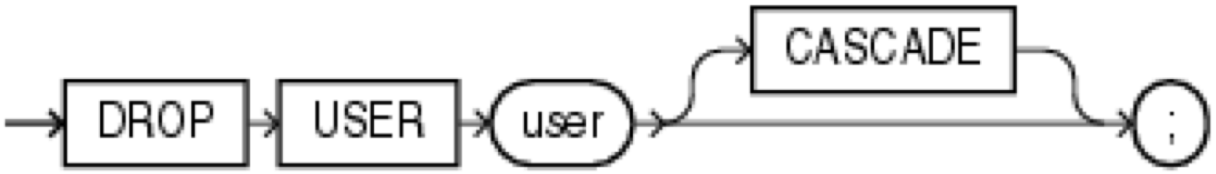
Example:

```
ALTER USER sidney
IDENTIFIED BY second_2nd_pwd
DEFAULT TABLESPACE example;
```

https://docs.oracle.com/database/121/SQLRF/statements_4003.htm#SQLRF01103

User accounts DROP USER

- Use the **DROP USER** statement to remove a database user and optionally remove the user's objects.



© Oracle

User accounts DROP USER

Examples:

```
DROP USER sidney;

DROP USER sidney CASCADE;
```

https://docs.oracle.com/database/121/SQLRF/statements_9008.htm#SQLRF01811

User Privileges

- When you create a user account, you must not only assign a user name, a password, and default tablespaces for the account, but you must also do the following:
  - **Grant the appropriate system privileges, object privileges,** and roles to the account.
  - If the user will be creating database objects, then give the user account a space usage quota on each tablespace in which the objects will be created.
- Oracle recommends that you grant each user just enough privileges to perform his job, and no more.

User Privileges

- **User privileges provide a basic level of database security.**
- They control user access to data and to limit the kinds of SQL statements that users can execute.
- When creating a user, you grant privileges to enable the user to connect to the database, to run queries and make updates, to create schema objects, and more.

https://docs.oracle.com/database/121/ADMQS/GUID-289A4BF6-F703-4ED5-8357-89F651A6D1DE.htm#ADMQS12001

User Privileges

- The main types of user privileges are:
  - **System privileges**—A system privilege gives a user the ability to perform a particular action, or to perform an action on any schema objects of a particular type. For example, CREATE TABLE, CREATE USER
  - **Object privileges**—An object privilege gives a user the ability to perform a particular action on a specific schema object. For example the privilege to select rows from the EMPLOYEES table or to delete rows from the DEPARTMENTS table

Roles

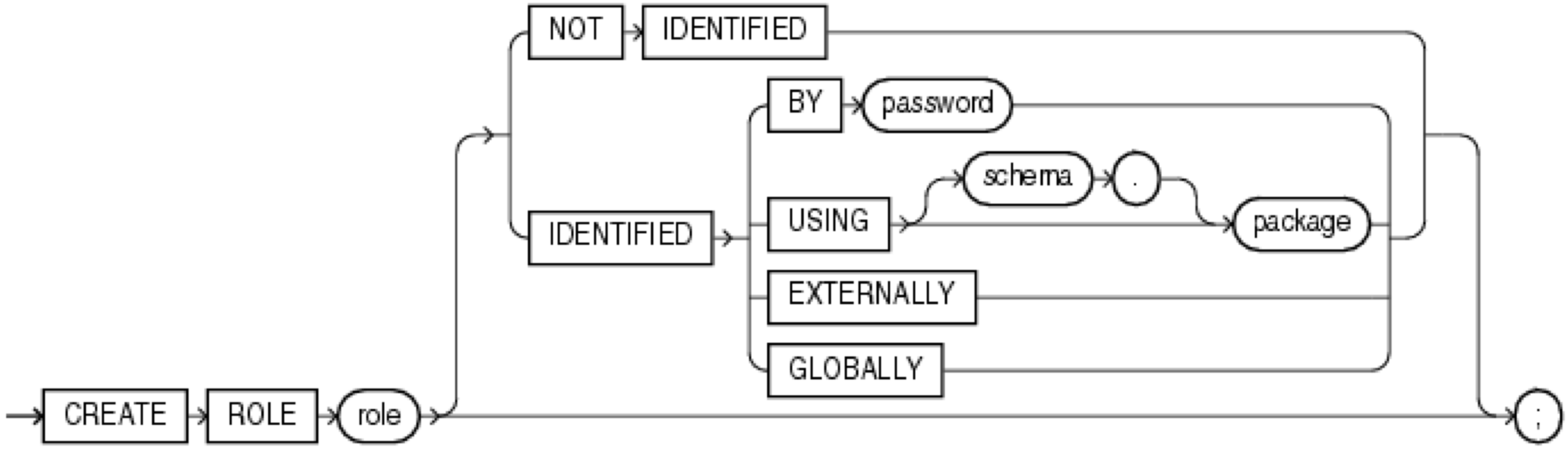
- Managing privileges is made easier by using **roles**, which are named groups of related privileges.
- You create roles, grant system and object privileges to the roles, and then grant roles to users.
- Oracle Database Predefined Roles: https://docs.oracle.com/database/121/ADMQS/GUID-289A4BF6-F703-4ED5-8357-89F651A6D1DE.htm#GUID-289A4BF6-F703-4ED5-8357-89F651A6D1DE__CHDHGCGH

Roles
CREATE ROLE

- Use the **CREATE ROLE** statement to create a role, which is a set of privileges that can be granted to users or to other roles.
- You can use roles to administer database privileges. You can add privileges to a role and then grant the role to a user.
- The user can then enable the role and exercise the privileges granted by the role.

https://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_6012.htm#SQLRF01311

Roles
CREATE ROLE



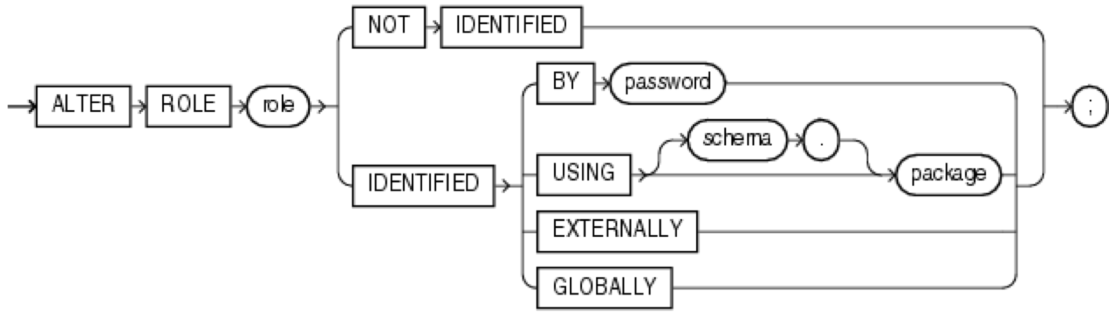
Examples:

```
CREATE ROLE dw_manager;

CREATE ROLE dw_manager IDENTIFIED BY warehouse;
```

Roles
ALTER ROL

- Modifying a role



- https://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_2009.htm#SQLRF00815
- **Examples ALTER**

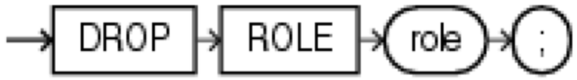
```
ALTER ROLE warehouse_user NOT IDENTIFIED;
```

© Oracle

```
ALTER ROLE dw_manager IDENTIFIED BY data;
```

Roles
DROP ROLE

- Delete a role



- https://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_8028.htm#SQLRF01530

- **Examples DROP**
    - `DROP ROLE dw_manager;`

© Oracle

Privileges to user, rol GRANT

- Use the GRANT statement to grant privileges:
  - **System privileges** to users and roles.

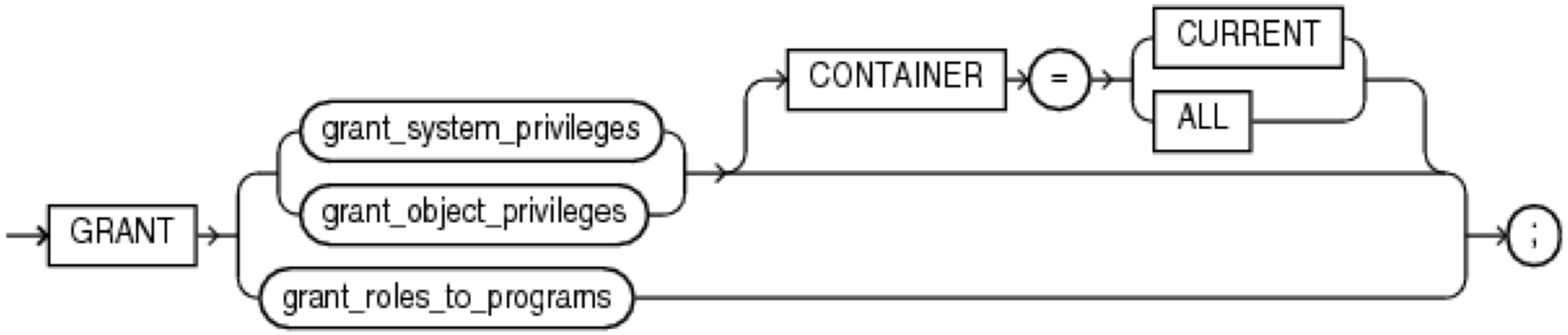
    https://docs.oracle.com/database/121/SQLRF/statements_9014.htm#BABEFFEE
  - **Roles to users.** The granted roles can be either user-defined (local or external) or predefined.
  - **Object privileges** for a particular object to users and roles.

    https://docs.oracle.com/database/121/SQLRF/statements_9014.htm#BGBCIIEG

Privileges to user, rol GRANT

- **System_privileges:** ALTER DATABASE, AUDIT SYSTEM, CREATE TABLE, DROP ANY TABLE, DELETE ANY TABLE, LOCK ANY TABLE, UPDATE ANY TABLE, SELECT ANY TABLE, CREATE VIEW, CREATE ROLE, ALTER ANY ROLE, DROP ANY ROLE, ALTER SESSION, CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER, CREATE TYPE, CREATE ROLLBACK SEGMENT, CREATE USER, etc.
- **Object_privileges:** LDD (ALTER, REFERENCES, INDEX); LMD (INSERT, DELETE, UPDATE, SELECT); READ, EXECUTE, etc.
- **Roles predefinidos:** CONNECT, RESOURCE, DBA, EXP_FULL_DATABASE, IMP_FULL_DATABASE, etc.

Privileges to user, rol GRANT



- https://docs.oracle.com/database/121/SQLRF/statements_9014.htm#SQLRF01603

© Oracle

Privileges to user, rol GRANT

Examples:

- **Granting System Privileges to a Role:**

```
GRANT CREATE SESSION TO hr;

GRANT CREATE ANY MATERIALIZED VIEW , ALTER ANY
MATERIALIZED VIEW , DROP ANY MATERIALIZED VIEW
, QUERY REWRITE , GLOBAL QUERY REWRITE TO
dw_manager;
```

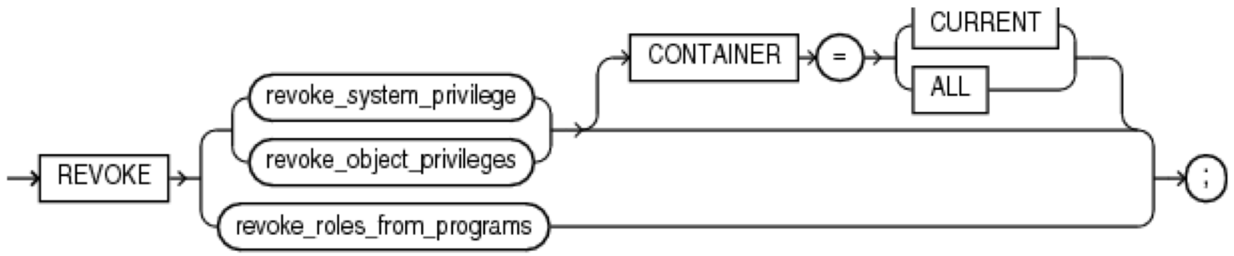
- **Granting Object Privileges to a Role: Examples :**

```
GRANT SELECT ON sh.sales TO warehouse_user;

GRANT SELECT, UPDATE ON emp_view TO PUBLIC;
```

Privileges to user, rol REVOKE

- Use the **REVOKE** statement to:
  - Revoke system privileges from users and roles
  - Revoke roles from users, roles, and program units.
  - Revoke object privileges for a particular object from users and roles



https://docs.oracle.com/database/121/SQLRF/statements_9022.htm#SQLRF01609

© Oracle

Privileges to user, rol REVOKE

Examples:

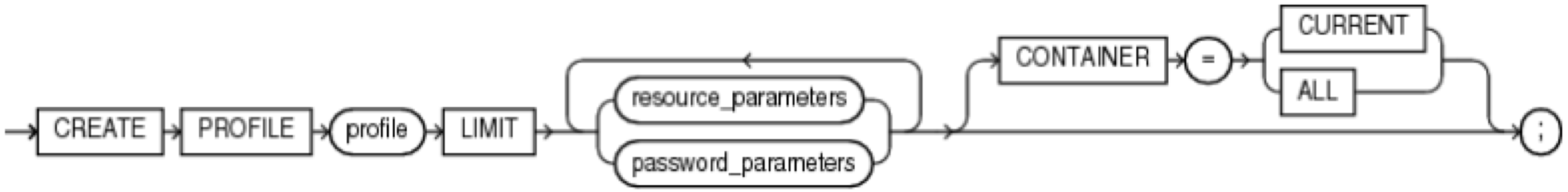
- **Revoking a System Privilege from a User:**
  - `REVOKE DROP ANY TABLE FROM hr, oe;`
  - `REVOKE CREATE TABLESPACE FROM dw_manager;`
- **Revoking an Object Privilege from a User:**
  - `REVOKE DELETE ON orders FROM hr;`
  - `GRANT SELECT, UPDATE ON emp_details_view TO public;`

Profiles

- A **profile is a mechanism that limits the use of resources**, with a name that can be assigned to a user.
- The following resources can be limited:
  - CPU time per call and/or per session
  - Number of logical reads per call and/or per session
  - Number of sessions open at the same time by a user
  - Idle time per session
  - The total duration of the session
- The following functionalities can be implemented:
  - Account lockout when a specified number of failed login attempts are exceeded
  - Passwords lifetime
  - …

Profiles
CREATE PROFILE

- Use the **CREATE PROFILE** statement to create a profile, which is a set of limits on database resources.
- If you assign the profile to a user, then that user cannot exceed these limits.



- https://docs.oracle.com/database/121/SQLRF/statements_6012.htm#SQLRF01310

© Oracle

Profiles
CREATE PROFILE

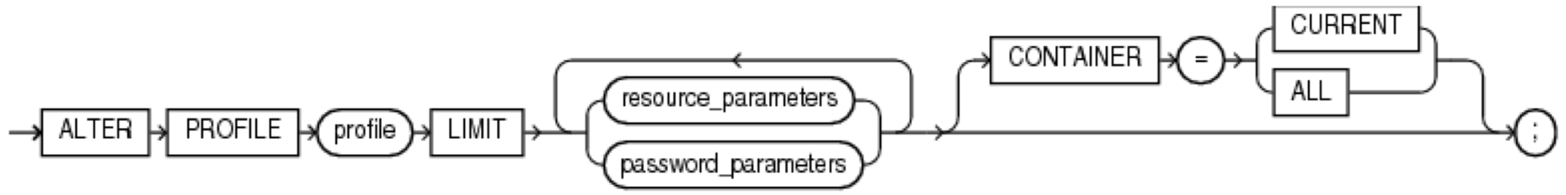
- **Examples:**

```
CREATE PROFILE name_profile LIMIT
PASSWORD_REUSE_MAX 10
PASSWORD_REUSE_TIME 30;

CREATE PROFILE name_profile LIMIT
SESSIONS_PER_USER 3
IDLE_TIME= 30
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LIFE_TIME 30
PASSWORD_REUSE_TIME 180;
```

Profiles
ALTER PROFILE

- Modify a profile



- https://docs.oracle.com/database/121/SQLRF/statements_2010.htm

- **EXAMPLES:**

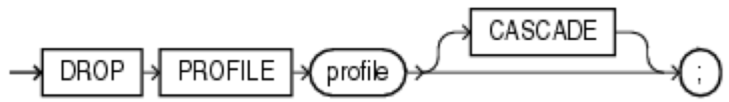
```
ALTER PROFILE name_profile LIMIT
PASSWORD_REUSE_TIME 90
PASSWORD_REUSE_MAX UNLIMITED;
```

© Oracle

```
ALTER PROFILE name_profile LIMIT SESSIONS_PER_USER
5;
```

Profiles
DROP PROFILE

- Delete a profile



- https://docs.oracle.com/database/121/SQLRF/statements_8030.htm#SQLRF01529

- **EXAMPLE:**

```
DROP PROFILE app_user CASCADE;
```

© Oracle

REFERENCES

- Oracle Database Online Documentation 12c Release 1 (12.1) https://docs.oracle.com/database/121/index.htm

- Administering User Accounts and Security: https://docs.oracle.com/database/121/ADMQS/GUID-7FC1D8BE-4BB9-4642-A4CE-29CD2B8A5F23.htm#ADMQS007

Content

- Introduction
  - Oracle Database Architecture
  - Database administrator's responsibilities
- Storage management: tablespace
- Administering User Accounts.
- **Query Optimizer. Managing table Indexes.**
- Transaction. Database Backup and Recovery

Managing table Indexes

Introduction

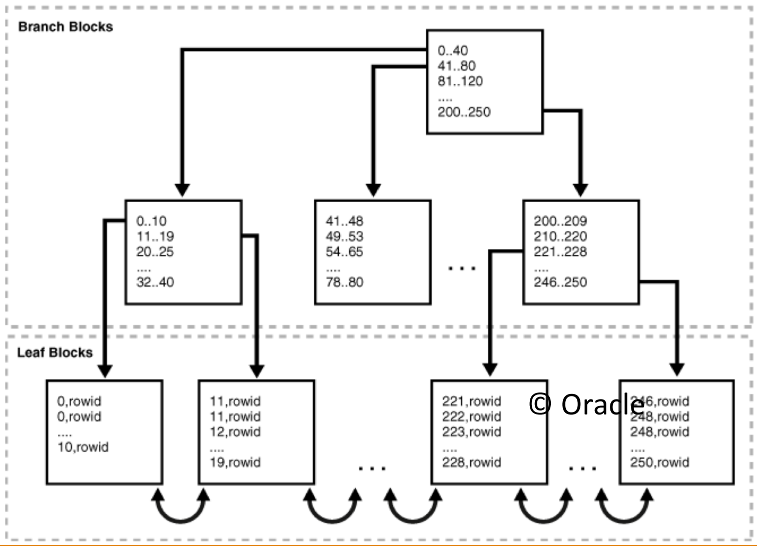
- The index allows quick access to the records in the table when you run a search based on the index key.
  - Analogy: index of a book
- An index is one more object in the database, it is stored in the dictionary, and it needs its own storage space.
- An index is a structure defined on one or more columns of a table; the column or columns constitute the index key.

Types of Indexes

- Oracle Database provides several indexing schemes, which provide complementary performance functionality.
  - B-tree indexes
  - Bitmap and bitmap join indexes
  - Function-based indexes
  - Application domain indexes:

Types of Indexes

- **B-trees (balanced trees)** is the most common type of database index.
- B-trees provide excellent retrieval performance for a wide range of queries, including exact match and range searches.
- B-tree indexes are excellent for primary key and highly-selective indexes .
- Subtypes B-tree indexes:
  - Index-organized tables
  - Reverse key indexes
  - Descending indexes
  - B-tree cluster indexes



Index selection criteria
Index and keys

- An index can be unique or non-unique:
  - **Unique** - An index key value is present only once in the table.
  - **Not Unique** - An index key value can be present multiple times in the table.
- Oracle advises not to create unique indexes explicitly, but rather to define integrity constraints (PRIMARY KEY or UNIQUE) **for which Oracle automatically creates unique indexes.** Non-unique indexes, on the other hand, must be explicitly created.

Index selection criteria Index and queries

- Candidate columns for indexing are the columns often included in WHERE clauses in selective queries
  - It is recommended to use an index when the query returns between 5% and 10% of the records in the table.

    (If a query uses an index that returns more than 10% of records, Oracle will have better performance fully traversing the table than using the index)

=> this implies that:

- § Column values are relatively unique (many different values) and that the WHERE conditions are selective.
- § Columns with few distinct values need not be indexed. Example: gender

- Do not use indexes in small tables
- Do not index columns IS NULL

Advantages and disadvantages

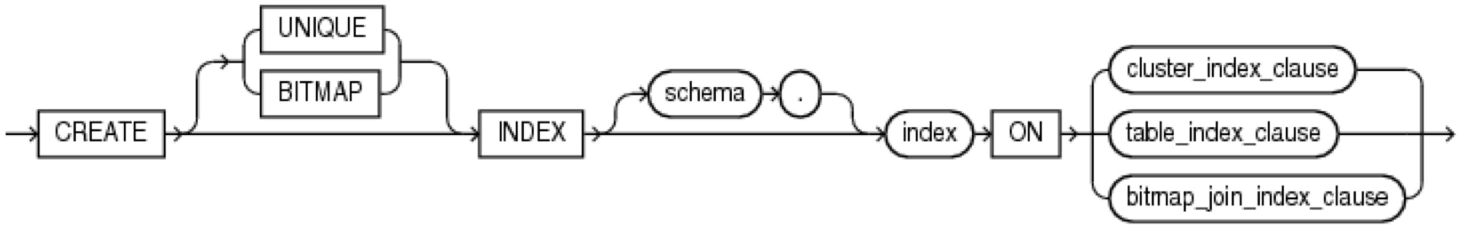
- Advantages: The index improves the performance of queries (SELECT, UPDATE, and DELETE) that use the index key in the WHERE clause.
- Disadvantages:
  - A large volume of storage is required.
  - The performance of updates is degraded. (The indexes are updated in each update (INSERT, UPDATE, DELETE))

=> therefore, you should not index all the columns of a table

Indexes and Performance Tuning

- Adding indexes to a schema is a common task in many database projects.
- As a performance-tuning task, usually occurs after DB contains some data, and queries are slow
  - Always avoid premature optimization!
  - Always find out what the DB is doing first!
- Indexes impose an overhead in both space and time
  - Speeds up selects, but slows down all modifications
- Always need to verify that a new index is actually being used by the database. If not, get rid of it!

CREATE INDEX



- https://docs.oracle.com/database/121/SQLRF/statements_5013.htm#SQLRF01209

© Oracle

CREATE INDEX

- Example:

Query:

```
  SELECT First_name, Lname
  FROM Employee
  WHERE UPPER(Lname)= "SMITH"
```

```
CREATE INDEX upper_ix ON Employee (UPPER(Lname));
```

CREATE INDEX

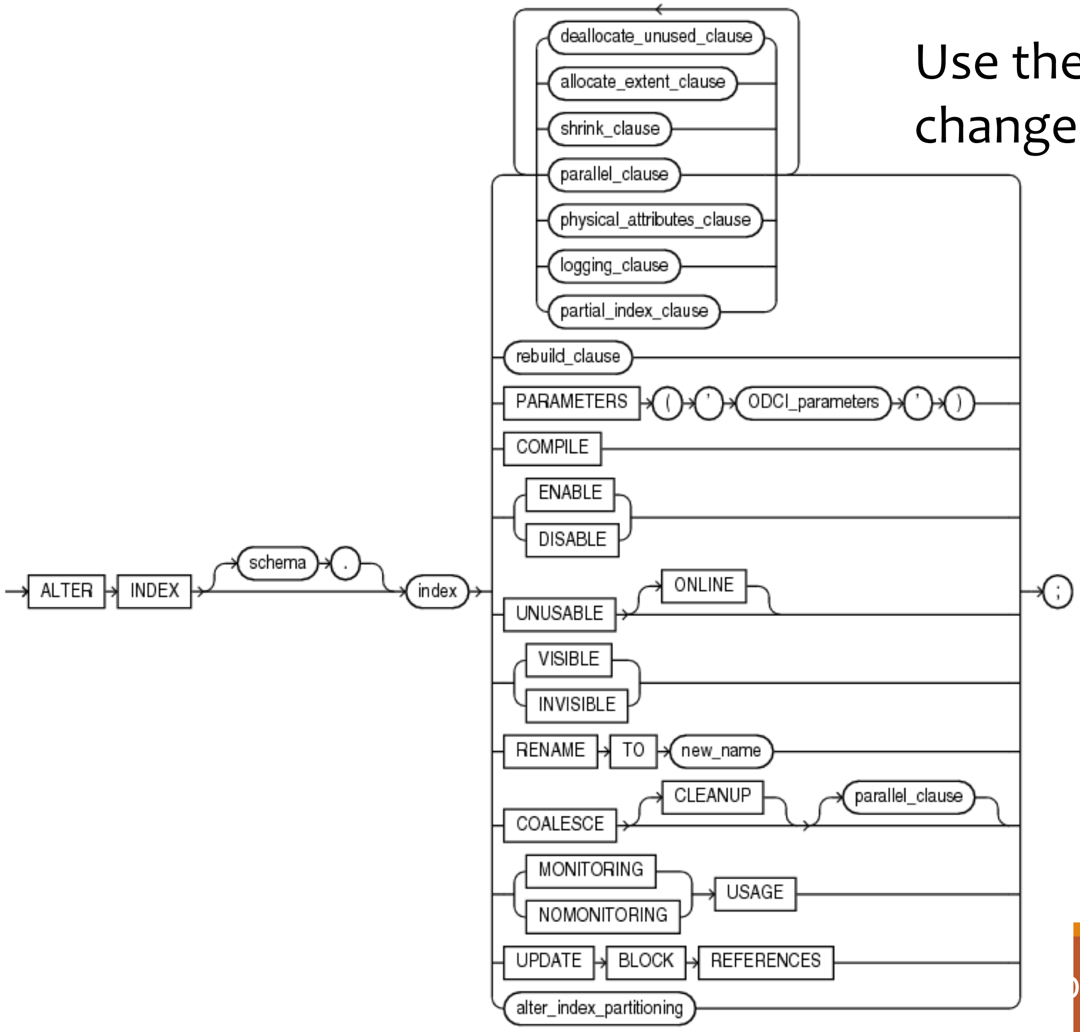
- Example:

Query:

```
SELECT First_name, Lname
FROM Employee
WHERE ((Salary*Commission_pct) + Salary ) >
15000

CREATE INDEX income_ix ON Employee(Salary +
(Salary*Commission_pct));
```

ALTER INDEX



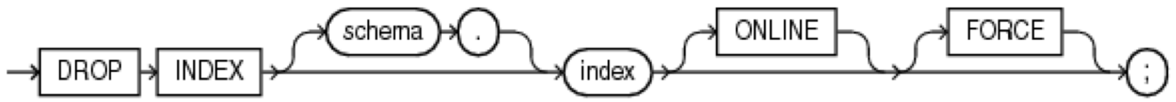
Use the ALTER INDEX statement to change or rebuild an existing index.

https://docs.oracle.com/database/121/SQLRF/statements_1012.htm#SQLRF00805

© Oracle

DROP INDEX

- Use the DROP INDEX statement to remove an index or from the database.



- https://docs.oracle.com/database/121/SQLRF/statements_8018.htm#SQLRF01510

© Oracle

VIEWS

- Several views of the data dictionary allow to obtain information from the indexes:
  - DBA_INDEXES
  - DBA_IND_COLUMNS
  - USER_IND_COLUMNS
  - INDEX_STATS

References

- Oracle 12 Indexes and Index-Organized Tables
https://docs.oracle.com/database/121/CNCPT/indexiot.htm#CNCPT721
- Oracle 12, CREATE INDEX
https://docs.oracle.com/database/121/SQLRF/statements_5013.htm#SQLRF01209

Query optimization

Introduction

- The goal is to reduce the execution time of the queries most frequently performed on a database
- How?
  - Modify the physical design
    - Add redundancy and modify organization: add or change indexes, split tables, partition tables.
  - **Query processing**

Query processing

- 1) **Query analysis**
  - Tree representation of the query structure
- 2) **Rewriting the query**
  - Representation in relational algebra
  - Transformation to a more efficient logical plan
- 3) **Generation of a physical execution plan**
  - Selection of algorithms for the execution of each logical operation

1) Query analysis

- Generation of the query tree
  - **Check SQL syntactic**
- **Preprocessing: Semantic Checks**
  - Relations: the tables exist in the DB
  - Attributes: are defined for the tables
  - Types: the types of the attributes used in the conditions are compatible

2) Rewriting the query Logical plan of the query

- Tree transformation to relational algebra
- **Algebraic optimization rules**
  - Commutativity / Associativity
  - Lower selections (reduces the number of tuples)
  - Lower projections (reduce size)
  - Remove subqueries from conditions

3) Generation of a physical execution plan. Types of optimization

- **Rule-based optimization**
  - General heuristics based on the experience of the administrator or designers
- **Cost-based optimization**
  - Estimation of the costs of performing a physical operation. It depends:
    - Structure size: statistics
    - Memory size: depends on the processes running simultaneously

ORACLE Cost-based optimization Query optimizer

- The **query optimizer** is a built-in database software that determines the most efficient method for a SQL statement to access requested data.
- Its purpose is to generate the **most optimal execution plan** for a SQL statement.
  - The optimizer chooses the plan with the lowest cost among all considered candidate plans.
  - The optimizer uses available statistics to calculate the cost.
- All SQL statements use the optimizer.

ORACLE Cost-based optimization ORACLE Query optimizer

- Example: a query might request information about employees who are managers.

  If the optimizer statistics indicate that 80% of employees are managers, then the optimizer may decide that a full table scan is most efficient. However, if statistics indicate that very few employees are managers, then reading an index followed by table access by rowid may be more efficient than a full table scan.

Cost-Based Optimization

- Query optimization is the overall process of choosing the **most efficient means** of executing a SQL statement.
- The database optimizes each SQL statement based on statistics collected about the accessed data.
- The optimizer determines the optimal plan for a SQL statement by examining multiple access methods, such as full table scan or index scans, different join methods such as nested loops and hash joins, different join orders, and possible transformations.

Cost-Based Optimization

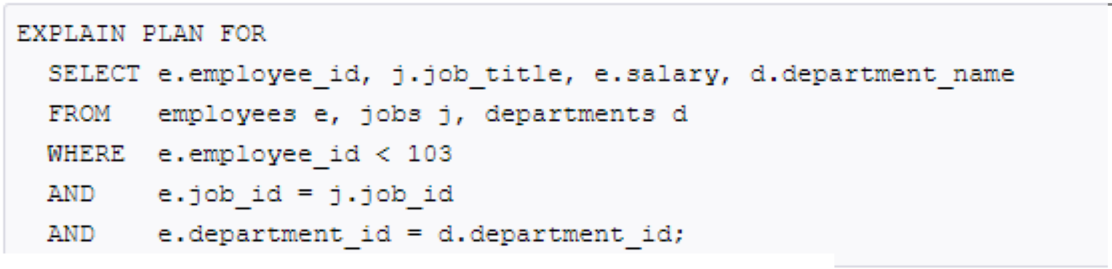
- The optimizer cost model accounts for the machine resources that a query is predicted to use.
  - System resources, which includes estimated I/O, CPU, and memory
  - Estimated number of rows returned (cardinality)
  - Size of the initial data sets
  - Distribution of the data
  - Access structures
- For a given query and environment, the optimizer assigns a relative numerical cost to each step of a possible plan.
- After calculating the costs of alternative plans, the optimizer chooses the plan with the lowest cost estimate.

Generating and Displaying Execution Plans

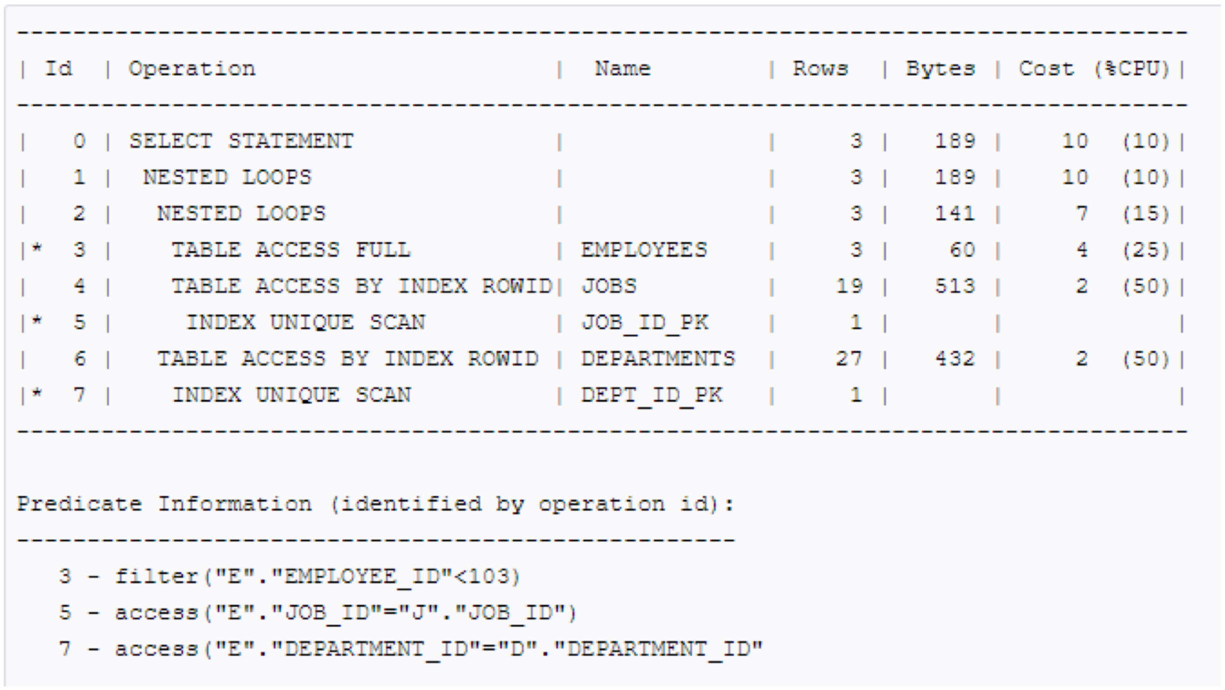
- An execution plan describes a recommended method of execution for a SQL statement.
- The plan shows the combination of the steps Oracle Database uses to execute a SQL statement.
- The **`EXPLAIN PLAN` statement** displays execution plans that the optimizer chooses for SELECT, UPDATE, INSERT, and DELETE statements

Generating and Displaying Execution Plans

- Example:



```
EXPLAIN PLAN FOR
  SELECT e.employee_id, j.job_title, e.salary, d.department_name
  FROM   employees e, jobs j, departments d
  WHERE  e.employee_id < 103
  AND    e.job_id = j.job_id
  AND    e.department_id = d.department_id;
```



```
-----------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)|
-----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |     3 |   189 |    10  (10)|
|   1 |  NESTED LOOPS                 |               |     3 |   189 |    10  (10)|
|   2 |   NESTED LOOPS                |               |     3 |   141 |     7  (15)|
|*  3 |    TABLE ACCESS FULL          | EMPLOYEES     |     3 |    60 |     4  (25)|
|   4 |    TABLE ACCESS BY INDEX ROWID| JOBS          |    19 |   513 |     2  (50)|
|*  5 |     INDEX UNIQUE SCAN         | JOB_ID_PK     |     1 |       |            |
|   6 |   TABLE ACCESS BY INDEX ROWID | DEPARTMENTS   |    27 |   432 |     2  (50)|
|*  7 |    INDEX UNIQUE SCAN          | DEPT_ID_PK    |     1 |       |            |
-----------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   3 - filter("E"."EMPLOYEE_ID"<103)
   5 - access("E"."JOB_ID"="J"."JOB_ID")
   7 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID"
```

© Oracle

Generating and Displaying Execution Plans

- How do we tell what indexes the DB uses for a query?
  How do we create additional indexes on our tables?

=> use the optimizer cost and query planner (execution plan) looks for indexes on relevant columns and optimizing your query

Views

- V$SQL_PLAN
- V$SQL_PLAN_STATISTICS
- V$SQL_PLAN_STATISTICS_ALL

References

- Database SQL Tuning Guide :
https://docs.oracle.com/database/121/TGSQL/toc.htm

Content

- Introduction
  - Oracle Database Architecture
  - Database administrator's responsibilities
- Storage management: tablespace
- Administering User Accounts.
- Query Optimizer. Managing table Indexes.

Transaction. Database Backup and Recovery

Transaction

Introduction

- A transaction is an atomic unit of work that should either be completed in its entirety or not done at all.
- For recovery purposes, the system needs to keep track of when each transaction starts, terminates, and commits, or aborts.
- Atomicity requirement
  - If the transaction fails after step 3 and before step 6, money will be "lost" leading to an inconsistent database state
    - Failure could be due to software or hardware
  - The system should ensure that updates of a partially executed transaction are not reflected in the database
- Durability requirement — once the user has been notified that the transaction has completed (i.e., the transfer of the $50 has taken place), the updates to the database by the transaction must persist even if there are software or hardware failures.

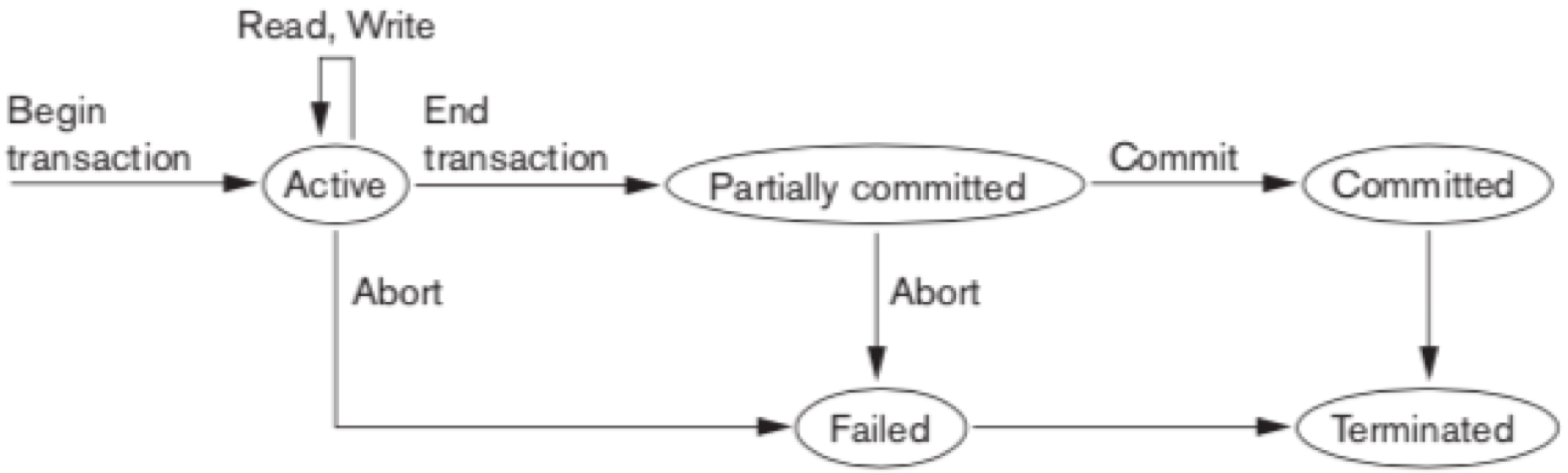
Transaction States and Operations

- BEGIN_TRANSACTION
- READ or WRITE
- END_TRANSACTION
- COMMIT_TRANSACTION
- ROLLBACK (or ABORT)

Transaction States and Operations

- **Active** – the initial state; the transaction stays in this state while it is executing
- **Partially committed** – after the final statement has been executed.
- **Failed** -- after the discovery that normal execution can no longer proceed.
- **Aborted** – after the transaction has been rolled back and the database restored to its state prior to the start of the transaction. Two options after it has been aborted:
  - Restart the transaction
    - can be done only if no internal logical error
  - Kill the transaction
- **Committed** – after successful completion.

Transaction States and Additional Operations



ACID Properties

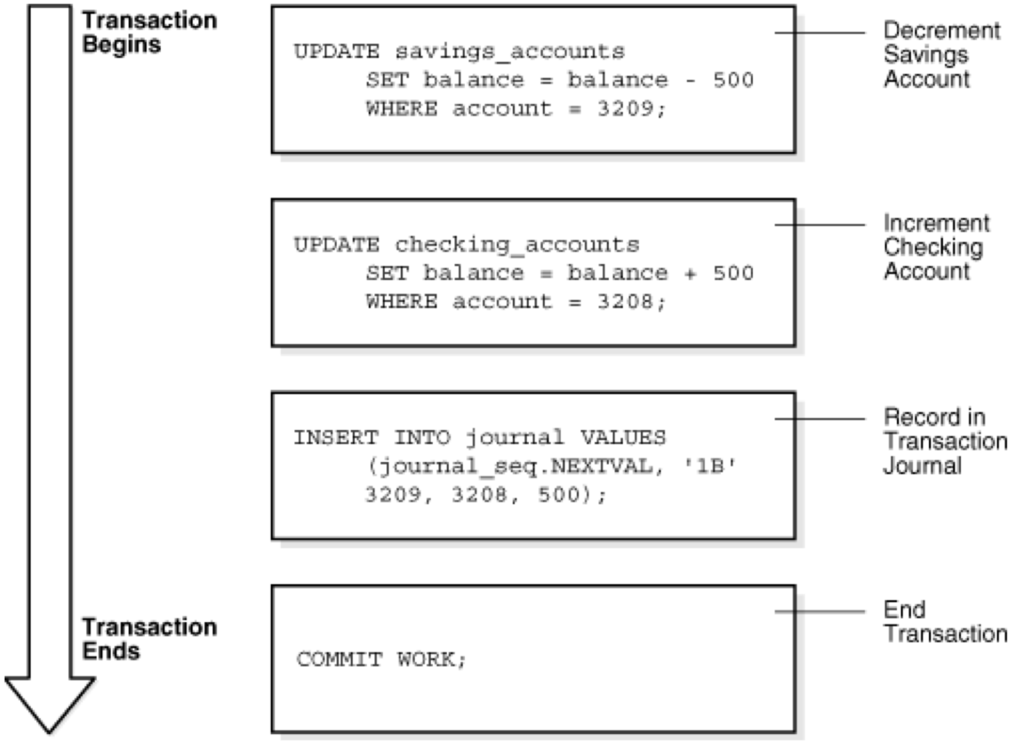
- **Atomicity**. Either all operations of the transaction are properly reflected in the database or none are.
- **Consistency**. Execution of a transaction in isolation preserves the consistency of the database.
- **Isolation**. Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions.
  - That is, for every pair of transactions Ti and Tj, it appears to Ti that either Tj, finished execution before Ti started, or Tj started execution after Ti finished.
- **Durabilit**y. After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

Transaction - ORACLE

- A transaction is a logical, atomic unit of work that contains one or more SQL statements.
- A transaction groups SQL statements so that they are either all committed, which means they are applied to the database, or all rolled back, which means they are undone from the database.
- All Oracle transactions obey the basic properties of a database transaction, known as ACID properties.

https://docs.oracle.com/database/121/CNCPT/transact.htm#CNCPT89321

Transaction - ORACLE Example: A Banking Transaction


Transaction Begins
Decrement Savings Account
Increment Checking Account
Record in Transaction Journal
Transaction Ends
End Transaction

```
UPDATE savings_accounts
     SET balance = balance - 500
     WHERE account = 3209;
```

```
UPDATE checking_accounts
     SET balance = balance + 500
     WHERE account = 3208;
```

```
INSERT INTO journal VALUES
     (journal_seq.NEXTVAL, '1B'
     3209, 3208, 500);
```

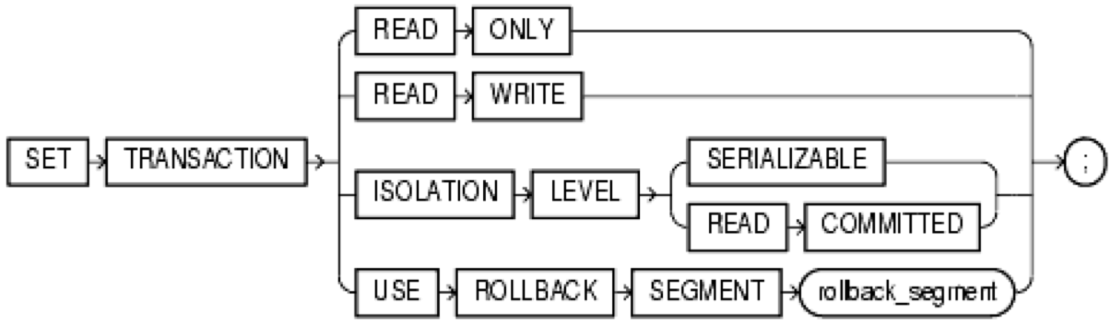
```
COMMIT WORK;
```

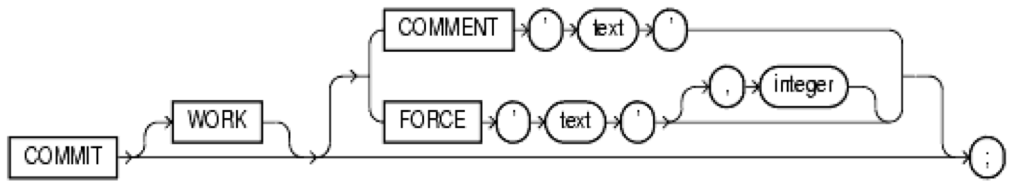
© Oracle

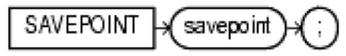
Transaction - ORACLE Transaction Control

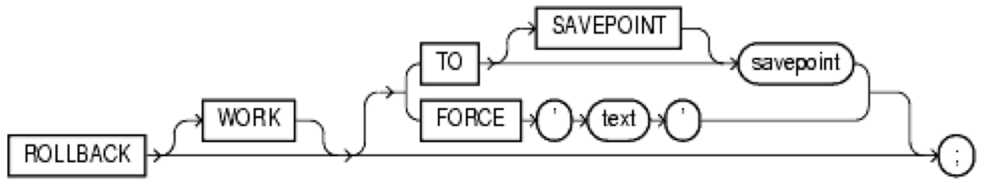
- Transaction control involves using the following statements:
  - The **COMMIT** statement ends the current transaction and makes all changes performed in the transaction permanent. COMMIT also erases all savepoints in the transaction and releases transaction locks.
  - The **ROLLBACK** statement reverses the work done in the current transaction; it causes all data changes since the last **COMMIT** or **ROLLBACK** to be discarded. The **ROLLBACK** TO **SAVEPOINT** statement undoes the changes since the last savepoint but does not end the entire transaction.
  - The S**AVEPOINT** statement identifies a point in a transaction to which you can later rollback.

Transaction ORACLE SQL









© Oracle

Oracle Doc: https://docs.oracle.com/database/121/CNCPT/transact.htm#CNCPT1121

Database Backup and Recovery

Introduction

- DBMS must provide tools to avoid or remedy failures.
- For any type of failure, the administrator must ensure that after an updade, the database is in a consistent state.
- A recovery system consists of restoring the DB to a state that is known to be correct, after any failure that has left it in an incorrect or at least suspicious state.

Introduction

- A DB administrator must to devise, implement, and manage a backup and recovery strategy.
- The purpose of a backup and recovery strategy is to protect the database against data loss and reconstruct the database after data loss.
- Backup administration tasks:
  - Planning and testing responses to different kinds of failures
  - Configuring the database environment for backup and recovery
  - Setting up a backup schedule
  - Monitoring the backup and recovery environment
  - Troubleshooting backup problems
  - Recovering from data loss if the need arises

Backup and Recovery

- Oracle provides support to automatic failure recovery, recovery processes vary depending on the type of failure that occurred and the structures affected.

- **Recovery Structures**
  - Redo Log files
  - Control files
  - Rollback Segments
  - Backups

- **Failure types**
  - Media Failures
  - User Errors
  - Application Errors

Oracle doc: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/bradv/introduction-backup-recovery.html#GUID-9997EF87-B293-44D4-92F3-DD938E79170D

Backup and Recovery Failure types

- The following problems typically require DBA intervention and data recovery:
  - **Media Failures:** They are a physical problem with a disk that causes a failure of a read from or write to a disk file that is required to run the database. The appropriate recovery technique depends on the files affected and the types of backup available.
  - **User Errors:** User errors occur when, either due to an error in application logic or a manual mistake, data in a database is changed or deleted incorrectly. The appropriate recovery technique is a combination of the management of privileges and a backup strategy.
  - **Application Errors:** A software malfunction can corrupt data blocks. In physical corruption, the database does not recognize the block. If the corruption is not extensive, the DB can be repaired easily with block media recovery.

Oracle Recovery Structures Redo Log files

- Files that **store the changes produced in the DB.**
- The redo log files are organized in groups written in a circular way, therefore, the saved information is periodically deleted by default.
- These files can be applied in a backup copy to **guide and reproduce all the modifications produced between the backup and an incident that has damaged the DB.**
  - The DB can be recovered from a failure of one or more transactions by undoing or redoing the operations individually, transaction by transaction, from the redo log file.
- For this, you must keep the redo log files, **ARCHIVELOG mode.**

Oracle Doc: https://docs.oracle.com/database/121/ADMIN/onlineredo.htm#ADMIN007

Oracle Recovery Structures Control file

- Every Oracle Database has a **control file,** which is a small binary file that **records the physical structure of the database.**
- The control file includes:
  - The database name
  - Names and locations of associated data files and redo log files
  - The timestamp of the database creation
  - The current log sequence number
  - Checkpoint information
- Without the control file, the database cannot be mounted and recovery is difficult.
- **The control file guides the recovery process**

Oracle doc: https://docs.oracle.com/database/121/ADMIN/control.htm#ADMIN006

Oracle Recovery Structures Rollback Segment

- Rollback Segment is an object that Oracle Database uses to store data necessary to reverse, or undo, changes made by transactions.
- Rollback segments store transactions that have not yet been validated.

Oracle doc:
https://docs.oracle.com/database/121/SQLRF/statements_6015.htm#SQLRF01312

Oracle Recovery Structures Backup

- The primary task of the backup administrator is making and monitoring backups for data protection.
- A **backup** is a copy of data of a database that you can use to reconstruct data.
- A backup can be either a **physical backup** or a **logical backup.**

Oracle Recovery Structures Backup

- **Physical backups**
  - Copies of physical database files and storing database information to some other location, whether on disk or some offline storage such as tape.
  - Types
    - Cold (offline) backup: the DB must be stopped in normal mode and copy the files on which it sits. Once the copy is made, it can be restarted.
    - Hot (online) backup: The copy is made while the DB is open and running in ARCHIVELOG mode. It consists of copying all the files corresponding to a given tablespace, for all the tablespaces in the DB.
- **Logical backups**
  - Contain logical data (for example, tables and stored procedures)

Oracle Recovery Structures Backup

- Recovery strategy:
  - **Physical backups are the foundation** of any sound backup and recovery strategy.
  - **Logical backups used to supplement physical backups.** Logical backups are not sufficient protection against data loss without physical backups.
- Tools:
  - **Physical backups:**
    - Oracle Recovery Manager (RMAN) utility.
    - Operating system utilities.(user-managed backup)
  - **Logical backups**
    - Extract with an Oracle utility (Export utility) and stored in a binary file.

Backup and Recovery Strategies

- In general, the typology of failures is:
  - Local failures to the transaction, which are detected in the application code => Responsibility of the Programmer
  - Local failures to the transaction, which are not detected by the application code => The DBMS forces Rollback of the Transaction
  - System failures that do not damage the DB => Hot recovery process: the log file is consulted to undo and redo transactions.
  - System failures that do damage the DB => Emergency recovery: physical backup + recovery based on the log file.
  - Fatal Error (Loss of log file (s)) => Cold recovery: Backup. Loss of Last Transactions

Oracle Recovery Manager (RMAN)

- Recovery Management Utility - Simplifies backup, restore, and recovery processes.
- Usage: in Command Line Mode or in GUI Mode.
- It can be used with or without a retrieval catalog, but the use of a catalog provides greater functionality
  - Repository (Recovery Catalog) with information on:
    - Backup, data and redo log files
    - Data file copies
    - Archived redo log and copies
    - Physical diagram of the BD
    - Stored scripts (sequences of statements)

Oracle doc: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/bradv/getting-started-rman.html#GUID-871FF5B2-C82B-462E-8182-FA28CF7B3E3B

Oracle Recovery Manager (RMAN)

- Backup created through RMAN can be:
  - **Image copies**: exact duplicate of a datafile, control file, or archived log.
    - restore them as-is without performing additional processing by using either operating system utilities(user-managed backup) or RMAN.
  - **Backup sets:** backup in a proprietary format that consists of one or more physical files called backup pieces.
    - contain more than one database file, and it can also be backed up using special processing

Starting Up and Shutting Down a database

- **Starting Up :** It requires three steps and combines the creation of an instance and the start of the DB:
  1. Start an instance:
     ```
     connect internal
     startup nomount
     ```
  2. Mount a DB : `alter database mount`
  3. Open a DB: `alter database open`
- Startup alternatives: with a single command (startup), only for the DBA (startup restrict), forced startup (startup force)

Starting Up and Shutting Down a database

- **Shutting Down:** There are three ways to stop a DB
  - **Normal:** In waiting state because there are users still connected. It does not allow new connections. Recovery is not performed during a subsequent boot.

```
shutdown
```

  - **Immediate:** The current connections are completed. Recovery is not performed during subsequent boot.

```
shutdown immediate
```

  - **Aborting:** Drastic stop, does not wait for current connections to complete. Recovery is required during a subsequent boot.

```
shutdown abort
```

References

- Database Fundamentals of Database Systems by Elmasri, Navathe7th ed. 2017
- Connolly, Thomas M, Begg , Carolyn E. Database systems: a practical approach to design, implementation, and management. Addison Wesley. 2015
- Oracle Database Backup and Recovery User's Guide https://docs.oracle.com/database/121/BRADV/toc.htm

Bibliography

- Elmasri Database Fundamentals of Database Systems by Elmasri, Navathe 7th ed. 2017
- Connolly, Thomas M, Begg, Carolyn E. Database systems: a practical approach to design, implementation, and management. Addison Wesley. 2015