Outline

- So far, load data, then query.
  - What if we want to change something?
- DB Modifications.
  - Insertion.
  - Deletion.
  - Updates.
- Schema creation.
- Views.

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`  
`  );`
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;

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.

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;

Defining a Database Schema

- CREATE TABLE name (list of elements).
- Principal elements are attributes and their types,
  but key declarations and constraints also
  appear.
- Similar CREATE X commands for other schema
  elements X: views, indexes, assertions, triggers.
- DROP X name
  - deletes the created element of kind X with that
    name.

Example

- DROP TABLE Bars;
- CREATE TABLE Bars (  
  bar CHAR(20),  
  beer VARCHAR(20)  
  price REAL  
 );
Types

1. INT or INTEGER.
2. REAL or FLOAT.
3. CHAR(n) = fixed length character string, padded with "pad characters."
4. VARCHAR(n) = variable-length strings up to n characters.
   - Oracle uses VARCHAR2(n) as well. Difference: storage for VARCHAR2 is truly varying length; VARCHAR uses fixed array with end marker.
   - VARCHAR in Oracle is "deprecated" (they may discontinue it in the future), so they suggest you always use VARCHAR2.

More Types

- Dates. SQL form is DATE 'yyyy-mm-dd'
  - Oracle uses a different format – check out the online guide.
- Times. Form is TIME 'hh:mm:ss[.ss...]' in SQL but not in Oracle!
- In Oracle: NUMBER is either integer or floating point as appropriate.

Declaring Keys

- Use PRIMARY KEY or UNIQUE.
- Oracle treats these as synonyms.
- But only one primary key, many uniques allowed.
- SQL does not allow nulls in primary key, but allows them in unique columns (which may have two or more nulls, but not repeated non-null values).

Declaring Keys

Two places to declare:
1. After an attribute's type, if the attribute is a key by itself.
2. As a separate element.
   - Essential if key is >1 attribute.

Example

```sql
CREATE TABLE Sells (  
  bar CHAR(20),  
  beer VARCHAR(20),  
  price REAL,  
  PRIMARY KEY(bar,beer)  
);```

Example

```sql
CREATE TABLE Sells (  
  bar CHAR(20),  
  beer VARCHAR(20),  
  price REAL,  
  invoice NUMBER UNIQUE,  
  PRIMARY KEY(bar,beer)  
);```
Other Properties You Can Give to Attributes

- **NOT NULL =** every tuple must have a value for this attribute.
- **DEFAULT value =** a value to use whenever no other value of this attribute is known.

```
CREATE TABLE Drinkers (
    name CHAR(30) PRIMARY KEY,
    addr CHAR(50) DEFAULT '1100 E 58th str',
    phone CHAR(16)
);
```

Example

- **INSERT INTO Drinkers(name) VALUES(’Sally’)** results in the following tuple:
  (Sally, ’1100 E 58th str.’, NULL)
- Primary key is by default not NULL.
- This insert is legal.
  - OK to list a subset of the attributes and values for only this subset.
  - But if we had declared phone CHAR(16) NOT NULL then the insertion could not be made.

Changing Columns

- Add an attribute of relation \( R \) with
  ```
  ALTER TABLE \( R \) ADD <column declaration>;
  ```
- ```
  ALTER TABLE Bars ADD phone CHAR(16) DEFAULT ’unlisted’;
  ```
- Columns may also be dropped.
  ```
  ALTER TABLE Bars DROP license;
  ```

Views

- An expression that describes a table without creating it.
- View definition form is:
  ```
  CREATE VIEW <name> AS <query>;
  ```
- Create a view CanDrink for the set of drinker-beer pairs such that the drinker frequents at least one bar that serves the beer.

Querying Views

- Treat the view as if it were a materialized relation.
- What beers can Sally drink?
  ```
  SELECT beer
  FROM CanDrink
  WHERE drinker = ’Sally’;
  ```

Semantics of View Use

- Combine the view definition and the query and derive a new SQL expression.
  1. SQL query over view to relational expression
  2. SQL view def. to relational algebra
  3. Combine and optimize
  4. Translate back to SQL.
Example

Compose

Optimize Query
1. Push selections down tree.
2. Eliminate unnecessary projections.