Practice and Applications of Data Management

CMPSCI 345

Lecture 11: Constraints and Triggers
Midterm

- Wednesday, Oct 21, in class.

- Review:
  - Practice quizzes. Token: A36AB9A3
  - Lectures.
  - In-class practice and handouts
  - Review session on Monday, Oct 19.

- Conditions:
  - Closed book
  - Allowed: single-page letter-size cheat sheet (one-sided)
Updates on views

```
practice=# CREATE VIEW productPrice
practice-# AS SELECT pname, price
practice-# FROM product;
CREATE VIEW

practice=# INSERT INTO productPrice
practice-# VALUES ('iPad', 299.99);
```
Triggers!

- Stored code that is executed on particular events (e.g., INSERT, UPDATE, DELETE)

```
CREATE [ CONSTRAINT ] TRIGGER name
{ BEFORE | AFTER | INSTEAD OF } { event [ OR ... ] } ON table
[ FOR [ EACH ] { ROW | STATEMENT } ]
EXECUTE PROCEDURE function_name ( arguments )
```
```
practice=# CREATE FUNCTION upView()
practice-# RETURNS TRIGGER
practice-# LANGUAGE plpgsql
practice-# AS $function$
practice$# BEGIN
practice$# INSERT INTO Product
practice$# VALUES(NEW.pname, NEW.price,
practice$# 'Gadgets', NULL);
practice$# RETURN NEW;
practice$# END
practice$# $function$;
create function
```
practice=# CREATE TRIGGER triggerUpdate
practice-# INSTEAD OF INSERT
practice-# ON productPrice
practice-# FOR EACH ROW
practice-# EXECUTE PROCEDURE upView();
CREATE TRIGGER
Practice

```
practice=# INSERT INTO productPrice
practice-# VALUES ('iPad', 299.99);
```
## Constraints in SQL

- Keys, foreign keys
- Attribute-level constraints
- Tuple-level constraints
- Global constraints: assertions

The more complex the constraint, the harder it is to check and enforce.
Keys

Product(name, category)

CREATE TABLE Product (
    name CHAR(30) PRIMARY KEY,
    category VARCHAR(20))

OR:

CREATE TABLE Product (
    name CHAR(30),
    category VARCHAR(20)
    PRIMARY KEY (name))
Other Keys

CREATE TABLE Product (  
productID CHAR(10),  
name CHAR(30),  
category VARCHAR(20),  
price INT,  
PRIMARY KEY (productID),  
UNIQUE (name, category))

There is at most one PRIMARY KEY;  
there can be many UNIQUE
CREATE TABLE Purchase (  
  prodName CHAR(30)  
  REFERENCES Product(name),  
  date DATETIME)  

prodName is a foreign key to Product(name)  
name must be a key in Product
Practice

```
practice=# DELETE FROM Company
practice-# WHERE cname = 'Hitachi';

practice=# UPDATE Company
practice-# SET cname = 'Sony'
practice-# WHERE cname = 'Canon';
```
create table Product
    (pname varchar(30) primary key,
     price float,
     category varchar(30),
     manufacturer varchar(30)
    references Company
    ON DELETE SET NULL
    ON UPDATE CASCADE);
Copy the new create table command and then the insert statements from practiceData.sql
Practice

```
practice=# DELETE FROM Company
practice-# WHERE cname = 'Hitachi';

practice=# UPDATE Company
practice-# SET cname = 'Sony'
practice-# WHERE cname = 'Canon';
```
Tuple checks

```
practice=# CREATE TABLE practice-# A (x int CHECK (x > 10));

practice=# INSERT INTO A practice-# VALUES (5);
```
Tuple checks

```
practice=# CREATE TABLE B (x int , y int
practice-# CHECK (x + y > 10));

practice=# INSERT INTO B
practice-# VALUES (5,2);
```
Constraint Examples

CREATE TABLE Purchase (  
    prodName CHAR(30)  
    CHECK (prodName IN  
    SELECT Product.name  
    FROM Product),  
    date DATETIME NOT NULL)

How is this different from a foreign key constraint?

(in the SQL standard, but not always implemented)
CREATE ASSERTION myAssert CHECK

NOT EXISTS(
    SELECT Product.name
    FROM Product, Purchase
    WHERE Product.name = Purchase.prodName
    GROUP BY Product.name
    HAVING count(*) > 200)

(in the SQL standard, but not always implemented)
Semantic optimization with constraints

Product(name, price)
Purchase(buyer, seller, prodName, store)

SELECT Purchase.store
FROM Product, Purchase
WHERE Product.name = Purchase.prodName

SELECT Purchase.store
FROM Purchase
Semantic optimization with constraints

\[
\text{Product}(\text{name}, \text{price})
\]
\[
\text{Purchase}(\text{buyer, seller, prodName, store})
\]

\[
\text{SELECT} \quad \text{Purchase}.\text{store}
\]
\[
\text{FROM} \quad \text{Product, Purchase}
\]
\[
\text{WHERE} \quad \text{Product}.\text{name} = \text{Purchase}.\text{prodName}
\]

Yes, if \text{Purchase}.\text{prodName} is a foreign key, and not null