CS 1555

Lecture 14

**Access Control**

Reliability

- Ensuring data integrity

- Security: encryption, authentication, compartmentalization

- Access control: who (user/role), what (data), how (operations)

- Mandatory and discretionary

- Access permissions in the catalog

Access control

- Grant command

- GRANT <privilege list> | ALL PRIVILEGES ON <object\_names> TO <user-name-list> | <role-name-list> | PUBLIC [WITH GRANT OPTION]

- Privilege list: privilege [(attribute-list)]

- Privileges: SELECT, DELETE, INSERT, UPDATE, REFERENCE (with CREATE TABLE)

- [WITH GRANT OPTION]: can grant permissions to others

RBAC: Role-based access control

- Roles are similar to user groups

- CREATE ROLE <role\_name>; and DROP ROLE <role\_name>;

- Users can be given the privilege to be part of a role

Revoking privileges

- Revoke command is similar to the grant command

- REVOKE <privilege\_list> | ALL PRIVILEGES ON <object\_names> FROM <user-name list> | <role-name list> | PUBLIC [CASCADE | RESTRICT]

- Cascade revokes privileges from all users who got the privileges from a user in the user-name list

- Revokes only the privileges derived/granted by the user invoking the revoke statement

Access control lattice example

- DBA grants read privilege to Alex and Christine with grant option

- Alex grants read to Nathan with grand option

- Christine grants read to Nathan with grant option

- DBA revokes read from Alex with CASCADE

- Nathan still has access to read, because his privilege granted from Christine counts separately

- DBA revokes read from Christine: will be rejected because there was no CASCADE option included

**End of midterm material**

**Integrity Constraints in SQL**

Integrity constraints

- Structural: key, entity integrity, referential integrity

- Semantic: logical rules derived from the application

Structural constraints: NOT NULL, DEFAULT, PRIMARY KEY, UNIQUE, FOREIGN KEY

Referential triggered actions

- Actions if referential integrity constraint is violated

- SET NULL, CASCADE, DEFAULT

- Qualify actions by the triggering condition

- ON DELETE, ON UPDATE

Semantic integrity constraints

- Constraint is expressed as a predicate, a condition similar to the one at the WHERE clause of a query

- Three DDL constructs: checks, assertions, triggers

Check constraints

- CHECK prohibits an operation on a table that would violate the constraint. It is a local constraint

- Ex: CONSTRAINT section\_budget\_ic1 CHECK((Budget >= 0) AND (BUDGET IS NOT NULL))

Assertions

- Similar to CHECK but they are global constraints

- CREATE OR REPLACE ASSERTION <assertion\_name> CHECK <predicate> [Mode of evaluation]

- Predicate usually involves EXISTS and NOT EXISTS

- Drop with DROP ASSERTION <name>

Triggers

- A trigger consists of 3 parts: event(s), condition, action

- Ex: notify the dean whenever the number of students in any major exceeds 1800

- AFTER INSERT OR UPDATE OF Major\_Code ON Student WHEN(EXISTS(… )) CALL email\_dean()

When triggers can fire in Postgres

- Time: BEFORE, AFTER, INSTEAD OF

- Event: INSERT, DELETE, UPDATE, TRUNCATE

- NEW and OLD are valid references in the trigger function

Row-level triggers

- FOR EACH ROW EXECUTE PROCEDURE

- OLD TABLE only for UPDATE or DELETE

- NEW TABLE only for UPDATE or INSERT

- No REFERENCING and no UPDATE of specific columns

Statement level triggers

- Referencing is valid only for statement-level triggers

- Only for one event and only for after event time

Enable/disable event triggers

- ALTER TABLE <name> ENABLE TRIGGER ALL;

- ALTER TABLE <name> ENABLE/DISABLE TRIGGER <name>

Drop triggers

- DROP TRIGGER statement is incompatible with SQL standard 🡪 trigger names are not local to tables

Mutating trigger

- Recursive call of triggers is not permitted