3. Updates

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

```
update <relation>
set <list of attribute assignments>
where <condition on tuples>;
```

- Example: Change patron Nigel's phone number to 250-214-9999

```
update Patrons
set phone = '250-214-9999'
where name = 'Nigel';
```

Update several tuples

- Make $4 the maximum price for beer:

```
update sells
set price = 4.00
where price > 4.00;
```

Constraints & Triggers

- **Constraint**
  - A relationship among data elements the DBMS is required to enforce
  - Example: key constraints
- **Triggers**
  - A technique for implementing some types of constraints
  - Only executed when a certain condition or event occurs
  - Example: Trigger executed when a tuple is inserted into a table.
  - Rationale: Triggers are a way to implement seemingly complex constraints.
- We will look at each of these mechanisms in some detail

Constraints: different kinds

- **Keys**
- **Foreign keys**
  - Also referred to as referential integrity constraints
- **Value-based** constraints
  - Constrain values for a particular table attribute
- **Tuple-based** constraints
  - Apply to relationships amongst components in the database
- **Assertions**
  - Anything that can be denoted as an SQl boolean expression

Recall: single-attribute keys

- Used the keywords **primary key** or **unique**
- These are placed after the statement of type in a table declaration

```
create table Beers (
    name  char(20)  unique,
    manf  char(20)
);
```

Recall: multi-attribute keys

- The **bar** and **beer** attributes together are the key for the Sells relationship

```
create table Sells (
    pub     char(20),
    beer    varchar(20),
    price   real,
    primary key (pub, beer)
);
```

Foreign keys

- Idea:
  - Values appearing in attributes of one relation **must appear** together as specific in another relation.
- Example:
  - Consider **Sells(pub, beer, price)**
  - We might expect that a **beer** value also appears in **Beers** as **name**
  - (Note: Here we are ourselves asserting the degree to which our model must correspond to the real-world equivalent.)

Foreign keys: expressing

- Use the keyword **references**
  - After an attribute (for one-attribute keys), or
  - As an element of the schema.
- Attributes referenced in the other schema must themselves be declared primary key or unique

```
foreign key (<list of attributes>)
references <relation> (<attributes>)
```

Example: denoted with attribute

```
create table Beers (
    name   char(20)  primary key,
    manf   char(20)
);
```

```
create table Sells (
    pub    char(20),
    beer   char(20) references Beers(name),
    price  real
);
```

Example: denoted as schema element

```
create table Beers (
    name   char(20)  primary key,
    manf   char(20)
);
```

```
create table Sells (
    pub    char(20),
    beer   char(20),
    price  real,
    foreign key(beer) references Beers(name)
);
```

Foreign keys: enforcing the constraints

- If there is a foreign-key constraint from relation **R** to relation **S**...
- ... then **two violations are possible**.

1. An **insert** or **update** to **R** introduces values not found in S.
2. A **deletion** or **update** to **S** causes some tuples of R to "dangle"

Foreign keys: actions taken

- Example:
  - Suppose R corresponds to Sells ...
  - ... and S corresponds to Beers
- Possibility 1: An insert or update to Sells that introduces a non-existent beer must be rejected.
- Possibility 2: A deletion or update to Beers removing a beer value found in some tuples of Sells can be handled by:
  - **Rejecting** ...
  - **Cascading** ...
  - ... or **Nulling**.

Foreign keys: actions taken

- Default: **rejecting**
  - modification is not applied (i.e., it is rejected)
- **Cascading**: Same change in Beers is made in Sells
  - Updated beer: change all corresponding values in Sells
  - Deleted beer: delete all corresponding Sells tuples
- **Null**: Change beer in Sells tuples to the **null** value.

Cascade: examples

- Recall: R == Sells, S == Beers
- Delete from S
  - Delete the ('Blue', 'Labatt''s') tuple from Beers.
  - Result: DBMS delete all tuples from Sells where beer = 'Blue'
- Update to S
  - Change the ('Blue', 'Labatt''s') tuple in Beers to ('Blue (Pilsner)', 'Labatt''s).
  - DBMS changes all Sells tuples with beer = 'Blue' to 'Blue (Pilsner)'

Set null: examples

- Delete ('Blue', 'Labatt''s') tuple from Beers
  - DBMS changes all tuples of Sells with beer = 'Blue'...
  - ... to beer = null.
- Update the ('Blue', 'Labatt''s') tuple in Beers
  - Same change as in previous slide...
  - ... results now results in affects Sells tuples having null for the beer attribute.

Choosing a policy

- To choose reject, cascade, or update...
- ... we do so at table creation time ...
- ... and also select specific policy for action (i.e., update vs. delete).
- Follow the foreign-key declaration with **on [update, delete] [set null cascade]**
  - Two such clauses may be used
- With no such clause, default (i.e., reject) is used.

Example: specifying policy

```
create table Sells (
    pub    char(20),
    beer   char(20),
    price  real,
    foreign key(beer)
        references Beers(name)
        on delete set null
        on update cascade
);
```