Database Privileges

Dr. Nan Wang

Database Sharing

- Databases are all about sharing data.
- A database is accessible to many users, and not every user should be allowed to do everything.
- SQL allows us to assign different types of privileges to different users.
- A user is specified by a user identifier.
  - CREATE USER username IDENTIFIED BY password
  - Create user 'nan'@'localhost' identified by 'abcd1234'

- The user that creates a database object, such as a table, is called the owner.
- The owner of an object can do just about anything he or she wants with that object, including determining the privileges of other users for that object.

Grant

You can give privileges on an object to a user with the GRANT statement:

```
GRANT {ALL PRIVILEGES | <privilege>[, <privilege>...]}
ON <database object>
TO <grantee> [, <grantee>...]
[WITH GRANT OPTION]
```

GRANT gives the specified privilege(s) on the named object to the list of identified users. Specifying the WITH GRANT OPTION allows the identified users to grant their privileges to other users.

Privilege	Permits
SELECT[(<*column list*>)]	SELECT on the specified table. If <*column list*> is specified, the user may only access values for those columns; otherwise, the user may access all columns.
INSERT[(<*column list*>)]	INSERT for new row(s) on the specified table. If <*column list*> is specified, the user may only specify values for those columns; all other columns are given the default value. If <*column list*> is not specified, the user may specify values for all columns.
UPDATE[(<*column list*>)]	UPDATE on existing rows on the specified table. If <*column list*> is specified, the user may only update values for those columns; otherwise, the user may update all columns.
DELETE	DELETE of existing rows in the specified table.
REFERENCES[(<*column list*>)]	References to columns in the specified table including foreign key and CHECK constraints. If <*column list*> is specified, the user may only define references for those columns; otherwise, the user may reference all columns.
ALL PRIVILEGES	All privilege types.

DCL 12.1 Give SELECT privileges

```
GRANT SELECT(storeid, itemid) ON ORDERS TO john, mary;
GRANT SELECT ON STORES TO john, mary;
```

DCL 12.2 Give SELECT privileges with GRANT OPTION

```
GRANT SELECT ON ORDERS TO ed WITH GRANT OPTION;
```

DCL 12.3 Give INSERT/DELETE privileges

```
GRANT INSERT(storeid, manager), DELETE ON stores TO rachel;
```

DCL 12.4 Give UPDATE privileges

```
GRANT UPDATE(address, city, state, zip) ON stores TO jeff, greg;
```

DCL 12.5 Give REFERENCE privileges

```
GRANT REFERENCES(vendorid) ON vendors TO jane;
```

DCL 12.6 Give ALL privileges

```
GRANT ALL PRIVILEGES ON stores TO jack WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON orders TO jack WITH GRANT OPTION;
...
```

Revoke

You can remove privileges on an object from a user using REVOKE:

```
REVOKE [GRANT OPTION FOR]
{ALL PRIVILEGES | <privilege>[, <privilege>...]}
ON <database object>
FROM <grantee> [, <grantee>...]
[CASCADE | RESTRICT]
```

DCL 12.7 REVOKE privileges

```
REVOKE UPDATE ON stores FROM greg;
```

- Cascade
- Restrict

Public

- We can grant a privilege to all users by assigning it to PUBLIC.
- Of course, this means we should be very careful with any privileges granted to PUBLIC.

DCL 12.8 Give PUBLIC privileges

```
GRANT SELECT(name, prices) ON menuitems TO PUBLIC;
```

DCL 12.9 REVOKE SELECT privileges

```
REVOKE SELECT(name) ON menuitems TO jane;
```

Creating a Set of Privileges Using ROLEs

- We can create a *role that represents a type of database* user and assign privileges to that role.
- Assigning a role to a user gives that user all of the privileges granted to the role.
- We can create a role using the following:
- **CREATE ROLE <role name> [WITH ADMIN OPTION]**
- If WITH ADMIN OPTION is specified, the role grantee may grant the role to others.

Create a Role and Grant Privileges

DCL 12.10 Create a ROLE and GRANT privileges

```
CREATE ROLE cashier;
GRANT INSERT ON orders TO cashier;
```

We assign a role using GRANT.

```
GRANT <role name>, [, <role name>...] TO <grantee> [, <grantee>...]
[WITH ADMIN OPTION]
```

DCL 12.11 Grant ROLE to users

```
GRANT cashier TO abe, sara;
```

Revoke a role

REVOKE [ADMIN OPTION FOR]
<role>[, <role>...]
FROM <grantee> [, <grantee>...]
[CASCADE | RESTRICT]

DCL 12.12 Revoke user ROLE

```
REVOKE cashier FROM abe;
```

We can delete a role altogether.

```
DROP ROLE <role>
```

Using Privileges and Views

- We can grant and revoke privileges to views, just like a table.

Summary

- Databases usually are accessed and manipulated by many different users. SQL allows us to control the kinds of operation each user is permitted to perform including SELECT,INSERT, UPDATE, DELETE, and even external references.
- GRANT adds new privileges to a user, whereas REVOKE removes existing privileges.
- Controlling privileges on a user-by-user basis is very tedious. SQL provides a special identifier, PUBLIC, that allows us to determine privileges available to all users.
- We can create and assign privileges to database roles. We can then grant or revoke roles to or from users and other roles.

Exercise

1. Give everyone the privilege of seeing all employee information except salary.
2. Give sue the privilege of updating the revenue of projects.
3. Give tom the privilege of adding new projects but not end dates. Allow him to grant this privilege to someone else.
4. Create a role called dept_head for tom and sue. Allow dept_head to delete departments.
5. Assuming the previous questions have been successful, revoke all of the privileges granted to tom, but allow sue to keep all of her privileges. Allow tom to keep all PUBLIC privileges.