

CS411

Database Systems

06c: SQL-3

DB Modification, Definition, Views

Database Modification

Database Modifications

- A modification command does not return a result as a query does, but it changes the database in some way.
- There are three kinds of modifications:
 1. *Insert* a tuple or tuples.
 2. *Delete* a tuple or tuples.
 3. *Update* the value(s) of an existing tuple or tuples.

Insertion

- To insert a single tuple:

```
INSERT INTO <relation>  
VALUES ( <list of values> );
```

- Example: add to Likes (person, beer) the fact that Sally likes Bud.

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

Specifying Attributes in Insert

- Another way to add the fact that Sally likes Bud to Likes (person, beer):

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

Specifying Attributes in INSERT

- We may add to the relation name a list of attributes.
- There are two possible reasons to do so:
 1. We may have forgotten the standard order of attributes for the relation.
 2. We don't have values for all attributes, and we want the system to fill in missing components with NULL or a default value.

Inserting Many Tuples

- We may insert the entire result of a query into a relation, using the form:

```
INSERT INTO <relation>  
( <subquery> );
```

E.g., INSERT INTO Beers(name)
SELECT beer from Sells;

Example: Insert a Subquery

- Using `Frequents(person, bar)`, enter into the new relation `PotBuddies(name)` all of Sally's "potential buddies," i.e., those persons who frequent at least one bar that Sally also frequents.

Solution

The other
person

INSERT INTO PotBuddies

(SELECT d2.person

FROM Frequents d1, Frequents d2
WHERE d1.person = 'Sally' AND
d2.person <> 'Sally' AND
d1.bar = d2.bar

);

Pairs of person
tuples where the
first is for Sally,
the second is for
someone else,
and the bars are
the same.

Deletion

- To delete tuples satisfying a condition from some relation:

```
DELETE FROM <relation>  
WHERE <condition>;
```

Example: Deletion

- Delete from Likes (person, beer) the fact that Sally likes Bud:

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

Example: Delete all Tuples

- Make the relation Likes empty:

```
DELETE FROM Likes;
```

- Note no WHERE clause needed.

Example: Delete Many Tuples

- Delete from Beers(name, manf) all beers manufactured by Anheuser-Busch.

DELETE FROM Beers

WHERE name = 'Anheuser-Busch';

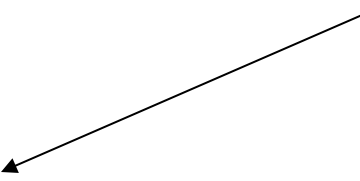
Another Example: Delete Many Tuples

- Delete from Beers (name, manf) all beers for which there is another beer by the same manufacturer.

```
DELETE FROM Beers b  
WHERE name IN (
```

```
SELECT b1.name  
FROM Beers b1, Beers b2  
WHERE b1.manf = b2.manf AND  
      b1.name <> b2.name);
```

Beers with the same manufacturer and a different name from the name of the beer represented by tuple b.



Note: Cannot do this in MySQL. Syntax error! But there is a way out. (Not important here.)

Semantics of Deletion -- 1

- Suppose Busch makes only Bud and Bud Lite. ... Suppose we come to the tuple b for Bud first. ... The subquery is nonempty, because of the Bud Lite tuple, so we delete Bud. ... Now, When b is the tuple for Bud Lite, do we delete that tuple too?

Semantics of Deletion -- 2

- The answer is that we *do* delete Bud Lite as well.
- The reason is that deletion proceeds in two stages:
 1. Mark all tuples for which the WHERE condition is satisfied in the original relation.
 2. Delete the marked tuples.

Updates

- To change certain attributes in certain tuples of a relation:

UPDATE <relation>

SET <list of attribute assignments>

WHERE <condition on tuples>;

Example: Update

- Change Fred's phone number to 555-1212:

```
UPDATE Person
```

```
SET phone = '555-1212'
```

```
WHERE name = 'Fred';
```

Example: Update Several Tuples

- Increase price that is cheap:

```
UPDATE Sells
```

```
SET price = price * 1.07
```

```
WHERE price < 3.0;
```

Defining a Database Schema

Views

Views

- A view is a “virtual table,” a relation that is defined in terms of the contents of other tables and views.
- Declare by:

```
CREATE VIEW <name> AS <query>;
```
- Views are not stored in the database, but can be queried as if they existed.
- In contrast, a relation whose value is really stored in the database is called a *base table*.

Example: View Definition

- CanDrink (person, beer) is a view “containing” the person-beer pairs such that the person frequents at least one bar that serves the beer:

```
CREATE VIEW CanDrink AS
  SELECT person, beer
  FROM Frequents, Sells
  WHERE Frequents.bar = Sells.bar;
```

Example: Accessing a View

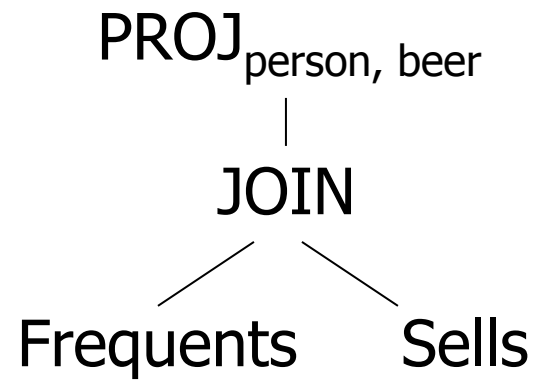
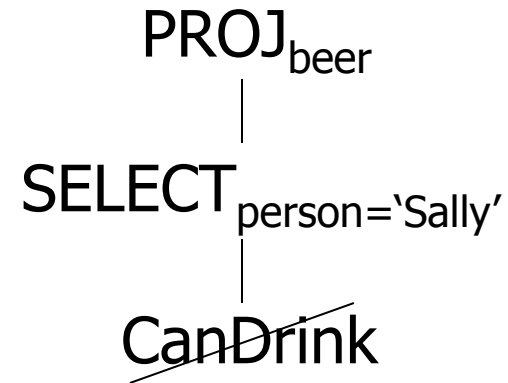
- You may query a view as if it were a base table.
- Example:

```
SELECT beer FROM CanDrink  
WHERE person = 'Sally';
```


What Happens When a View Is Used?

- The DBMS starts by interpreting the query as if the view were a base table.
 - Typical DBMS turns the query into something like relational algebra.
- The queries defining any views used by the query are also replaced by their algebraic equivalents, and “spliced into” the expression tree for the query.

Example: View Expansion



Updating Views

- You may modify a view also. But only if the modification “makes sense as a modification of the underlying base table”.

Updating Views

How can I insert a tuple into a table that doesn't exist?

```
CREATE VIEW JoeBarSells AS  
  SELECT beer, price  
  FROM Sells  
  WHERE bar = 'joe bar';
```

If we make the
following insertion:

```
INSERT INTO JoeBarSells  
VALUES("bud special", 3.5)
```

It becomes:

```
INSERT INTO Sells  
VALUES(NULL, 'bud special', 3.5)
```

Q: Is the new tuple in table Sells? In the view JoeBarSells?

Makes more sense to:

```
CREATE VIEW JoeBarSells AS  
  SELECT bar, beer, price  
  FROM Sells  
  WHERE bar = 'joe bar';
```

If we make the
following insertion:

```
INSERT INTO JoeBarSells  
VALUES('joe bar', 'bud special', 3.5)
```

It becomes:

```
INSERT INTO Sells  
VALUES('joe bar', 'bud special', 3.5)
```

Now a query on the view JoeBarSells will return 'bud special' also

View Update Can be Tricky!

- Check textbook for rules of “updatable” views.
- Loosely speaking the rules are:
 - The view must have been defined using SELECT on some attributes from *one* relation R
 - The WHERE clause must not involve R in a subquery
 - The SELECT list must include every attribute for which we cannot fill in NULL as default value
- Even when an update is allowed, it may not work intuitively as you would expect.