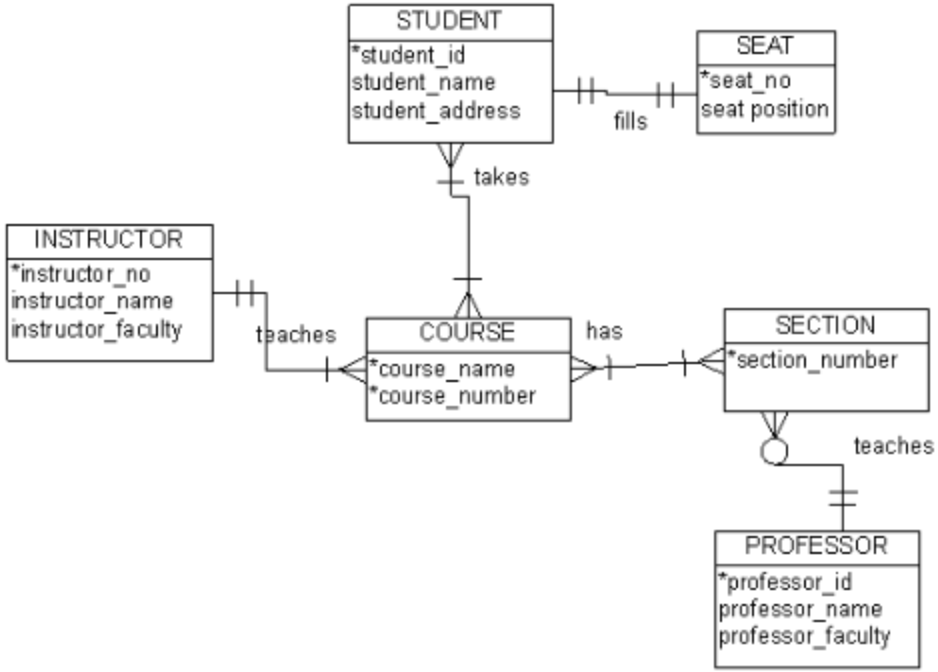
Relational databases : modeling

- Entity-Relationship model, UML is one representation
- Usually, the "Crow Foot" notation is used
- What is "Crow Foot"? It is very similar to UML

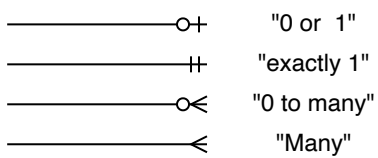
Crow Foot Notation



Credits: http://www2.cs.uregina.ca

Crow Foot Notation (2)

Connectors explanation



Exercice : a patient appointment system

Draw the following system using the crow foot notation

A doctor can be scheduled for many appointments, but may not have any scheduled at all. Each appointment is scheduled with exactly 1 doctor.

A patient can schedule 1 or more appointments. One appointment is scheduled with exactly 1 patient. An appointment must generate exactly 1 bill, a bill is generated by only 1 appointment.

One payment is applied to exactly 1 bill, and 1 bill can be paid off over time by several payments. A bill can be outstanding, having nothing yet paid on it at all. One patient can make many payments, but a single payment is made by only 1 patient.

Some patients are insured by an insurance company. If they are insured, they can only carry insurance with one company. An insurance company can have many patients carry their policies. For patients that carry insurance, the insurance company will make payments, each single payment is made by exactly 1 insurance company.

credits : http://www2.cs.uregina.ca

Toward implementation : a patient appointment system

Create the Tables corresponding to the diagram you just designed

You will probably have to add attributes to entities, like in the following exemples

Remember : CREATE TABLE syntax

```
CREATE TABLE DOCTORS (
    doc_id integer NOT NULL,
    doc_firstname varchar(255) NOT NULL,
    doc_lastname varchar(500) NOT NULL
);
```

Modify table structure : ADD or DELETE columns

Syntax to modify a table : `ALTER`

ADD COLUMN

```
ALTER TABLE DOCTORS ADD COLUMN "bdate" DATE;
```

DROP (REMOVE) COLUMN

```
ALTER TABLE DOCTORS DROP COLUMN bdate;
```

Modify table structure : duplicate columns

ADD COLUMN

```
ALTER TABLE DOCTORS ADD COLUMN "newCol" DATE;
UPDATE DOCTORS SET newCol = bdate;
```

Creating constraints : primary and foreign keys

In order to keep database integrity, some "keys" are necessary to implement:

- It is necessary to add the **primary key** concept to ensure **uniqueness** of the targeted tuple. Primary keys will avoid duplicates to be created and ensures uniqueness while doing modifications queries
- To reference external relations, it is necessary to add the **foreign key** concepts. Foreign keys will be checked while removing some linked tuples to avoid integrity issues (foreign keys pointing to nothing)

Creating constraints : syntax

PRIMARY KEY

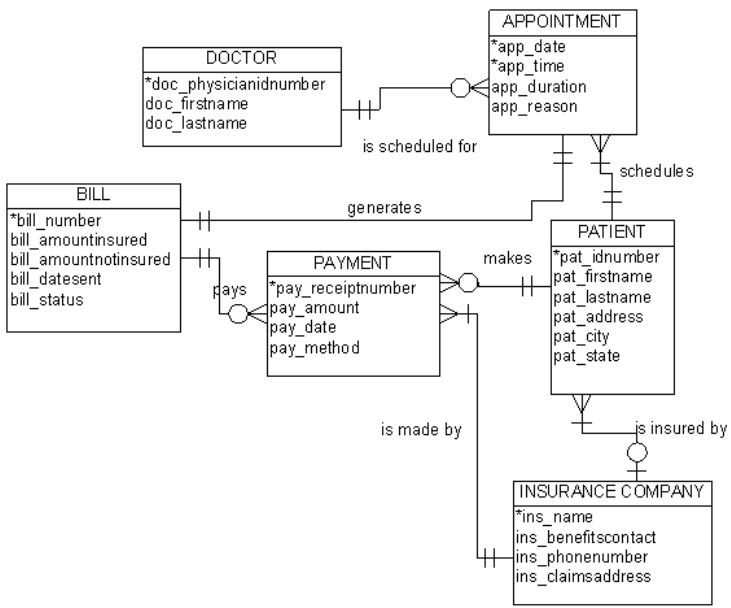
```
ALTER TABLE ONLY DOCTORS ADD CONSTRAINT doc_pk PRIMARY KEY (doc_id);
```

FOREIGN KEY

```
ALTER TABLE ONLY APPOINTMENTS ADD CONSTRAINT fk_doc FOREIGN KEY (f_doc_id) REFERENCES DC
```

Notice the `ON DELETE SET NULL` meaning that if a doctor is deleted, then the appointment remains with null in the f_doc_id field

excercise : modify your scripts to integrate the constraints from the following schema



credits : http://www2.cs.uregina.ca/