# CIS4301 Notes:

Ryan Roden-Corrent

February 12, 2014

## 1 Database Modifications

### 1.1 Insert

Listing 1: multiple value insertion

```
INSERT INTO Likes
VALUES ('Sally', 'Bud'), ('Jim','Miller'); --comma separated tuples to enter
```

#### 1.1.1 Default Values

Listing 2: price defaults to 5 if not specified

```
...,
price Real DEFAULT 5, --make sure price is a reasonable, non-NULL value
...,
```

Listing 3: another default example

```
CREATE TABLE Drinkers (
  name CHAR(30) PRIMARY KEY,
  addr CHAR(50)
    DEFAULT '123 Sesame St.',
  phone CHAR(16)
);
```

#### 1.1.2 Subqueries in insertion

Listing 4: insertion via subquery

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

Find all the drinkers at the bars Sally frequents and insert them into PotBuddies (Potential Buddies, what did you think it stands for?)

| d1.bar | d2.bar |
|---|---|
| 'Sally' | NOT 'Sally' |
| 'Sally' | NOT 'Sally' |

## 1.2 Deletion

Listing 5: Sally no longer likes Bud

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

Delete all rows where the drinker is 'Sally' and the beer is 'Bud'

Listing 6: clear out entire table

```
DELETE FROM Likes; -- no WHERE clause needed
```

Listing 7: delete with subquery

```
DELETE FROM Beers b
WHERE EXISTS ( --check if another beer is made by the same manufacturer
  SELECT name FROM Beers --implicit join of Beers with itself
  WHERE manf = b.manf AND
    name <> b.name
);
```

Delete all beers where there is another beer by the same manufacturer.

| name | manf | |
|---|---|---|
| Bud | Budweiser | mark as dirty |
| BudLite | Budweiser | mark as dirty |

Delete is a **mark-and-sweep** process: first mark items for deletion, then delete all marked items. (If items were deleted immediately, it could disrupt the condition for deleting other items during the same deletion process).

## 1.3 Updates

Listing 8: UPDATE template

```
UPDATE <relation>
SET <list of attribute assignments>
WHERE <condition on tuples>;
```

Listing 9: Change Fred's Phone number

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

Listing 10: set maximum price on beers

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

Listing 11: add tax to price

```
UPDATE Sells
SET price = 1.05 * price --value can be result of a computation on attributes
WHERE price > 4.00;
```

## 2 Constraints

**constraint** relations enforced by DBMS

**trigger** only executed when a condition occurs

**Keys**

**Foreign-keys** referential integrity

**value-based** constrain value of attribute

**tuple-based** relationships between components

**assertions** boolean expression

### 2.1 Keys

#### 2.1.1 Single Attribute Keys

Listing 12: ensure names are unique

```
CREATE TABLE Beers (
  name CHAR(20) UNIQUE, --note: name can still be NULL!
  manf CHAR(20)
);
```

### 2.1.2 Multi Attribute Keys

Listing 13: tuple as a primary key

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

### 2.1.3 Foreign Keys

## 2.2 Foreign Keys

Indicate that a key REFERENCES another relation and is used as a key.
Referenced attributes must be declared PRIMARY KEY or UNIQUE.

Listing 14: tuple as a primary key

```
CREATE TABLE Sells (
bar CHAR(20),
beer VARCHAR(20),
price REAL,
FOREIGN KEY (beer) REFERENCES Beer);
```