USERS & PRIVILEGES

CIS-673, LECTURE#20

BY RAJ PATIL

AUTHORIZATION

- Permitting only certain users to access, process, or alter data.
- **Privilege:** is the right to run a particular type of SQL statement, or the right to access an object that belongs to another user, run a PL/SQL package, and so on.
- **Role:** it groups several privileges (and roles) together, so that they can be granted to and revoked from users simultaneously.
- You grant privileges to users so they can accomplish tasks required for their jobs.
- Excessive granting of unnecessary privileges can compromise security.

GRANT – 2 TYPES

- A user can receive a privilege in two ways:

- Grant privileges to users explicitly.
  - For example, you can explicitly grant to user 'Smith' the privilege to insert records into the employees table.

- Grant privileges to a role, and then grant the role to one or more users.
  - For example, you can grant the privileges to select, insert, update, and delete records from the employees table to the role named clerk, which in turn can be granted to users 'Smith and Robert'.

PRIVILEGES

- Privileges are assigned using **GRANT** command, and withdrawn with **REVOKE**.

- Two types:
  - System privileges: lets user perform actions that affect data-dictionary (meta-data).
  - Object privileges: lets user perform actions that affect the data.

SYSTEM & OBJECT PRIVILEGES

- **Common System Privileges:**
  - CREATE SESSION, CREATE TABLE, CREATE VIEW, CREATE MATERIALIZED VIEW,
  - CREATE TRIGGER, CREATE PROCEDURE, CREATE SEQUENCE

  //normally, create statements are system-privileges as they generate meta-data

- **Common Object Privileges:**

  Select, insert, update, delete, alter, execute

- To grant ALL object privileges on cities-table to user john, use the 'ALL' keyword:
  - Grant ALL on SYSTEM.cities to JOHN;

GRANTING & REVOKING

- Use "**with admin option**" so that the grantee can pass his/her *system* privilege on to a third party

- Revocation of system privilege will NOT cascade.

- Use "**with grant option**" so that the grantee can pass his/her *object* privilege on to a third party

- Revocation of object privilege will cascade all those in the chain.

CASCADING EFFECTS

- There are no cascading effects when revoking a system privilege related to DDL operations
    - This is regardless of whether the privilege was granted with or without the ADMIN OPTION.

- Revoking object privileges have cascading effects.
    - The object privilege grants propagated using the GRANT OPTION are revoked if the object privilege of a grantor is revoked.

8 GRANTING AND REVOKING PRIVILEGES ON COLUMNS

- User can GRANT INSERT, UPDATE, or REFERENCES privileges on INDIVIDUAL COLUMNS in a table.
- **For example:** GRANT INSERT (acct_no) ON accounts TO psmith;

  GRANT INSERT(ename, job) ON emp TO jfee, tsmith;

- REVOKE:
  - Although users can grant column-specific INSERT, UPDATE, and REFERENCES privileges for tables and views, they cannot selectively revoke column-specific privileges with a similar REVOKE statement.
  - Instead, the grantor must first revoke ALL the object privilege for ALL columns of a table or view, and then selectively repeat the grant of the column-specific privileges that the grantor intends to keep in effect.

9 EXAMPLE OF GRANT AND REVOKE

- GRANT CREATE SESSION, CREATE USER, CREATE TABLE, CREATE VIEW, CREATE TRIGGER, CREATE PROCEDURE

  to BOB with admin option;

- GRANT select, insert,update,delete on SYSTEM.states

  to BOB with grant option;

- REVOKE select,insert,update,delete on SYSTEM.states from BOB;

- REVOKE CREATE SESSION, CREATE USER, CREATE TABLE, CREATE VIEW, CREATE TRIGGER, CREATE PROCEDURE

  from BOB;

CREATE AND DROP USER

- Create user identified by password;
- A username can never be changed after creation. Instead, it must be dropped and recreated.
- When username is dropped all the objects in the user’s schema are also dropped.
- Cascade option:
  - If a user owns any database objects, that user can only be dropped with the DROP USER CASCADE command.
  - The Oracle DROP USER CASCADE command drops a user and all owned objects.

ELIGIBILITY TO GRANT

- To be eligible to grant a system privilege:
  - a user must be granted the system privilege **WITH ADMIN OPTION** or
  - must be granted the **GRANT ANY PRIVILEGE**.

- To be eligible to grant an object privilege:
  - User must own the object specified, or
  - The **WITH GRANT OPTION** clause was specified when the user was granted the object privilege, or
  - Must be granted the **GRANT ANY OBJECT PRIVILEGE**

- To be eligible to grant a role:
  - a user must be granted the role **WITH ADMIN OPTION** or
  - was granted the **GRANT ANY ROLE** system privilege.

ELIGIBILITY TO REVOKE

- To be eligible to revoke a system privilege:
    - Any user **with the ADMIN OPTION** for a system privilege can revoke the privilege from any other database user or role.
    - The revoker does not have to be the user that originally granted the privilege; users with **GRANT ANY PRIVILEGE** can revoke any system-privilege.
- To be eligible to revoke an object privilege:
    - You previously granted the object privilege to the user
    - You possess the **GRANT ANY OBJECT PRIVILEGE** system privilege that enables you to grant and revoke privileges on behalf of the object owner.
- To be eligible to revoke a role:
    - Any user with the **ADMIN OPTION** for a role can revoke role from any other database user or role.
    - The revoker does not have to be the user that originally granted the role. Users with **GRANT ANY ROLE** can revoke any role.

ROLES

- A role is a bundle of system and/or object privileges that can be granted and revoked as a unit. Roles share the same namespace as users.
- Example:
  - Admin role: create users, create sessions, …
  - Developer role: insert, update, delete tables, …
  - Clerk role: read tables, views, …
- A role can be granted system or object privileges.
- Each role granted to a user is either enabled or disabled.
- A role can be granted to user or other roles.