Lecture 15

Friday, November 2, 2001

Outline

- More aggregates
- Null values (5.9)
- Constraints (6)

Aggregation

Author(<u>login</u>,name)
Document(<u>url</u>, title)
Wrote(login,url)
Mentions(url,word)

- Find all authors who wrote at least 10 documents:

```
Select author.name
From author, wrote
Where author.login=wrote.login
Groupby author.name
Having count(wrote.url) > 10
```


- Find all authors who have a vocabulary over 10000:

```
Select author.name
From author, wrote, mentions
Where author.login=wrote.login and wrote.url=mentions.url
Groupby author.name
Having count(distinct mentions.word) > 10000
```


Null Values and Outerjoins

- If x=Null then 4*(3-x)/7 is still NULL
- If x=Null then x="Joe" is UNKNOWN
- Three boolean values:
  - FALSE = 0
  - UNKNOWN = 0.5
  - TRUE = 1

Null Values and Outerjoins

- C1 AND C2 = min(C1, C2)
- C1 OR C2 = max(C1, C2)
- NOT C1 = 1 – C1

```
SELECT *
FROM Person
WHERE (age < 25) AND
      (height > 6 OR weight > 190)
```

Rule in SQL: include only tuples that yield TRUE

7

Null Values and Outerjoins

Unexpected behavior:

```
SELECT *
FROM   Person
WHERE  age < 25  OR  age >= 25
```

Some Persons are not included !

8

Null Values and Outerjoins

Can test for NULL explicitly:

- x IS NULL
- x IS NOT NULL

```
SELECT *
FROM   Person
WHERE  age < 25  OR  age >= 25 OR age IS NULL
```

Now it includes all Persons

9

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 !

10

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
```

11

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	-

12

Left Outer Joins and XML

```
<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 !

13

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

14

SQL: Constraints and Triggers

- Chapter 6 Ullman and Widom
- Certain properties we'd like our database to hold
- Modification of the database may break these properties
- Build handlers into the database definition
- Key constraints
- Referential integrity constraints.

15

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));
```

16

Primary Keys with Multiple Attributes

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

17

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));
```

18

Foreign Key Constraints

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

19

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

20

How do we Maintain them?

- Given a change to DB, there are several possible violations:
  - 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

21

How to Maintain?

- Recall, ActedIn has FK MovieName...

Movies(MovieName, year)
(Fatal Attraction, 1987)

ActedIn(ActorName, MovieName)
(Michael Douglas, Fatal Attraction)
insert: (Rick Moranis, Strange Brew)

22

How to Maintain?

- Policies for handling the change…
  - 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
```

23