CIS4301 Notes: Database Design

Ryan Roden-Corrent

April 4, 2014

1 SQL Authoriation

Stanford Slides

- Privileges
- Grant and Revoke
- Grant Diagrams

1.1 Authoriation

- A file system identifies certain privileges on the objects in manages
- A file system tracks these privileges for users, groups, and the world
- SQL priveleges are more detailed - typically shown in grant diagram

See the Postgres Docs on GRANT
Priveleges can be granted:

- per user
- per action
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
  - REFERENCES
  - TRIGGERS
- per column

1.1.1 Roles

- Analogous to users in a filesystem
- Each role has its own set of permissions
- Permissions can be inherited from roles

Listing 1: example query

```
INSERT INTO Beers(name)
SELECT beer FROM Sells
WHERE NOT EXISTS
  (SELECT * FROM Beers
  WHERE name = beer);
```

Priveleges Required for Listing 1:

- SELECT on Sells and Beers
- INSERT on Beers or Beers.name

1.2 Permissions in pgsql

- $\backslash du$ to list roles
- $\backslash dp$ to list permissions per table
  - Letters represent permissions
  - Look the up on Postgres docs
- Superuser

1.3 Views as access control

Suppose there is a relation **Emps(name,addr,salary)**

Listing 2: view to control access

```
CREATE VIEW SafeEmps AS
SELECT name, addr FROM Emps;
```

In Listing 2, a view is created to allow access to employee name and address while hiding their salary. This view is **updateable**. A complex join can prevent a view from being updateable if it does not preserve all information from the original tables.
As an example, consider a view that has the same employees but with all of the vowels removed from their names. There is no longer a mapping from the modified employee names back to their original names (for example, the names Josh and Joshe would both map to Jsh in the view).

1.4 The GRANT Statement

Listing 3: grant format

```
GRANT list_of_priveleges
ON relation_or_object
TO list_of_authorization_ids;
```

A GRANT option like this could be added to the end of a script to it is run every time.

1.5 The REVOKE Statement

Listing 4: revoke format

```
REVKE list_of_priveleges
ON relation_or_object
FROM list_of_authorization_ids;
```

1.5.1 REVOKE Options

CASCADE any grants made by revokee are no longer in force, no matter how far privilege passed

RESTRICT if the privilege was passed to others, the REVOKE fails, as a warning that something else should be done to chase it down

Suppose Rick was granted **UPDATE** on Wins table. Rick grants this privilege to Billy, who grants it to Anthony. The path of permission passage is tracked in the database. **REVOKING** this privilege from Billy with the **CASCADE** option would automatically revoke Anthony's privilege too, whereas the **RESTRICT** option would cause the operation to fail until Anthony's privilege was revoked too.
However, if Anthony were also granted this privilege by Rick, then he can keep the permission, as there is still a user who granted him this permission and still has it themself.