# SQL: Constraints, Triggers

CS348 Spring 2024

Instructor: Sujaya Maiyya

Sections: 002 and 003 only

# SQL features covered so far

- Query
  - SELECT-FROM-WHERE statements
  - Set and bag operations
  - Table expressions, subqueries
  - Aggregation and grouping
  - Ordering
  - Outerjoins (and NULL)
- Modification
  - INSERT/DELETE/UPDATE

Today: Constraints, schema changes, triggers

#### Constraints

- Restricts what data is allowed in a database
  - In addition to the simple structure and type restrictions imposed by the table definitions
- Why use constraints?
  - Protect data integrity (catch errors)
  - Tell the DBMS about the data (so it can optimize better)
- Declared as part of the schema and enforced by the DBMS

# Types of SQL constraints

- NOT NULL
- Key
- Referential integrity (foreign key)
- General assertion
- Tuple- and attribute-based CHECK's

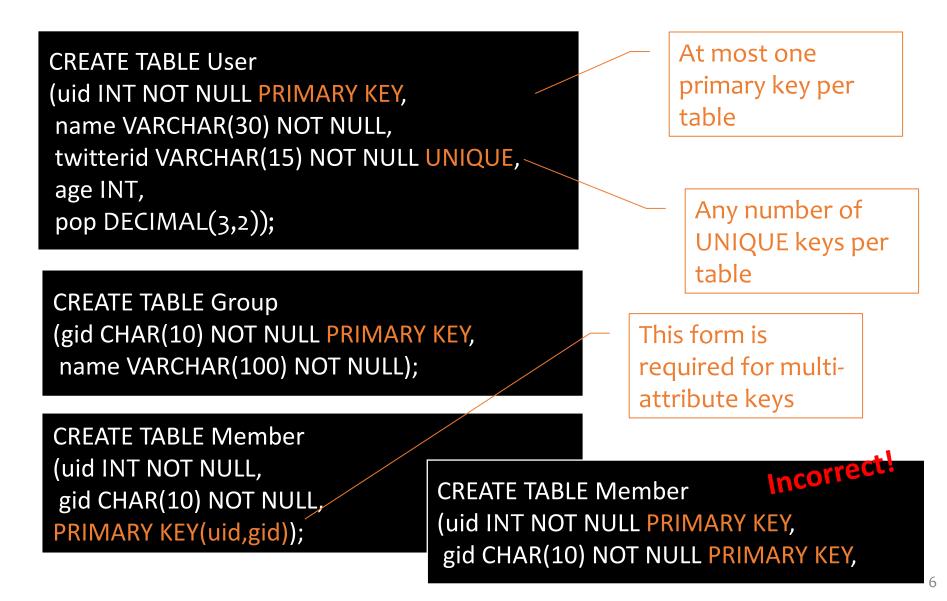
#### NOT NULL constraint examples

CREATE TABLE User (uid INT NOT NULL, name VARCHAR(30) NOT NULL, twitterid VARCHAR(15) NOT NULL, age INT, pop DECIMAL(3,2));

CREATE TABLE Group (gid CHAR(10) NOT NULL, name VARCHAR(100) NOT NULL);

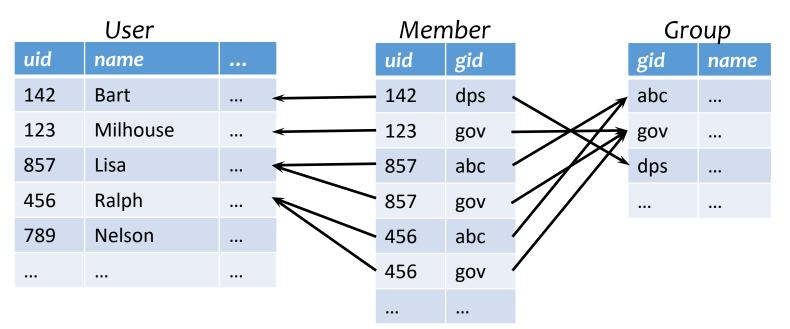
CREATE TABLE Member (uid INT NOT NULL, gid CHAR(10) NOT NULL);

#### Key declaration examples



# Referential integrity example

- If a uid appears in Member, it must appear in User
  - Member.uid references User.uid
- If a gid appears in Member, it must appear in Group
  - Member.gid references Group.gid
- That is, no "dangling pointers"



# Referential integrity in SQL

- Referenced column(s) must be PRIMARY KEY
- Referencing column(s) form a FOREIGN KEY
- Example

Some system allow them to be non-PK but must be UNIQUE

CREATE TABLE Member (uid INT NOT NULL REFERENCES User(uid), gid CHAR(10) NOT NULL, PRIMARY KEY(uid,gid), FOREIGN KEY (gid) REFERENCES Group(gid));

> This form is required for multiattribute foreign keys

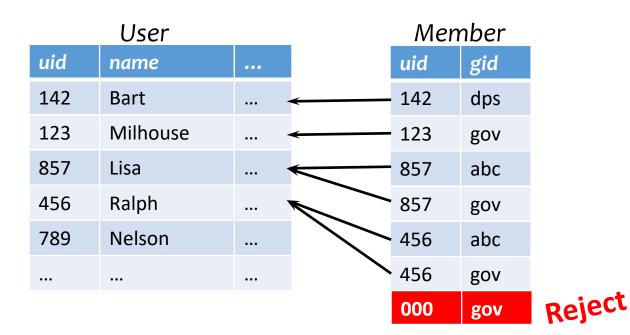
CREATE TABLE MemberBenefits (..... FOREIGN KEY (uid,gid) REFERENCES Member(uid,gid));

# Enforcing referential integrity

Example: Member.uid references User.uid

 Insert or update a Member row so it refers to a nonexistent uid

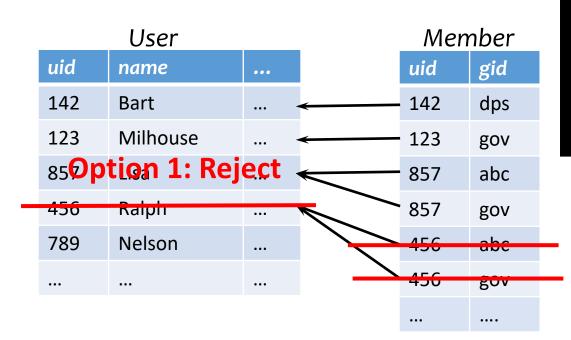
• Reject



# Enforcing referential integrity

Example: Member.uid references User.uid

- Delete or update a User row whose uid is referenced by some Member row
  - Multiple Options (in SQL)



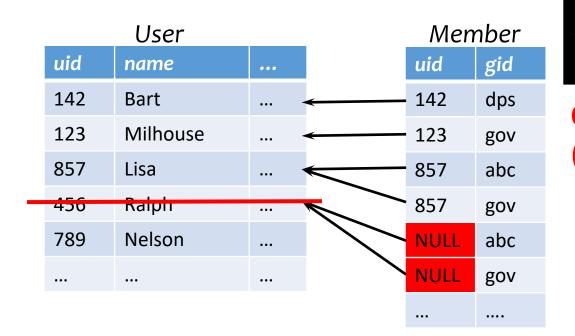
CREATE TABLE Member (uid INT NOT NULL REFERENCES User(uid) ON DELETE CASCADE, .....);

**Option 2: Cascade** (ripple changes to all referring rows)

# Enforcing referential integrity

Example: Member.uid references User.uid

- Delete or update a User row whose uid is referenced by some Member row
  - Multiple Options (in SQL)



CREATE TABLE Member (uid INT NOT NULL REFERENCES User(uid) ON DELETE SET NULL, .....);

#### **Option 3: Set NULL** (set all references to NULL)

#### General assertion

- CREATE ASSERTION assertion\_name CHECK assertion\_condition;
- assertion\_condition is checked for each modification that could potentially violate it
- Example: Member.uid references User.uid

CREATE ASSERTION MemberUserRefIntegrity CHECK (NOT EXISTS (SELECT \* FROM Member WHERE uid NOT IN (SELECT uid FROM User)));

Assertions are statements that must always be true

Can include

multiple

tables

# Tuple- and attribute-based CHECK's

- Associated with a single table
- Only checked when a tuple/attribute is inserted/updated
  - Reject if condition evaluates to FALSE
  - TRUE and UNKNOWN are fine
- Examples:

```
CREATE TABLE User(...
age INTEGER CHECK(age IS NULL OR age > 0),
...);
```

CREATE TABLE Member (uid INTEGER NOT NULL, CHECK(uid IN (SELECT uid FROM User)), Checked when new tuples are added to Member but not when User is modified

# Naming constraints

• It is possible to name constraints (similar to assertions)

```
CREATE TABLE User(...
age INT, constraint minAge check(age IS NULL OR age > 0),
...);
```

# In class exercises

Consider this db instance:

Member		Manala and an afita				
	uid	gid	MemberBenefits		nefits	
	857	dps		uid	gid	discou
	123	gov		857	dps	10
	857	abc		123	gov	25
	857	gov		857	abc	5
	456	abc				
	456	gov				

- MemberBenefits table references the Member table
- (uid, gid) forms the primary key of MemberBenefits table
- Assume discount is of type INT (and uid is INT and gid is string with a max of 30 characters)
- Write a DDL to create the MemberBenefits table

discount

### In class exercises

Consider this db instance:

Member		Manahar Dan afita			
uid	gid	MemberBenefits		nefits	
857	dps		uid	gid	discou
123	gov		857	dps	10
857	abc		123	gov	25
857	gov		857	abc	5
456	abc				
456	gov				

- MemberBenefits table references the Member table
- (uid,gid) forms the primary key of MemberBenefits table
- Assume discount is of type INT (and uid is INT and gid is string with a max of 30 characters)

**CREATE TABLE MemberBenefits** (uid INT, gid VARCHAR(30), discount INT, PRIMARY KEY (uid,gid), FOREIGN KEY (uid,gid) REFERENCES Member(uid,gid)); discount

# In class exercises

Consider this db instance:

Member		
uid	gid	
857	dps	
123	gov	
857	abc	
857	gov	
456	abc	
456	gov	

- Assume all foreign key references are set to ON DELETE SET NULL
- (Assume the db allows this, just for this exercise)
- What happens when user 857 is deleted from the User table? (Recall Member table references uid of User table)

#### In class ex.

User (<u>uid</u> int, name string, age int, pop float) Group (<u>gid</u> string, name string) Member (<u>uid</u> int, <u>gid</u> string)

• Assume the User table requires *pop* column values to be between 0 and 1. Complete the following DDL statement.

CREATE TABLE User (uid INT PRIMARY KEY, name VARCHAR(30), age INT, pop DECIMAL(3,2) ???);

#### In class ex.

User (<u>uid</u> int, name string, age int, pop float) Group (<u>gid</u> string, name string) Member (<u>uid</u> int, <u>gid</u> string)

• Assume the User table requires *pop* column values to be between 0 and 1. Complete the following DDL statement.

CREATE TABLE User (uid INT PRIMARY KEY, name VARCHAR(30), age INT, pop DECIMAL(3,2) CHECK(pop IS NULL OR (pop >= 0 AND pop < 1));

#### Take home ex.

User (<u>uid</u> int, name string, age int, pop float) Group (<u>gid</u> string, name string) Member (<u>uid</u> int, <u>gid</u> string)

 Say every user with pop >=0.9 must belong to the Book Club (gid='abc'). Create as assertion to check this constraint.

### Schema modification

- How to add constraints once the schema is defined??
- Add or Modify attributes/domains
- Add or Remove constraints

# Add or Modify attributes/domains

- Alter table table\_name Add column column\_name
- Alter table table\_name Rename column old\_name to new\_name
- Alter table table\_name Drop column column\_name

#### Domain change:

 Alter table table\_name Alter column column\_name datatype

> Error if column already has conflicting data!

#### Add or Remove constraints

• Alter table table\_name Add constraint constraint\_name constraint\_condition

ALTER TABLE Member ADD CONSTRAINT fk\_user FOREIGN KEY(uid) REFERENCES User(uid)

• Alter table table\_name Drop constraint constraint\_name

ALTER TABLE Member DROP CONSTRAINT fk\_user

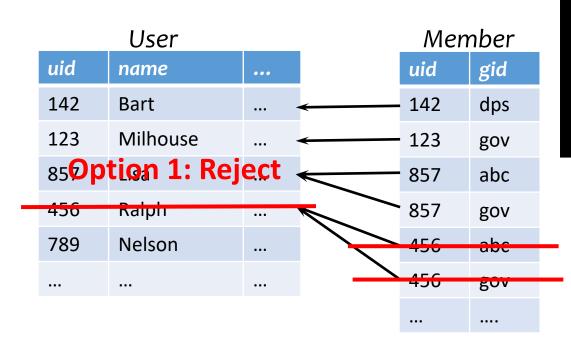
# SQL

- Constraints
- Schema changes
- Triggers

# Recall "referential integrity"

Example: Member.uid references User.uid

- Delete or update a User row whose uid is referenced by some Member row
  - Multiple Options (in SQL)



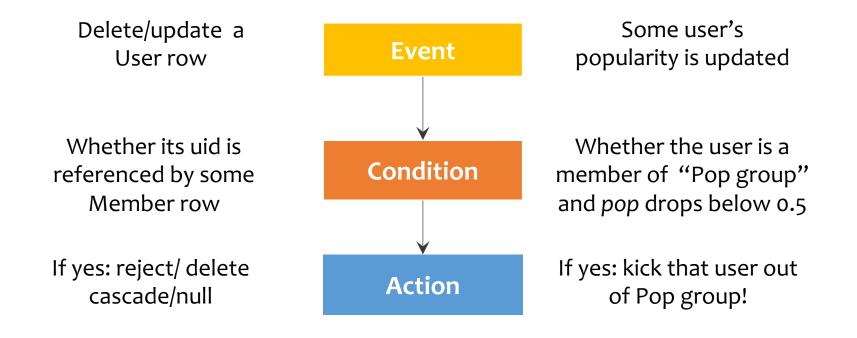
CREATE TABLE Member (uid INT NOT NULL REFERENCES User(uid) ON DELETE CASCADE, .....);

**Option 2: Cascade** (ripple changes to all referring rows)

## Can we generalize it?

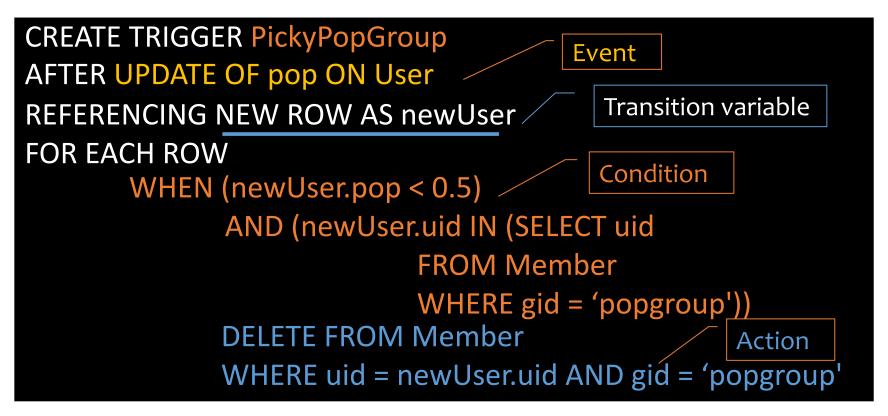
#### Referential constraints

#### Data Monitoring



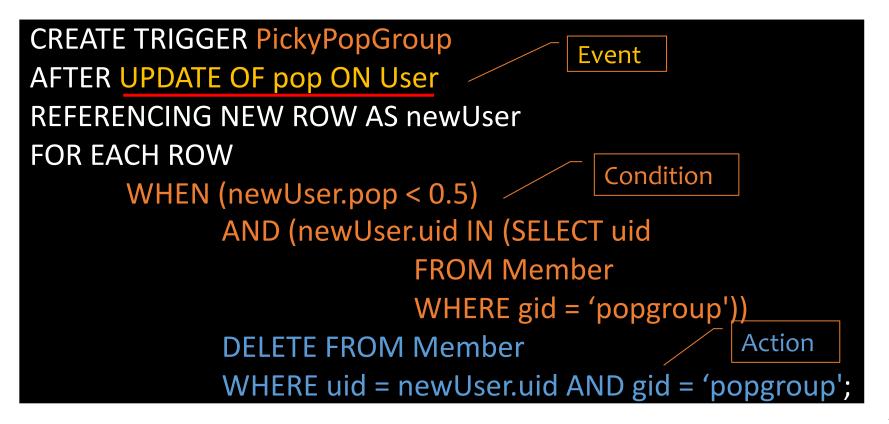
# Triggers

- A trigger is an event-condition-action (ECA) rule
  - When event occurs, test condition; if condition is satisfied, execute action



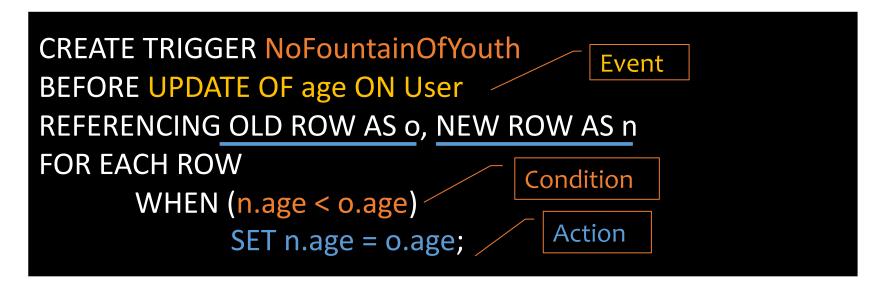
# Trigger option 1 – possible events

- Possible events include:
  - INSERT ON table; DELETE ON table; UPDATE [OF column] ON table



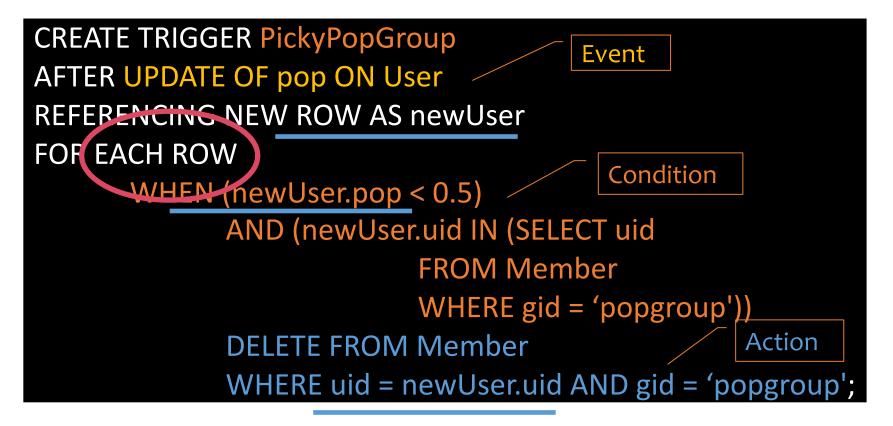
# Trigger option 2 – timing

- Timing—action can be executed:
  - AFTER or **BEFORE** the triggering event
  - INSTEAD OF the triggering event on views (more later)



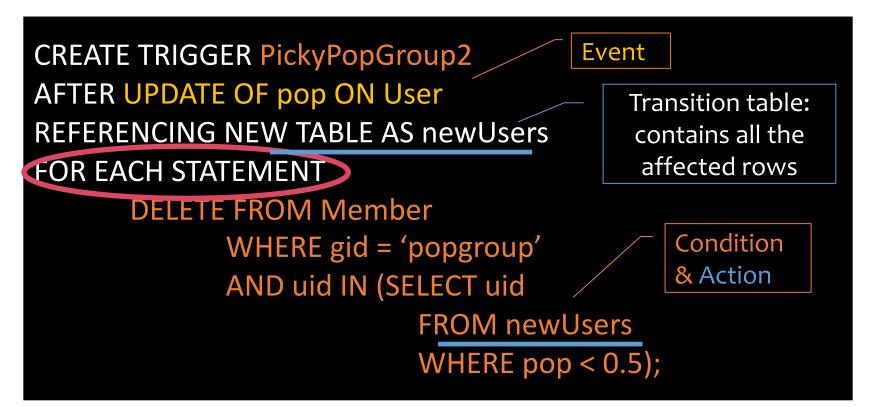
# Trigger option 3 – granularity

- Granularity—trigger can be activated:
  - FOR EACH ROW modified



# Trigger option 3 – granularity

- Granularity—trigger can be activated:
  - FOR EACH ROW modified
  - FOR EACH STATEMENT that performs modification



# Trigger option 3 – granularity

- Granularity—trigger can be activated:
  - FOR EACH ROW modified
  - FOR EACH STATEMENT that performs modification

CREATE TRIGGER PickyPopGroup2 **AFTER UPDATE OF pop ON User REFERENCING NEW TABLE AS newUsers** FOR EACH STATEMENT **DELETE FROM Member** WHERE gid = 'popgroup' AND uid IN (SELECT uid **FROM newUsers** 

Transition table: contains all the affected rows

Can only be used with **AFTER** triggers

WHERE pop < 0.5);

#### Transition variables/tables

- OLD ROW: the modified row before the triggering event
- NEW ROW: the modified row after the triggering event
- OLD TABLE: a read-only table containing all old rows modified by the triggering event
- NEW TABLE: a table containing all modified rows after the triggering event

Event	Row	Statement		
Delete	old r; old t	old t		
Insert	new r; new t	new t		
Update	old/new r; old/new t	old/new t		
AFTER Trigger				

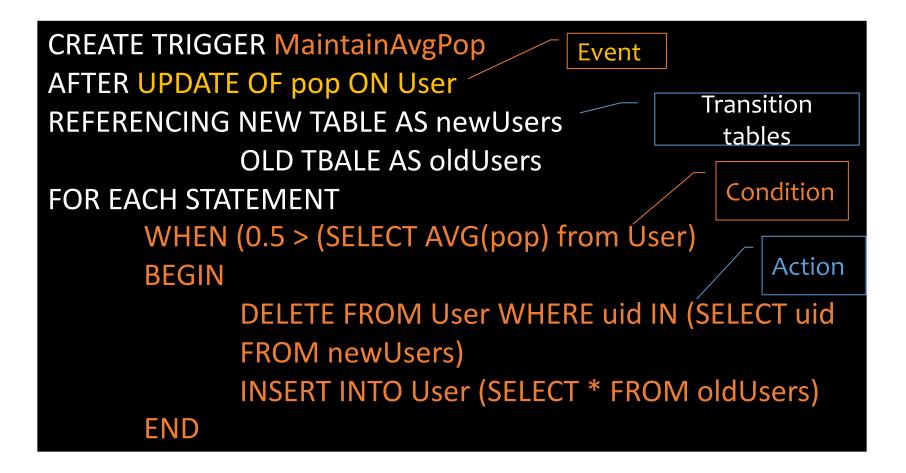
Event	Row	Statement
Update	old/new r	-
Insert	new r	-
Delete	old r	-

BEFORE Trigger

# Statement- vs. row-level triggers

- Simple row-level triggers are easier to implement
  - Statement-level triggers: require significant amount of state to be maintained in OLD TABLE and NEW TABLE
- However, in some cases a row-level trigger may be less efficient
  - E.g., 4B rows and a trigger may affect 10% of the rows. Recording an action for 4 Million rows, one at a time, is not feasible due to resource constraints.
- Certain triggers are only possible at statement level
  - E.g., ??

# Certain triggers are only possible at statement level



### System issues

- Recursive firing of triggers
  - Action of one trigger causes another trigger to fire
  - Can get into an infinite loop
- Interaction with constraints (tricky to get right!)
  - When to check if a triggering event violates constraints?
    - After a BEFORE trigger
    - Before an AFTER trigger
    - (based on db2, other DBMS may differ)
- Best to avoid when alternatives exist

# SQL features covered so far

Basic & Intermediate SQL

- Query
- Modification
- Constraints
- Triggers

Sext: Views, Indexes, Programming & recursion