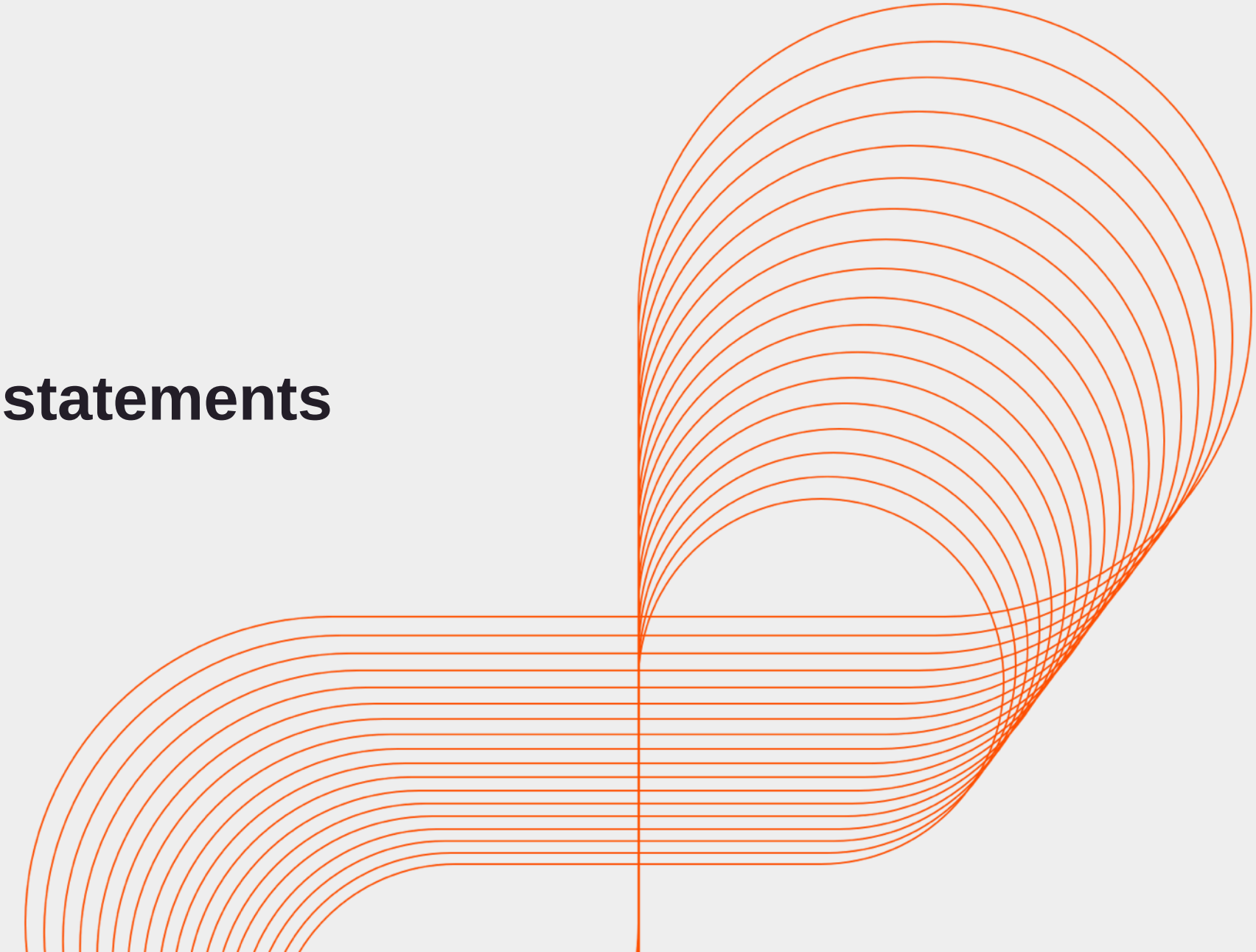
Persistent

# NUGGET 8:
# TCL and DCL statements

Persistent University



## Key Learning Points

1. Database Transaction
2. Transaction Control Statements
3. Grant and Revoke
4. Roles and Privileges

# Database Transaction

- Database Transaction consists of one of the following
  - Any number of DML statements that can be treated as single entity or logical unit of work.
  - One DDL statement or one DCL statement

- A database transaction begins with first DML statement and ends with one of the following events
  - A COMMIT or ROLLBACK statement is issued
  - DDL statement or DCL statement executes (auto commit)
  - The user exits iSQL*plus
  - A machine fails or system crashes

- IMPLICIT Transaction Processing: An automatic COMMIT happens in case any DDL or DCL statement is issued OR Normal exit from sql*plus. An automatic ROLLBACK happens under an abnormal termination of SQL*Plus or system failure.

# Transaction control statements

Below are the Explicit Transaction Control Statements:

COMMIT

ROLLBACK

SAVEPOINT

- COMMIT: Ends the current transaction by making all the pending changes permanent.
- ROLLBACK: Ends the current transaction by discarding all the pending changes.
- SAVEPOINT name: Marks a savepoint within the transaction.
- ROLLBACK to SAVEPOINT name: It will rollback the current transaction to the specified savepoint, thereby discarding any changes and/or savepoints created after the savepoint upto which you are rolling back. Savepoints are logical. There is no way to list the savepoints created.

# Transaction control statements

- State of the data before COMMIT or ROLLBACK
  - The previous state of the data can be recovered.
  - The current user can review the results of the DML operations by using the SELECT statement.
  - Other users can not view the results of the DML statements by current user.
  - The affected rows are locked; other users can not change the data within the affected rows.
- State of the Data after COMMIT
  - Data changes are made permanent to the database.
  - The previous state of the data is permanently lost.
  - All users can view the results. Locks on the affected rows are released; those rows are available for other users to manipulate.
  - All savepoints are erased.
- State of the Data after ROLLBACK
  - Data changes are undone
  - Previous state of the data is restored
  - Locks on the affected rows are released

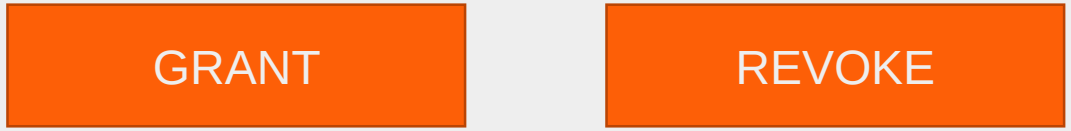
# Transaction control statements

- Example: Execute all the below statements on the sample data given

```
CREATE table new_emp (employeeid NUMBER, name VARCHAR2 (30));
INSERT INTO new_emp SELECT employeeid, lastname from employee;
SAVEPOINT s1;
UPDATE new_emp set name = UPPER (name);
SAVEPOINT s2;
Delete from new_emp;
ROLLBACK to s2;
select * from new_emp;
Delete from new_emp where employeeid =1;
UPDATE new_emp set name = 'James';
ROLLBACK to s2;
UPDATE new_emp set name = 'James' WHERE employeeid =1;
COMMIT;
select * from new_emp;
```

# Data Control Language

- DCL is abbreviation of **Data Control Language**.
- DCL Statement is used for securing the database.
- DCL controls access to the data and to the database.
- Database Administrator or owner of the database object can provide/remove privileges on a database object.
- DCL is used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it.
- Below are the DCL statements.




  - GRANT: to grant a privilege to a user
  - REVOKE: to revoke (remove) a privilege from a user

# GRANT

- **GRANT** is a command used to provide access or privileges on the database objects to the users.
- A **Privilege** is a right to execute a particular type of SQL statement or to access another user's object.
- DBA is a high-level user with the ability to grant users access to the database and its objects.
- There are two distinct categories of privileges:
  - System privileges
  - 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.
- Below are few examples of system privileges:
  - CREATE ANY TABLE
  - ALTER ANY TABLE
  - CREATE ANY INDEX
  - CREATE ANY VIEW
  - DROP ANY PROCEDURE etc
- Only two types of users can grant system privileges to other users or revoke those privileges from them:
  - Users with the system privilege GRANT ANY PRIVILEGE.
  - Users who were granted a specific system privilege with the ADMIN OPTION.

# System Privileges

- Syntax to grant system privileges :

```
GRANT privilege_name | role TO {user_name |PUBLIC |role_name}
[WITH ADMIN OPTION];
```

In the syntax :

- privilege_name is the access right or privilege granted to the user. Some of the access rights are ALTER TABLE, DROP USER.
- role is a set of privileges grouped together which can be granted to other user or role.
- user_name is the name of the user to whom an access right is being granted.
- PUBLIC is used to grant access rights to all users.
- Role_name is a set of privileges grouped together to which a role or privilege can be granted.
- WITH ADMIN OPTION enables grantee to:
  - Grant the privilege to another user or role
  - Revoke the privilege or role from another user or role

## System Privileges

- Example:

```
GRANT create trigger TO user1 WITH ADMIN OPTION;
```

On executing this query user user1 not only gets create trigger system privilege but can also grant, revoke the create trigger privilege to and from any user and roles.

# Object Privileges

- **Object privilege** is a privilege or right to perform a particular action on a specific schema object:
  - Table
  - View
  - Sequence
  - Procedure
  - Function
  - Package
- Each type of object has different privileges associated with it.
- Examples of object privileges:

| | |
|---|---|
| - ALTER | - DELETE |
| - EXECUTE | - INDEX |
| - INSERT | - REFERENCES |
| - SELECT | - UPDATE |

# Object Privileges

- Syntax to grant object privileges :

```
GRANT privilege_name
ON object_name
TO {user_name |PUBLIC |role_name}
[WITH GRANT OPTION];
```

- In the syntax :
  - privilege_name is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
  - object_name is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
  - user_name is the name of the user to whom an access right is being granted.
  - PUBLIC is used to grant access rights to all users.
  - Role_name is a set of privileges grouped together.
  - WITH GRANT OPTION allows a user to grant access rights to other users.

# Object Privileges

- Example:

```
GRANT SELECT ON User1.employees
TO User2 WITH GRANT OPTION;
```

User2 receives the SELECT privilege on User1's employees table and can also grant that privilege to other system users.

- Who Can Grant Object Privileges?
  - Owner of an Object
  - the user who got the object privileges from the owner of the object with the GRANT OPTION
  - a user with the GRANT ANY OBJECT PRIVILEGE

  Otherwise, the grantee can use the privilege, but cannot grant it to other users.

# Object Privileges

- Example1 : User1 is the owner of table employees. User2 is another user in same database seeking access on employees. Grantor user1 can issue below DCL commands

```
GRANT SELECT ON employees TO user2;
```

Above privilege will give only SELLECT access on table employees to user user2.

- Example2 :

```
GRANT UPDATE ON employees TO user2 WITH GRANT OPTION;
```

: By executing the above query user user2 will get update privilege on employees table as well as User2 can now grant and revoke the same privilege on same object to and from any other user or role.

# REVOKE

- The **REVOKE** command removes or un-grants privileges on the database objects from the users.
- Use the REVOKE statement to:
  - Revoke system privileges from users and roles
  - Revoke roles from users and roles
  - Revoke object privileges for a particular object from users and roles

# REVOKE

- Syntax to revoke object privileges:

```
REVOKE privilege_name
ON object_name
FROM {user_name |PUBLIC |role_name}
```

- In the syntax :
  - privilege_name is the access right or privilege needs to be revoked from the user.
  - object_name is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
  - user_name is the name of the user.
  - PUBLIC is used to revoke access rights from all users.
  - Role_name is a set of privileges grouped together.
- Example: Below command will REVOKE a SELECT privilege on table employees from user user2.

```
REVOKE SELECT ON employees FROM user2;
```

# REVOKE

- Syntax to revoke system privileges:

  REVOKE privilege_name

  FROM {user_name |PUBLIC |role_name}

- In the syntax
  - privilege_name is the access right or privilege needs to be revoked from the user.
  - user_name is the name of the user.
  - PUBLIC is used to revoke access rights from all users.
  - Role_name is a set of privileges grouped together

- Example: Below command will REVOKE Create any table and Create any view privileges from user2.

  REVOKE Create any table, Create any View FROM user2;

Note: The REVOKE statement can revoke only privileges and roles that were previously granted with a GRANT statement.

# Cascading Revoke

- **Cascading revoke** : Consider there are 3 users u1, u2, u3. User u1 has privilege CREATE TABLE.

  - Connect with user u1 and create table table1.

    Grant select on table1 to u2 with grant option;
  - Connect with user u2.

    Grant select on u1.table1 to u3;
  - Connect with user u1.

    Select * from user_tab_privs;

```
GRANTEE OWNER TABLE_NAME GRANTOR PRIVILEGE GRANTABLE HIE
-------------------------------------------------------------------------------------------------------------
U3      U1    TABLE1     U2     SELECT            NO       NO
U2      U1    TABLE1     U1     SELECT            YES      NO
```

    Revoke select on table1 from u2;

    Select * from user_tab_privs;

  - no rows selected

# Cascading Revoke

- As you can see, although u1 revoked the select privilege only from user u2, select privilege given to user u3 by u2 user also got revoked, because a "Cascading Revoke" occurred.
- In summary, any privilege given to any user(u3) by another grantor user(u2) having this privilege "WITH GRANT OPTION" will automatically be revoked when this privilege revoked by object owner(u1) from his grantor user(u2).
- Cascading of Revoke will not happen in case of system privileges granted using WITH ADMIN OPTION

# Role

- A **role** is a set or group of privileges that can be granted to users or another role.
- Roles are created by users (usually administrators) to facilitate the granting and revoking of multiple privileges or roles to users.
- A user can have access to several roles, and several users can be assigned the same role.
- 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.
- A new role is initially empty. You add privileges to a role with the GRANT statement. A role contains all privileges granted to the role and all privileges of other roles granted to it.

# Role

- Advantages of Roles:
  - Rather than assigning privileges one at a time directly to a user, you can create a role, assign privileges to that role, and then grant that role to multiple users and roles.
  - When you add or delete a privilege from a role, all users and roles that are assigned the particular role automatically receive or lose that privilege.
  - You can assign multiple roles to a user or role.
  - You can assign a password to a role.

# Role

- Syntax: CREATE ROLE role_name
  [ NOT IDENTIFIED |
  IDENTIFIED {BY password | USING [schema.] package | EXTERNALLY | GLOBALLY }] ;
- In the syntax:
  - NOT IDENTIFIED means the role is immediately enabled. No password is required to enable the role.
  - IDENTIFIED means a user must be authorized by a specified method before the role is enabled. There are 4 ways of authenticating a role.
    - BY password means a user must supply a password to enable the role.
    - USING package means you are creating an application role - a role that is enabled only by applications using an authorized package.
    - EXTERNALLY means a user must be authorized by an external service to enable the role. An external service can be an operating system or third-party service.
    - GLOBALLY means a user must be authorized by the enterprise directory service to enable the role.
- Example:
  - CREATE ROLE role1 IDENTIFIED BY password123;

# Role

- Grant privileges to Role:
  - Grant SELECT object privilege on employees table of user user1 to the role role1.

    GRANT SELECT ON user1.employees TO role1;
  - Grant a Role1 to another Role role2

    GRANT role1 TO role2;

- Revoke privileges from Role:
  - Revoke SELECT object privilege on employees table of user user1 to the role role1.

    REVOKE select ON user1.employees FROM role1;

# Role

- Examples with multiple roles or users.

  Consider there are two users u1, u2. There are two roles role1, role2.

  Let’s grant privileges to roles:

  GRANT CREATE TABLE, CREATE SESSION TO role1;

  GRANT CREATE SEQUENCE, CREATE SYNONYM to role2;

  A user can have access to several roles.

  GRANT role1, role2 TO u1;

  select * from DBA_ROLE_PRIVS where grantee ='U1’;

| GRANTEE | GRANTED_ROLE | ADMIN_OPTION | DEFAULT_ROLE |
|---|---|---|---|
| U1 | ROLE1 | NO | YES |
| U1 | ROLE2 | NO | YES |

  Here we can see that two roles namely role1 and role2 are granted to u1.

# Role

Several users can be assigned the same role.

GRANT role1 TO u2;

select * from DBA_ROLE_PRIVS where grantee ='U2';

| GRANTEE | GRANTED_ROLE | ADMIN_OPTION | DEFAULT_ROLE |
|---|---|---|---|
| U2 | ROLE1 | NO | YES |

Role1 was previously granted to U1 and now it has been granted to u2 also. i.e. same role can be granted to several users.

## Enable / Disable Role

To enable or disable a role for a current session, you can use the SET ROLE statement.

- Syntax: SET ROLE

  ( role_name [ IDENTIFIED BY password ] | ALL [EXCEPT role1, role2, ... ] | NONE );

- In the syntax:
  - IDENTIFIED BY password specifies the password for the role to enable it. If the role does not have a password, this phrase can be omitted.
  - ALL specifies all roles should be enabled for this current session, except those listed in EXCEPT.
  - NONE disables all roles for the current session (including all default roles).

- Examples:

  SET ROLE role1 IDENTIFIED BY test123;

  This example would enable the role called role1 with a password of test123.

  SET ROLE NONE;

  Disables all roles granted to you for the current session.

# Dropping a role

Dropping a role has the effect of removing the role from the database dictionary.

- Syntax:

```
DROP ROLE role_name;
```

- Example:

```
DROP ROLE role1;
```

# Reference Material: Sites

http://www.oracle-dba-online.com/sql/commit_rollback_savepoint.htm

http://www.techonthenet.com/oracle/roles.php

https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_6012.htm

http://www.rebellionrider.com/user-privileges/create-role-in-oracle-database.htm#.Vx7-xFZ97cs

# Session 8: Summary

With this we have come to an end of our 8th session where we discussed about

- Concept of Database Transaction.
- TCL statements in detail.
- DCL statements in detail.
- Concept of Roles and Privileges

Persistent

# Thank you!

Persistent University