Chapter 14: System and Object Privileges

Types of Privileges

1. System Privilege

- Right to perform a task in the database (e.g., `CREATE TABLE`, `CREATE USER`)

2. Object Privilege

- Right to perform operations on a specific object (e.g., `SELECT`, `UPDATE` on table `BENEFITS`)

3. Role

- A bundle of system/object privileges that can be granted together

Common System Privileges

Privilege	Description
`CREATE SESSION`	Connect to the database
`CREATE TABLE`	Create tables, includes `ALTER`, `DROP`
`CREATE VIEW`	Create views
`CREATE SEQUENCE`	Create sequences
`CREATE SYNONYM`	Create synonyms
`CREATE ROLE`	Create roles
`CREATE ANY TABLE`	Create tables in **any** user account
`GRANT ANY PRIVILEGE`	Grant any system privilege to any user
`GRANT ANY OBJECT PRIVILEGE`	Grant any object privilege for any object

System vs Object Privileges

- **System privilege**: Allows general action (e.g., create a table)
- **Object privilege**: Grants access to a **specific** object (e.g., modify user `EUNICE.BENEFITS`)
- **Analogy**:
  - Driver's license = system privilege
  - Car key = object privilege

Key SQL Statements

User Management

```sql
-- Create User
CREATE USER username IDENTIFIED BY password;

-- Alter User Password
ALTER USER username IDENTIFIED BY new_password;

-- Drop User
DROP USER username;
DROP USER username CASCADE; -- deletes user and all their objects
```

Privilege Management

```sql
-- Grant Privileges
GRANT privilege TO user;
GRANT privilege TO user WITH ADMIN OPTION;

-- Revoke Privileges
REVOKE privilege FROM user;

-- Grant All Privileges
GRANT ALL PRIVILEGES TO user;

-- Revoke All Privileges
REVOKE ALL PRIVILEGES FROM user;

-- Grant to All Users (PUBLIC)
GRANT privilege TO PUBLIC;
REVOKE privilege FROM PUBLIC;
```

WITH ADMIN OPTION

- Allows the grantee to **grant the privilege to others**
- Cascades until REVOKED from the latest grantee manually

ANY Keyword

- Extends privilege across all users' objects

- Example: `CREATE ANY TABLE` = create tables in **any** user schema
- **Owner** of the object is still the schema where it's created

Tablespaces (Not on exam)

For exam use:

```sql
GRANT UNLIMITED TABLESPACE TO username;
```

Example SQL Session

```sql
CONNECT SYSTEM/MANAGER
CREATE USER HAROLD IDENTIFIED BY LLOYD;
GRANT CREATE SESSION TO HAROLD;
GRANT UNLIMITED TABLESPACE TO HAROLD;
GRANT CREATE TABLE TO HAROLD;
CONNECT HAROLD/LLOYD
CREATE TABLE CLOCKTOWER (CLOCK_ID NUMBER(11));
-- fails if CREATE SEQUENCE not yet granted
CREATE SEQUENCE SEQ_CLOCK_ID;
```

🔐 Privileges and Ownership

- A user with the **CREATE TABLE** system privilege can create a table. The **creator becomes the owner**
- Table owners have full access (SELECT, INSERT, UPDATE, DELETE) without needing grants
- Other users **need explicit object privileges**, unless they have system privileges like:
    - `SELECT ANY TABLE`, `INSERT ANY TABLE`, `UPDATE ANY TABLE`, `DELETE ANY TABLE`

✅ Granting Object Privileges

- **Object privileges** apply to DML (SELECT, INSERT, UPDATE, DELETE) and some DDL (like ALTER)
- Syntax:

    ```sql
    GRANT SELECT, UPDATE ON webinars TO henry;
    ```

- **WITH GRANT OPTION**: Allows the grantee to grant the same privileges to others

```sql
GRANT SELECT ON webinars TO henry WITH GRANT OPTION;
```

- **Schema prefix** is required when referencing another user's table:

  ```sql
  SELECT * FROM lisa.webinars;
  ```

- Use **public synonyms** to avoid schema prefixes:

  ```sql
  CREATE PUBLIC SYNONYM webinars FOR lisa.webinars;
  ```

❌ Revoking Privileges

- Syntax:

  ```sql
  REVOKE SELECT, UPDATE ON webinars FROM henry;
  ```

- Revocation **cascades** to anyone who got privileges from the revoked user (if granted with `WITH GRANT OPTION`)
- `REVOKE ALL ON webinars FROM henry;` revokes all object privileges (keyword `PRIVILEGES` is optional for object privileges)

📦 ALL PRIVILEGES

- Grants or revokes **all object-level privileges** at once:

  ```sql
  GRANT ALL ON webinars TO henry;
  REVOKE ALL ON webinars FROM henry;
  ```

🔁 Dependent Privileges

- Granting a privilege on a **view** doesn't give access to the **underlying table** unless explicitly granted
- PUBLIC SYNONYM gives visibility but **not access**—privileges are still required on the object it points to

📊 Data Dictionary Views (Privileges)

View	Description
USER_SYS_PRIVS	System privileges for current user
DBA_SYS_PRIVS	System privileges for all users and roles
USER_TAB_PRIVS	Object privileges current user has granted/received
ALL_TAB_PRIVS	Object privileges across the database
DBA_TAB_PRIVS	Grants on all objects
ALL_TAB_PRIVS_RECD	Grants where user/role is the grantee
SESSION_PRIVS	Enabled privileges for current session

👥 Roles

- A **role** groups privileges and can be granted like a user
- Requires `CREATE ROLE` system privilege to create
- Can be granted **WITH ADMIN OPTION** to allow further role delegation
- Example:

  sql

  ```sql
  CREATE ROLE cruise_analyst;
  GRANT SELECT ON ships TO cruise_analyst;
  GRANT cruise_analyst TO henry;
  ```

Standard Roles (Discouraged by Oracle)

Role	Privileges
CONNECT	CREATE SESSION
RESOURCE	Various CREATE privileges (e.g., TABLE, TRIGGER)
DBA	Over 100+ system privileges

🔍 Role Privileges in Data Dictionary

View	Description
DBA_ROLES	All roles in database
DBA_ROLE_PRIVS	Roles granted to users/roles
ROLE_ROLE_PRIVS	Roles granted to other roles
ROLE_SYS_PRIVS	System privileges granted to roles
ROLE_TAB_PRIVS	Table privileges granted to roles
SESSION_ROLES	Roles enabled in current session

Certification Objective 14.03: Distinguish Between Privileges and Roles

🔑 Key Concepts

- **Role** = A named collection of privileges (can include system, object, or other roles). It doesn't hold privileges itself but grants access via the privileges it contains
- **Privileges** = Can be:
  - **System privileges** (e.g., `CREATE TABLE`)
  - **Object privileges** (e.g., `SELECT ON table_name`)

🧠 Behavior of Privileges vs. Roles

- Direct privileges granted to a user are **independent** of privileges granted via roles
- If a role is revoked, any **directly granted privileges remain** intact

🧪 Example Breakdown

```sql
01 GRANT SELECT ON INVOICES TO HENRY;
02 CREATE ROLE CRUISE_ACCOUNTANT;
03 GRANT SELECT ON INVOICES TO CRUISE_ACCOUNTANT;
04 GRANT CRUISE_ACCOUNTANT TO HENRY;
05 REVOKE CRUISE_ACCOUNTANT FROM HENRY;
```

- After line 1, HENRY has direct access
- Lines 2-4 give HENRY access via the role
- Even after revoking the role in line 5, HENRY still has `SELECT` via line 1

✅ **Direct privileges stay** even if the role is revoked

🔁 Reverse Scenario

- If the reverse happens (privilege is revoked directly, but HENRY still has access via the role), he still retains access **through the role**

❌ Full Removal of Privilege

To fully remove access, you must:

1. Revoke **direct privilege** from user
2. Revoke that **same privilege from any roles** the user has

📌 Exam Watch Tips

- To grant to **all users**: use `PUBLIC`, not `ALL`
- `GRANT ALL PRIVILEGES TO PUBLIC` is a **terrible idea**—don't do it
- **GRANT = Implicit COMMIT.** You can't ROLLBACK a GRANT
- **Privileges are lost** if a table is dropped
- If you use `FLASHBACK TABLE ... BEFORE DROP`, the privileges are recovered
- "Privileges" may refer to either system or object privileges — know the difference
- **Roles = groupings** of system and/or object privileges and/or other roles