Database Security

COSC 304 – Introduction to Database Systems

SQL Security

Security in SQL is based on ***authorization identifiers***, ***ownership***, and ***privileges***.

An authorization identifier (or user id) is associated with each user. Normally, a password is also associated with a authorization identifier. Every SQL statement performed by the DBMS is executed on behalf of some user.

The authorization identifier is used to determine which database objects the user has access to.

Whenever a user creates an object, the user is the owner of the object and initially is the only one with the ability to access it.

SQL Privileges

Privileges give users the right to perform operations on database objects. The set of privileges are:

- `SELECT` - the user can retrieve data from table
- `INSERT` - the user can insert data into table
- `UPDATE` - the user can modify data in the table
- `DELETE` - the user can delete data (rows) from the table
- `REFERENCES` - the ability to reference columns of a named table in integrity constraints
- `USAGE` - the ability to use domains, character sets, and translations (i.e. other database objects besides tables)

Notes:

- `INSERT` and `UPDATE` can be restricted to certain columns.
- When a user creates a table, they become the owner and have full privileges on the table.

SQL GRANT Command

The **GRANT** command gives privileges on database objects to users.

```
GRANT {privilegeList | ALL [PRIVILEGES]}
  ON ObjectName
  TO {AuthorizationIdList | PUBLIC}
  [WITH GRANT OPTION]
```

The privilege list is one or more of the following privileges:

```
SELECT [(columnName [,...]]
DELETE
INSERT [(columnName [,...]]
UPDATE [(columnName [,...]]
REFERENCES [(columnName [,...]]
USAGE
```

SQL GRANT Command (2)

The `ALL PRIVILEGES` keyword grants all privileges to the user except the ability to grant privileges to other users.

The `PUBLIC` keyword grants access to all users (present and future) of the database.

The `WITH GRANT OPTION` allows users to grant privileges to other users. A user can only grant privileges that they themselves hold.

GRANT Examples

Allow all users to query the `dept` relation:

```
GRANT SELECT ON dept TO PUBLIC;
```

Only allow users `Manager` and `Director` to access and change `Salary` in `emp`:

```
GRANT SELECT, UPDATE(salary) ON emp TO Manager,Director;
```

Allow the `Director` full access to `proj` and the ability to grant privileges to other users:

```
GRANT ALL PRIVILEGES ON proj TO Director
    WITH GRANT OPTION;
```

Required Privileges Example

What privileges are required for this statement:

```
UPDATE emp SET salary=salary*1.1
WHERE eno IN (
     SELECT eno FROM workson WHERE hours > 30)
```

Answer:

```
SELECT on emp
UPDATE(salary) on emp
SELECT on workson
```

Required Privileges Question

Question: What are the required privileges for this statement?

```
DELETE FROM dept WHERE dno NOT IN
    (SELECT dno FROM workson)
```

A) DELETE, SELECT

B) DELETE on dept, DELETE on workson

C) DELETE on dept, SELECT on workson

D) DELETE on dept

E) DELETE on dept, SELECT on workson, SELECT on dept

Required Privileges Question (2)

Question: What are the required privileges for this statement?

```
INSERT INTO workson (eno,pno) VALUES ('E5','P5')
```

A) INSERT on workson

B) INSERT

C) INSERT, SELECT

D) INSERT on workson, UPDATE on workson

E) none

GRANT Question

Question: **True or False:** A user `WITH GRANT OPTION` can grant a privilege that they do not hold themselves.

A) true

B) false

GRANT Question (2)

Question: **True or False:** A users may be granted the same privilege on the same object from multiple users.

A) true

B) false

SQL REVOKE Command

The REVOKE command is used to remove privileges on database objects from users.

```
REVOKE [GRANT OPTION FOR]{privilegeList | ALL [PRIVILEGES]}
  ON ObjectName
  FROM {AuthorizationIdList | PUBLIC} {RESTRICT|CASCADE}
```

Notes:

- ALL PRIVILEGES removes all privileges on object.
- GRANT OPTION FOR removes the ability for users to pass privileges on to other users (not the privileges themselves).
- RESTRICT - REVOKE fails if privilege has been passed to other users.
- CASCADE - removes any privileges and objects created using the revoked privileges including those passed on to others.

SQL REVOKE Command (2)

Privileges are granted to an object to a user *from* a specific user. If a user then revokes their granting of privileges, that only applies to that user.

Example:

- User A grants all privileges to user B on table T.
- User B grants all privileges to user C on table T.
- User E grants `SELECT` privilege to user C on table T.
- User C grants all privileges to user D on table T.
- User A revokes all privileges on table T from B (using `cascade`).
- This causes all privileges to be removed for user C as well, except the `SELECT` privilege which was granted by user E.
- User D now has only the `SELECT` privilege as well.

REVOKE Examples

Do not allow public (general) users to query the `dept` relation:

```
REVOKE SELECT ON dept FROM PUBLIC;
```

Remove all privileges from user `Joe` on `emp` table:

```
REVOKE ALL PRIVILEGES ON emp FROM Joe;
```

REVOKE Question

Question: User *A* executes:

```
GRANT SELECT, UPDATE ON T TO B WITH GRANT OPTION;
```

then User B executes:

```
GRANT SELECT, UPDATE ON T TO C;
```

then User A executes:

```
REVOKE GRANT OPTION FOR SELECT, UPDATE ON T FROM B;
```

True or False: User *C* loses `SELECT` on *T*.

A) true

B) false

SQL Security and Views

Views are used to provide security and access restriction to certain database objects.

Example: Consider the `emp` relation. We want the user `Staff` to have only query access but not be able to see users `birthdate` and `salary`. How do we accomplish this?

Step #1: Create a view on the `emp` relation.

```
CREATE VIEW EmpView AS
SELECT eno, ename, title, supereno, dno
FROM emp;
```

SQL Security and Views (2)

Step #2: Provide `SELECT` privilege to staff.

```
GRANT SELECT ON EmpView TO Staff;
```

Step #3: `REVOKE` privileges on base relation (`emp`)

```
REVOKE SELECT ON Emp From Staff;
```

Privileges on Views Question

Question: Table *T* is part of view definition query for view *V*. If user *A* has `SELECT` on *V*, but not on *T*, can user *A* run a query on *V* and see results?

A) yes

B) no

Security Practice Questions

Use `GRANT` and `REVOKE` to provide the desired access for each of these cases. You may also need views. Assume that you are the DBA who has full privileges (owner) of all objects currently in the database.

1) Allow all users read-only access to the `dept` relation.

2) Allow the user `Accounting` to read `emp` records and update the `salary` of `emp`.

3) Allow user `Davis` all privileges on a view containing the employees in Dept `D2` including the ability to grant privileges.

4) Allow user `Smith` to only see their employee record (`eno='E3'`) and `WorksOn` information. (need a view)

5) `Davis` is replaced as head of Dept. `D2`, so only leave him query access to employees in `D2`.

Conclusion

SQL security is enforced using `GRANT`/`REVOKE`.

Views are useful for security as users can be given privileges to access a view but not the entire relation.

Objectives

- Use `GRANT`/`REVOKE` syntax
- List the types of privileges and know when to use them
- Given a SQL command, explain what privileges are required for it to execute
- Explain how views are useful for security


UBC
THE UNIVERSITY OF BRITISH COLUMBIA