Principles of Databases

Authorisation in SQL

David Sinclair

Privileges

- The *privilege* mechanism is used in SQL:
  - to ensure users can only see data that they are allowed to see;
  - to ensure users can only modify data that they are allowed to modify; and
  - to protect the data, at the database level, from malicious users.
    - Privileges do not protect against programming or system issues, such as SQL injection errors.

- A user is assigned a set of privileges and these privileges limit how that user can operate on the data.

Privileges (2)

- Privileges in SQL are *fine-grained* and specify how a user can interact with a given relation (table) and even limit which attributes (columns) in relation the user can access or modify.
  - `SELECT ON R` or `SELECT (A1,A2,...,Am) ON R`
    - Read privileges
  - `INSERT ON R` or `INSERT (A1,A2,...,Am) ON R`
    - Write privileges
  - `UPDATE ON R` or `UPDATE (A1,A2,...,Am) ON R`
    - Modify privileges
  - `DELETE ON R`
    - Delete privileges
    - Only a complete relation (table) can be deleted, not individual attributes.

Example 1

Consider the query

```
UPDATE Apply
SET decision = 'Y'
WHERE sID IN (SELECT sID FROM Student
                         WHERE score >= 70.0);
```

What are the minimum set of *privileges* required?

- On the `Apply` table we need: `UPDATE(decision)`, `SELECT(sID)`
- On the `Student` table we need: `SELECT(sID,score)`

Example 2

Consider the query

```
DELETE FROM Student
WHERE sID NOT IN (SELECT sID FROM Apply);
```

What are the minimum set of *privileges* required?

- On the `Student` table we need: `DELETE`, `SELECT(sID)`
- On the `Apply` table we need: `SELECT(sID)`

Views and Authorisation

- A major use of *Views* in SQL is *Authorisation*.
- In our `Country` database, consider the situation where you only want to allow a user to read records where the `Person` comes from `Ireland`.
- To limit the part of the database that will be affected by the change in privileges, we create a view that only includes people from Ireland.

```
CREATE VIEW Irish AS
   SELECT * FROM Person
   WHERE countryID IN
     (SELECT countryID FROM Country
       WHERE Name = 'Ireland');
```

- To allow a user to read the records only from Ireland, we would grant the `SELECT` privileges on the relation/view called `Irish`.

```
SELECT * ON Irish
```

Views and Authorisation (2)

- To give a user the ability to delete records relating to Italians, we create the following view.

```
CREATE VIEW Italian AS
   SELECT * FROM Person
   WHERE countryID IN
     (SELECT countryID FROM Country
       WHERE Name = 'Italy');
```

- The appropriate privilege is:
  `DELETE ON Italian`
- It is **important** to ensure that this view is updatable/modifiable.

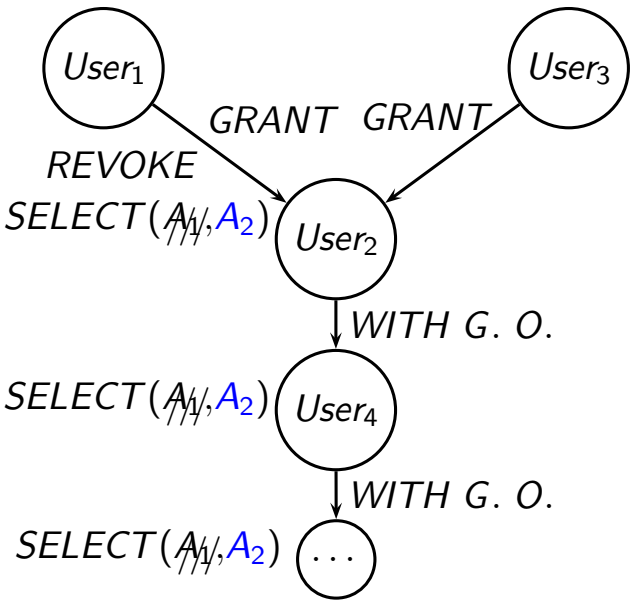
Granting Privileges

- The user that creates a relation is its *owner*.
- The *owner* of a relation has all privileges on that relation and can grant privileges to other users.
- The SQL syntax for granting privileges is:
  `GRANT` *privileges* `ON` *table* `TO` *users*
  `[WITH GRANT OPTION];`
- *privileges* can be a comma separated list of privileges.
- *users* can be a comma separated list of user, including a special user called `public` that means everyone.
- The optional `WITH GRANT OPTION` command enables the user receiving the privileges to grant these privileges, or a lesser set of these privileges, to other users.
  - `UPDATE(X,Z)` is a lesser set than `UPDATE(X,Y,Z)`.

Revoking Privileges

- The SQL syntax for revoking privileges is:
  `REVOKE` *privileges* `ON` *table* `FROM` *users*
  `[CASCADE | RESTRICT];`
- With the `CASCADE` option if some of user A's privileges are being revoked, then the privileges A granted from the privileges being revoked, are also revoked transitively, unless also granted from another source.
- With the `RESTRICT` option, disallow the REVOKE if there are any other privileges that depend on the privileges being revoked. This is the default option.

Revoking Privileges (2)



This is a *Grant Diagram*.

User$_1$

```
GRANT SELECT(A1,A2) ON R TO
User2 WITH GRANT OPTION;
```

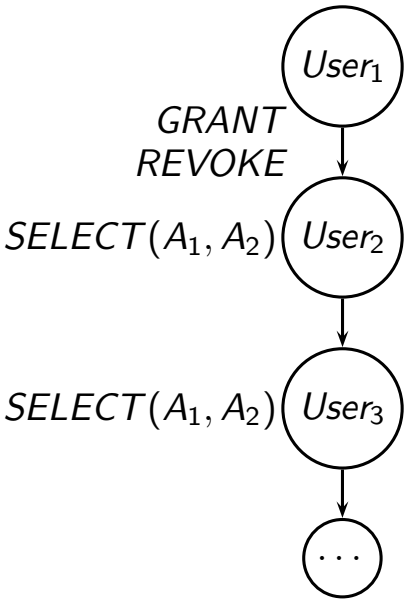
User$_3$

```
GRANT SELECT(A2) ON R TO
User2 WITH GRANT OPTION;
```

User$_1$

```
REVOKE SELECT(A1,A2) ON R
FROM User2 CASCADE;
```

Revoking Privileges (3)



$\underline{\text{User}_1}$

```
GRANT SELECT(A1,A2) ON R TO
User2 WITH GRANT OPTION;
```

$\underline{\text{User}_2}$

```
GRANT SELECT(A1,A2) ON R TO
User3;
```

$\underline{\text{User}_1}$

```
REVOKE SELECT(A1,A2) ON R
FROM User2 RESTRICT;
```

Fails as User_3's privileges depend on Users_2's privileges.