Lecture 14: SQL

Wednesday, October 25, 2000

Homework Update

```
SELECT count(distinct age, city)
FROM   Person
```

- Doesn't work in SQL-Server ☹
- Try *nested queries* (or *subqueries*)

```
SELECT count(*)
FROM (SELECT DISTINCT age, city
      FROM Person) AS X
```

Outline

- Null Values and Outerjoins 5.9
- Keys and Foreign Keys 6.1, 6.2

Null Values and Outerjoins

Explicit joins in SQL:

```
Product(name, category)
Purchase(prodName, store)

SELECT Product.name, Purchase.store
FROM   Product JOIN Purchase ON
           Product.name = Purchase.prodName
```

Same as:

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

But Products that never sold will be lost !

Null Values and Outerjoins

Left outer joins in SQL:

```
Product(name, category)
Purchase(prodName, store)

SELECT Product.name, Purchase.store
FROM   Product LEFT OUTER JOIN Purchase ON
           Product.name = Purchase.prodName
```

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	-

Left Outer Joins and XML

- Export the database in XML, grouped by products:

```
<Catalog>
    <product> <name> Gizmo </name>
              <store>  Wiz </store>
    </product>
    <product> <name> Camera </name>
              <store>  Ritz </store>
              <store>  Wiz </store>
    </product>
    <product>  <name> OneClick </name>
    </product>
</Catalog>
```

Don't want to lose OneClick just because it never sold !

Left Outer Joins and XML

- To generate the XML data first construct a table with outer joins:

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	-

- Sort it by Name
- Then do the "tagging"
- Note: in your project you may try something simpler…

Other Outer Joins

- Left outer join:
  - Incluce the left tuple even if there's no match
- Right outer join:
  - Incluce the right tuple even if there's no match
- Full outer join:
  - Include the both left and right tuples even if there's no match

SQL: Constraints and Triggers

- Chapter 6 Ullman and Widom
- Certain properties on the database that must hold
- Modification of the database may break them
- Build handlers into the database definition
- Kinds of constraints:
  - Key constraints
  - Referential integrity constraints
  - Others in Chapter 6 (reading assignment)

Declaring a Primary Keys in SQL

```
CREATE TABLE MovieStar (
    name CHAR(30) PRIMARY KEY,
    address VARCHAR(255),
    gender CHAR(1));
```

OR:

```
CREATE TABLE MovieStar (
    name CHAR(30),
    address VARCHAR(255),
    gender CHAR(1)
    PRIMARY KEY (name));
```

Primary Keys with Multiple Attributes

```
CREATE TABLE MovieStar (
    name CHAR(30),
    address VARCHAR(255),
    gender CHAR(1),
    PRIMARY KEY (name, address));
```

Other Keys

```
CREATE TABLE MovieStar (
   name CHAR(30),
   address VARCHAR(255),
   phone  CHAR(10)  UNIQUE,
   gender CHAR(1),
   petName  CHAR(50),
   PRIMARY KEY (name),
   UNIQUE (gender, petName));
```

- Only one primary key, arbitrary many UNIQUE's

Enforcing Key Constraints

- On insertion
  - Check if another one is there; reject if yes
- On updates
  - same
- On deletion
  - Nothing to check

Foreign Key Constraints

```
CREATE TABLE ActedIn (
   Name CHAR(30) PRIMARY KEY,
   MovieName CHAR(30)
        REFERENCES Movies(MovieName),
   Year INT);
```

Foreign Key Constraints

- OR

```
CREATE TABLE ActedIn (
   Name CHAR(30) PRIMARY KEY,
   MovieName CHAR(30),
   Year INT,
   FOREIGN KEY MovieName
        REFERENCES Movies(MovieName)
```

- MovieName must be a PRIMARY KEY

Maintaining Foreign Key Constraints

- Insert new tuple with bogus foreign key value
- Update a tuple to a bogus foreign key value
- Delete a tuple in the referenced table with the referenced foreign key value
- Update a tuple in the referenced table that changes the referenced foreign key value

Policies

- Policies for handling:
  - Reject the update (default)
  - Cascade (example: cascading deletes)
  - Set NULL
- Can set update and delete actions independently in CREATE TABLE

```
MovieName CHAR(30)
        REFERENCES Movies(MovieName)
                ON DELETE SET NULL
                ON UPDATE CASCADE
```