# Crow�s Foot Notation

A number of [data modeling](#) techniques are being used today. One of the most common is the [entity relationship diagram](#) (ERD). Several ERD notations are available. For the purposes of CS270 we will be using Crow�s Foot Notation[b1].

Components used in the creation of an ERD:

Entity � A person, place or thing about which we want to collect and store multiple instances of data. It has a name, which is a noun, and attributes which describe the data we are interested in storing. It also has an identifier, which uniquely identifies one instance of an entity. The attribute which acts as the identifier is marked with an asterisk.
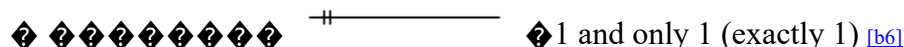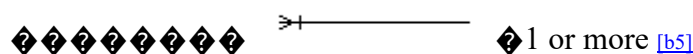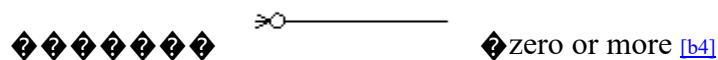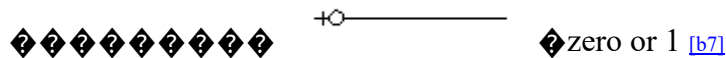
�

```
┌─────────────────┐
│      DESK       │
├─────────────────┤
│ *desk_id        │
│  desk_colour    │
│  desk_size      │
└─────────────────┘
```
���������� [b2]

Relationship � Illustrates an association between two entities. It has a name which is a verb. It also has cardinality and modality.

��������   is assigned
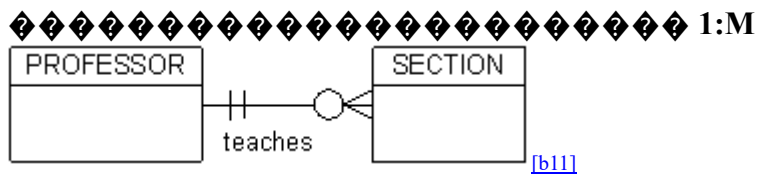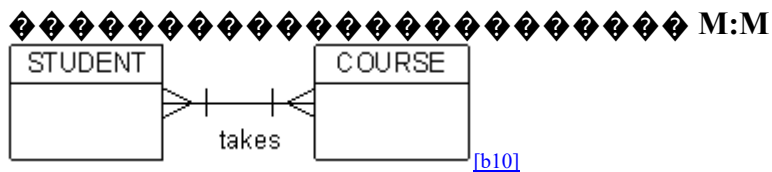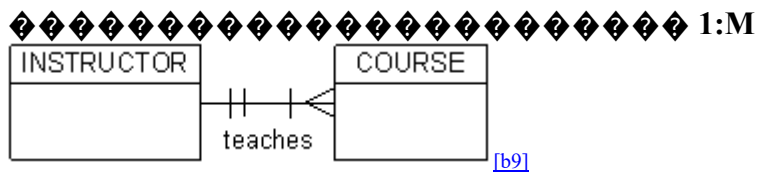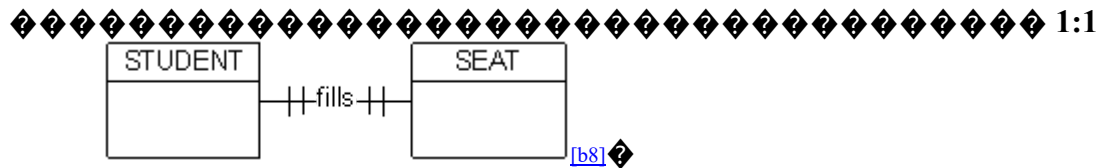   ────────────── [b3]
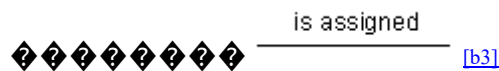
Cardinality and **Modality** are the indicators of the business rules around a relationship. Cardinality refers to the maximum number of times an instance in one entity can be associated with instances in the related entity. Modality refers to the minimum number of times an instance in one entity can be associated with an instance in the related entity.

Cardinality can be 1 or Many and the symbol is placed on the outside ends of the relationship line, closest to the entity, Modality can be 1 or 0 and the symbol is placed on the inside, next to the cardinality symbol. For a cardinality of 1 a straight line is drawn. For a cardinality of Many a foot with three toes is drawn. For a modality of 1 a straight line is drawn. For a modality of 0 a circle is drawn.

������   �zero or more [b4]

�������   �1 or more [b5]

� ��������   �1 and only 1 (exactly 1) [b6]

��������   �zero or 1 [b7]

Cardinality and modality are indicated at both ends of the relationship line. Once this has been done, the relationships are read as being 1 to 1 (1:1), 1 to many (1:M), or many to many (M:M).

����������������������������������� **1:1**

STUDENT —||fills||— SEAT [b8]�

�������������������������� **1:M**

INSTRUCTOR teaches COURSE [b9]

������������������������ **M:M**

STUDENT takes COURSE [b10]

������������������������� **1:M**

PROFESSOR teaches SECTION [b11]

Typically, ERDs are much more complex than this, involving quite a number of entities and relationships. If we join all of the above relationships together and add a few attributes, a small collection of data might be depicted in the following way using Crow�s Foot Notation:
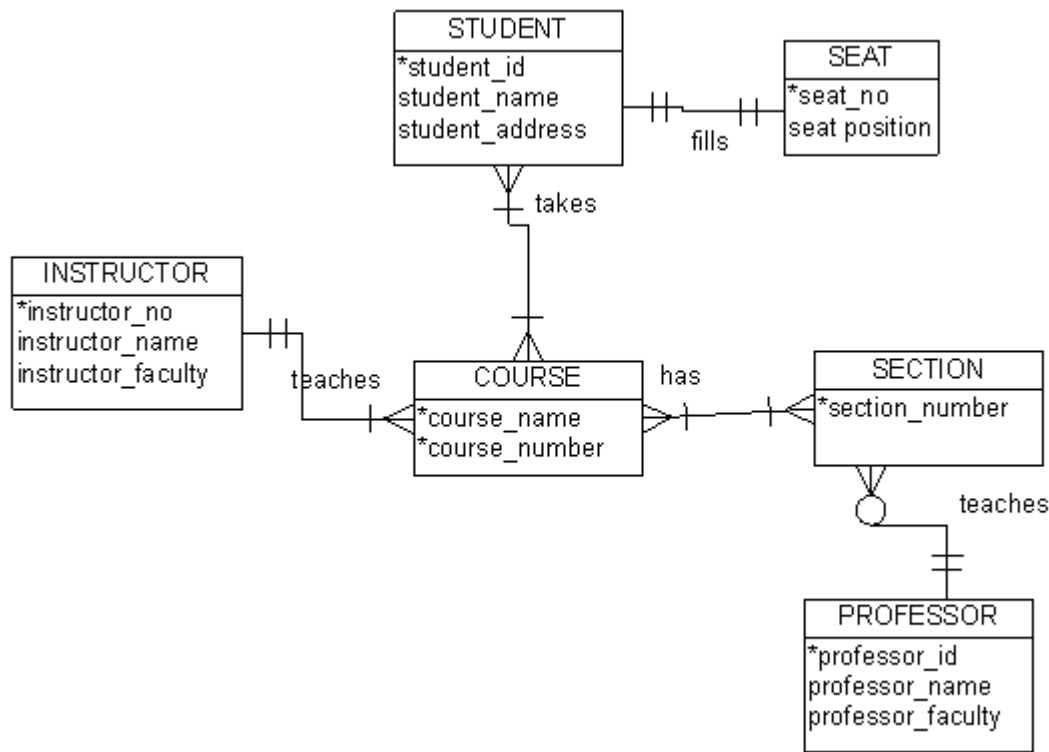
**Consider the following business rules for a patient appointment system:**

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 compnay 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.

Given the above information, the following ERD can be drawn:
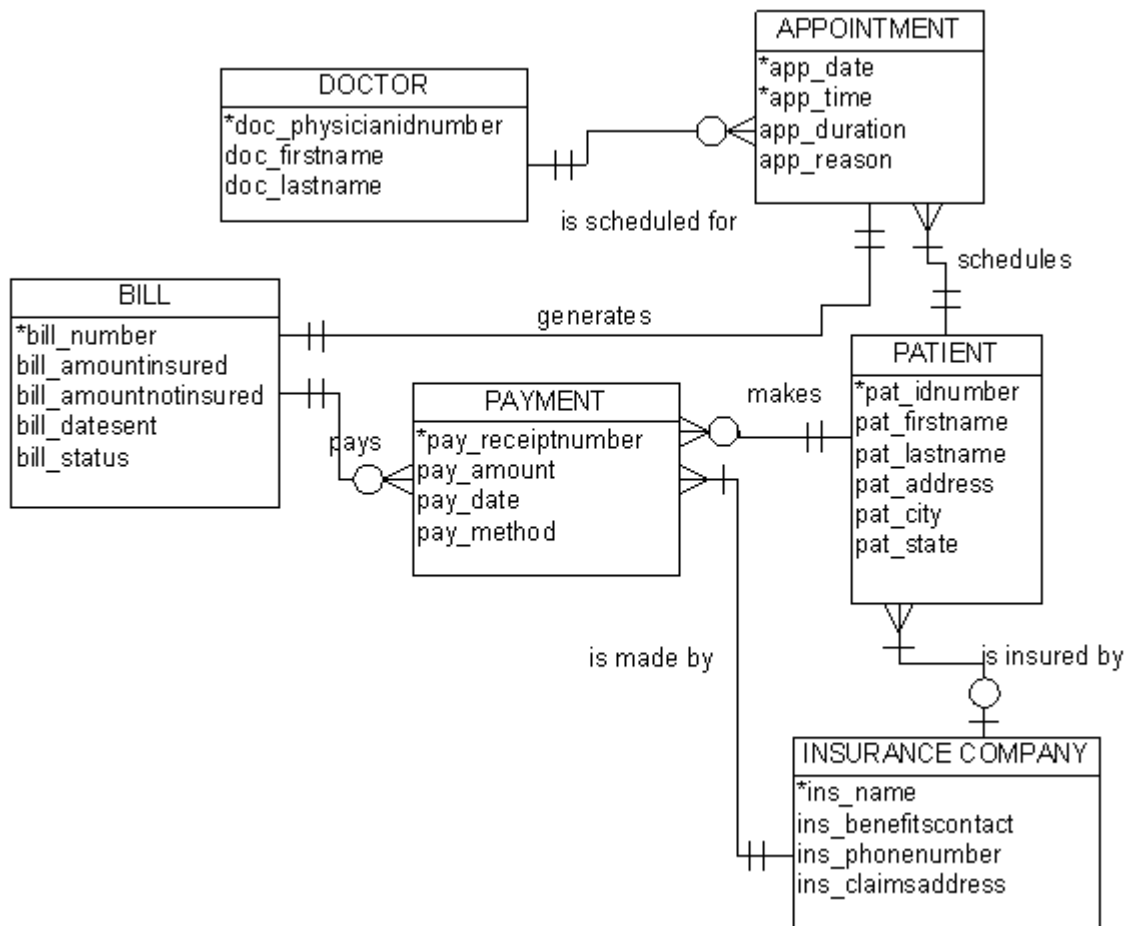
Intersection Entities

Intersection entities are used in the resolution of a many to many relationship. This resolution is done in order to store additional information which doesn�t fit into the attribute list of either entity in the M:M relationship. For instance, in the instructor-course example given earlier, there is a M:M relationship between Course and Section. A course can have many sections, and a section can have many courses. If we want to know how many students are registered in a particular class, we cannot store the information on either entity. We could store the total number of students registered in all sections of a particular course, in the course entity, ie: 300 students are registered in CS270, all sections included. We could store the total number of students registered in all courses with a particular section number, in the section entity, ie: 25,000 students are registered in a section numbered 002. If we want more practical information, such as how many students are registered in section 002 of CS270, we need another entity in which to store the information. So, we create an intersection entity, and place it between its related entities with the additional attributes. The M:M relationship becomes two 1:M relationships. See resolution below.
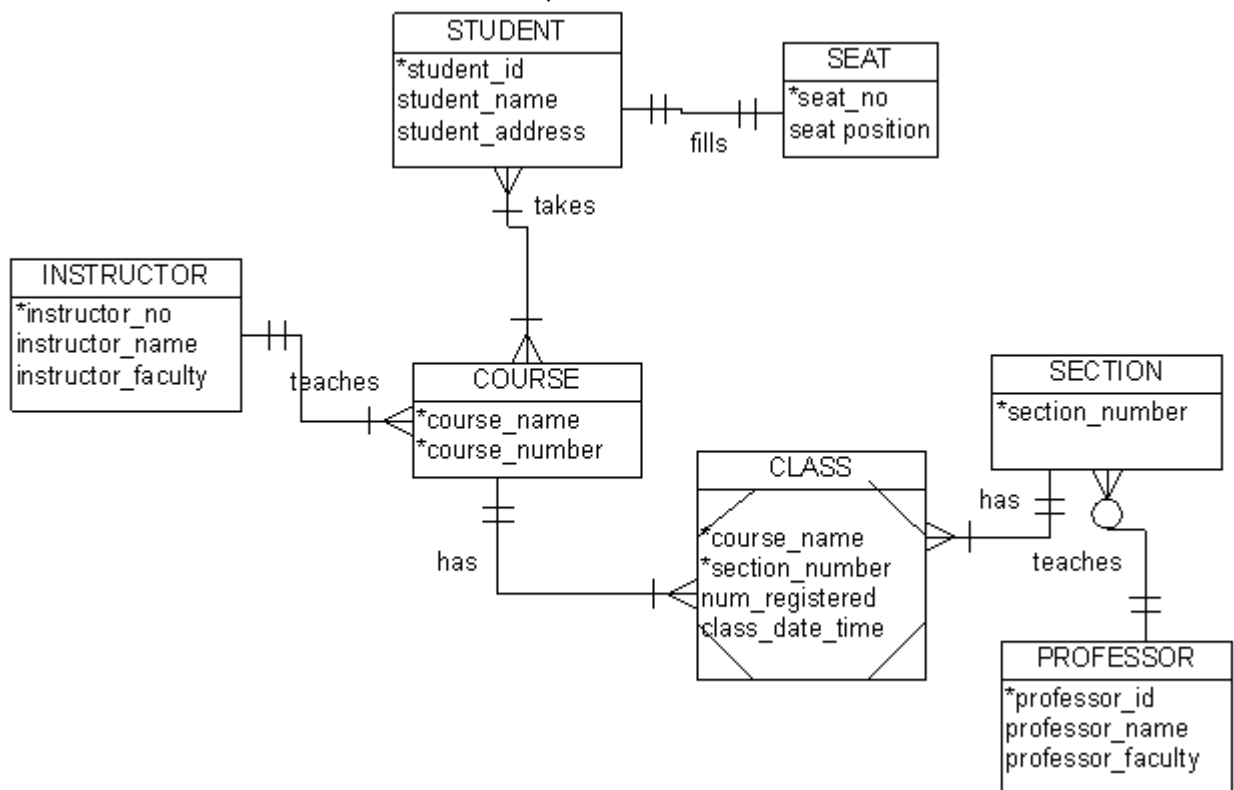
Independent Entities

Independent entities are those which can exist without another entity. Student, seat, instructor etc. are examples of independent entities.

Dependent Entities

Dependent entities rely upon the existence of other entities. Some of their attributes are taken from the identifiers in the entities upon which they depend. Class is an example of a dependent entity. It relies upon course and section, and gets two of its attributes from their identifiers.

[b1]The crow's foot notation was invented by Gordon Everest, who originally used the term "inverted arrow" but now just calls it a "fork". For cardinality, a fork or crow's foot intuitively indicates "many", by its many "toes".

[b2]This is an entity named DESK. We want to store information about several desks. The information we want to collect and store are the desk id, colour of the desk, and desk size. These are the attributes of the entity. Desk_id is the identifier. Given the id of a desk, we would be able to seek out one specific desk, as no two would have the same id number.

[b3]This ia a relationship named �is assigned�. It is shown by a line which normally connects two entities together. No cardinality or modality has been assigned yet.
[b4]cardinality is many, modality is zero
[b5]cardinality is many, modality is 1
[b6]cardinality is 1, modality is 1
[b7]cardinality is 1, modality is zero
[b8]A student fills a seat. This is a 1:1 relationship. 1 student can fill a maximum of 1 seat. 1 seat can be filled by a maximum of 1 student. Each side of the relationship has a

cardinality of 1. The modality on each side is also 1. A student must fill at least 1 seat, and 1 seat must be filled by at least one student. Although it may seem possible to have an empty seat, in which case the modality would be 0, the business rules we have defined determine that for the purposes of our database � we do not want to be able to store information about empty seats.

[b9]An instructor teaches a course. This is a 1:M relationship. One instructor can teach many courses, but one course can only be taught by one instuctor. The modality is one on both ends of the relationship. According to the business rules we have defined � an instructor must teach at least 1 course, and a course must be taught by 1 instructor.

[b10]A student takes a course. This is a M:M relationship. One student can take many courses, and one course can be taken by many students. For the purposes of the information we want to store, the business rules indicate that a student must take at least one course, and a course must be taken by at least one student. The modaility is 1 on both sides.

[b11]A professor teaches a section of a course. This is a 1:M relationship. One professor can teach many sections of a course, and 1 section of a course can only be taught by 1 instructor. On the modality side of things � a professor does not need to teach a section at all, so the modality is zero on that side of the relationshp. On the other hand, a section must be taught by 1 instructor.