CS 235: Introduction to Databases

Svetlozar Nestorov

Lecture Notes #11

Outline

- More aggregation queries
  - Grouping.
  - Having clause
- Database modifications
  - Insertion
  - Deletion
  - Updates

Grouping

- Follow select-from-where by GROUP BY and a list of attributes.
- The relation that is the result of the FROM and WHERE clauses is grouped according to the values of these attributes, and aggregations take place only within a group.
- Find the average price for each beer.

```
SELECT beer, AVG(price)
FROM Sells
GROUP BY beer;
```

Example

- Find, for each drinker, the average price of Bud at the bars they frequent.

```
SELECT drinker, AVG(price)
FROM Frequents, Sells
WHERE beer = 'Bud' AND
    Frequents.bar = Sells.bar
GROUP BY drinker;
```

- Note: grouping occurs after the $\times$ and σ operations.

Restriction on SELECT Lists With Aggregation

- If any aggregation is used, then *each* element of a SELECT clause must either be aggregated or appear in a group-by clause.
- The following might seem a tempting way to find the bar that sells Bud the cheapest:

```
SELECT bar, MIN(price)
FROM Sells
WHERE beer = 'Bud';
```

- But it is illegal in SQL.
- How would we find that bar?

HAVING Clauses

- HAVING clauses are selections on groups, just as WHERE clauses are selections on tuples.
- Condition can use the tuple variables or relations in the FROM and their attributes, just like the WHERE can.
  - But the t.v.'s range only over the group.
  - And the attribute better make sense within a group; *i.e.*, be one of the grouping attributes.

Example

- Find the average price of those beers that are either served in at least 3 bars or manufactured by Anheuser-Busch.

```
SELECT beer, AVG(price)
FROM Sells
GROUP BY beer
HAVING COUNT(*) >= 3 OR
     beer IN (
               SELECT name
               FROM Beers
               WHERE manf = 'Anheuser-Busch'
               );
```

Another Example

- Find, for each manufacturer, the beer with highest average price.

DB Modifications

- Results of modifications last beyond your session!
- Three types of modifications:
  - Insert new tuple.
  - Delete current tuple.
  - Update current tuple.
    - Update is not strictly necessary since it can be substituted by a delete and an insert.

Insertion

- INSERT INTO relation VALUES (list of values).
- Inserts the tuple = list of values, associating values with attributes in the order the attributes were declared.
  - You can also list the attributes as arguments of the relation.
- Insert the fact that Sally likes Bud in Likes(drinker, beer)

```
INSERT INTO Likes(drinker, beer) VALUES('Sally', 'Bud');
```

Insertion of the Result of a Query

- INSERT INTO relation (subquery).
- Create a (unary) table of all Sally's potential buddies, i.e., the people who frequent bars that Sally also frequents.
- Frequents(drinker, bar)

```
CREATE TABLE PotBuddies(
    name char(30)
);
```

Example

```
INSERT INTO PotBuddies
(SELECT DISTINCT d2.drinker
 FROM Frequents d1, Frequents d2
 WHERE d1.drinker = 'Sally' AND
        d2.drinker <> 'Sally' AND
        d1.bar = d2.bar
);
```

Bulk Loading

- Insert many tuples from a data file with a single command.

```
LOAD DATA
LOCAL INFILE “likes.dat”
INTO TABLE Likes;
```

- The keyword LOCAL means that the data file is on the client machine.

Deletion

DELETE FROM relation WHERE condition.

- Deletes all tuples satisfying the condition from the named relation.
- Sally no longer likes Bud.

```
DELETE FROM Likes
WHERE drinker = 'Sally' AND beer = 'Bud';
```

- Make the Likes relation empty.

```
DELETE FROM Likes;
```

  - In practice, it’s more efficient to drop and create the table.

Example

- Delete all beers for which there is another beer by the same manufacturer.

```
DELETE FROM Beers b
WHERE EXISTS
    (SELECT name
     FROM Beers
     WHERE manf = b.manf AND
           name <> b.name);
```

- Note alias for relation from which deletion occurs.
- Not (yet) allowed in MySQL.

Semantics

- Semantics is tricky. If A.B. makes Bud and BudLite (only), does deletion of Bud make BudLite not satisfy the condition?
- SQL semantics: all conditions in modifications must be evaluated by the system before any modifications due to that modification command occur.
  - In Bud/Budlite example, we would first identify both beers as targets, and then delete both.

Updates

- UPDATE relation SET list of assignments WHERE condition.
- Drinker Leo's phone number is 555-1212.

```
UPDATE Drinkers
SET phone = '555-1212'
WHERE name = ‘Leo;
```

- Make $4 the maximum price for beer.

```
UPDATE Sells
SET price = 4.00
WHERE price > 4.00;
```