CSCI 403: Databases
9 - Relational Database Design and Entity-Relationship Diagrams

Database Design

The notion that a database should be designed (rather than just thrown together in an ad-hoc fashion) leads to the development of Entity-Relationship Diagrams (ERDs) in around 1976 by Chen.

We can consider three different levels at which design activities occur:

- Conceptual - understanding the data *entities* and *relationships* among them. This is a high-level design which may be helpful for communicating with non-technical stakeholders about the data model. ERDs live at this level. (While ERDs may help with communication, it does require learning a new visual language.)
- Logical - mapping from an ERD to a SQL (or other) DBMS. This is concerned mainly with the realization of a data model into an actual database schema.
- Physical - the bare-metal stuff: where files, indexes, etc. should live on disk, network architectures, etc.

We will only be concerned in this lecture with the conceptual level of design. The following lecture will take us through the logical level with an algorithm that maps an ERD to a relational schema.

ERD Components

The three main components of an ERD are the entities, attributes, and relationships.

Entities - things or objects with independent existence, such as persons, products, companies, courses. Entities are the *nouns* of the ERD.

Attributes - the properties describing an entity.

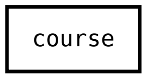
Relationships - the way entities interact or refer to each other. Relationships are the *verbs* of the ERD. For example, a person **supervises** a department, an instructor **teaches** a course, a person **works on** a project.

ERD: Visual Language

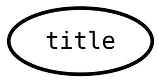
In this section we will develop an ERD for courses at Mines, modeling something very close to the actual tables we have in the database. (The ERD we develop is actually a better data model. The database tables need a slight refactoring.) Along the way we will examine all the elements of the visual language for an ERD.

Entities and Attributes

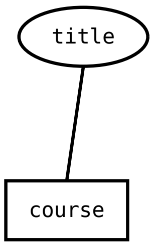
Let's start with an entity. A course is an entity in our database, as something which has a real existence. Entities are visualized using a rectangle containing the name of the entity:



An entity has attributes which describe it; for example, a course has a course title. Attributes are ovals with the name of the attribute:



Attributes are connected to their entity via straight lines:



Attributes may be designated as keys, which has basically the same meaning as a key in the relational model of the database: a key is a value which is unique for all instances of an entity (therefore uniquely identifying an instance). Key attributes have their names underlined.

Attributes may also be composite. A composite attribute represents some property of an entity which is not itself atomic, but which is composed of smaller sub-attributes. For example, a name attribute for a person entity might further break down into first and last names. In the diagram we are developing, the course entity has a composite key, designation, which is composed of course id and section id. Figure 1 shows the full course entity with all of its attributes. The reason in this diagram for the composite attribute is that keys cannot be broken up over multiple attributes in an ERD. Each underlined attribute is assumed to be a key on its own; therefore, a key with multiple attributes must be combined into a composite key attribute.

Attributes may also be multivalued. Multivalued attributes are shown as ovals with a double border; the attribute meetings in 1 is a multivalued attribute. A multivalued attribute is one which may have multiple values for a single instance of an entity. For example, a car entity might have a multivalued attribute for color, since some cars have more than one color; a person might have multiple titles (or even names). In the case of a course, a course at mines may meet in different rooms at different times on different days. I've chosen to model this as a composite multivalued attribute for our course entity.

Other modeling choices are available; for instance, a meeting could be modeled as a weak entity owned by course. A weak entity is one which is not wholly defined except in relationship with another entity. In particular, a weak entity will not have any keys, but may have a partial key. Weak entities are designated using a

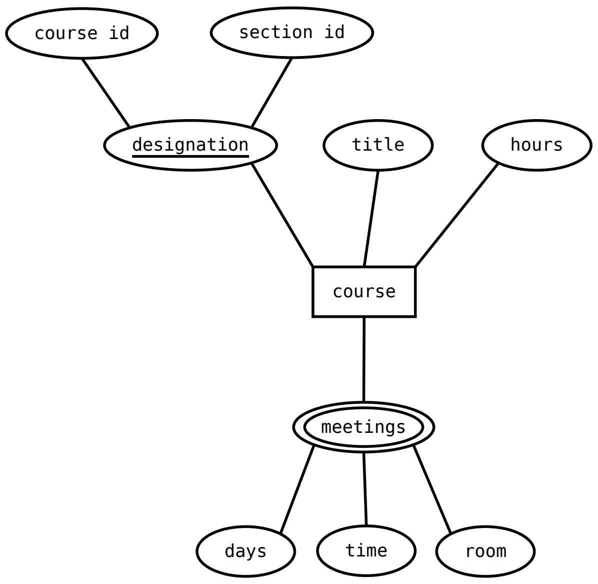


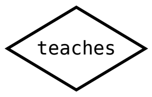
Figure 1: The full course entity

rectangle with a double border (see the section entity in figure 4). We will return to the weak entity a bit later when we refine our model.

Finally, a model may have derived attributes. A derived attribute is one which can be computed from other attributes or components of the model. For instance, age might be derivable from birthdate. A derived attribute is represented as a dashed oval.

Relationships

A relationship is drawn using a diamond shape containing a descriptive relationship name:



Relationships must be attached to two or more entities with straight lines (we focus on the case of two entities for this lecture). The connected entities are the ones participating in the relationship. The teaches relationship connects the instructor entity and the course entity in figure 2.

Relationships may also denote

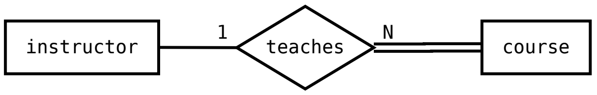


Figure 2: The teaches relationship and its participating entities

cardinality ratios by marking each connection to the relationship with 1 or N (or M). The cardinality ratio gives the maximum number of instances of the entity participating on each side of the relationship. For instance, 1 on the instructor side of the teaches relationship tells us that a course has at most 1 instructor. An N (or M) represents "many"; an instructor may teach 0 or more courses. The possible cardinality ratios are 1:1, 1:N, and N:M. Cardinality may be specified more exactly using ranges (see book for details).

A closely related concept is the participation of an entity in a relationship. An entity is said to have total participation in a relationship if the existence of an instance of the entity depends on there being a related instance on the other side. Total participation is represented using a doubled line connection on the side with total participation, whereas partial participation uses a single line. For example, in figure 2, we see that a course must have an instructor (while the opposite is not true). In some sense, whereas cardinality ratios give us a maximum, participation constraints give us a minimum.

A relationship connected to a weak entity, in which the other side of the relationship is the owning entity, is called an identifying relationship, and is drawn using a double border (see the has sections relationship in figure 4).

Finally, a relationship may also have associated attributes. These are represented using the usual attribute ovals attached by straight lines to a relationship diamond. Relationship attributes represent some properties unique to the relationship which don't properly belong to any of the connected entities. There are no examples of this in our example diagram, but the book gives an example.

Complete Example

A full example based on what we've discussed so far is given in figure 3. The course entity has already been thoroughly described. The instructor entity is fleshed out with attributes; the instructor's name is a key attribute, and other attributes are office and email. An additional entity, department, is shown in relationship with instructor. This relationship is N:M, signifying that an instructor can belong to more than one department, while a department may have more than one instructor within it. (This is a common situation at other institutions; not sure if it happens at Mines.)

The ERD in figure 3 is a fairly accurate reflection of the relations in the course database. However, an initial design is rarely "perfect" - modeling is an iterative process. In particular, this design feels as if it is combining two independent concepts into the course entity; courses and sections. As we will see when we discuss normalization, there are clues to bad design in the database, such as redundancy in some of the fields; for instance, a course should have a course title that doesn't vary by section, but that isn't well reflected in the design, and in the database we see that course title is repeated information - it is redundant over all sections of the course. Ditto for course hours (and description, if we had such, etc.)

This suggests that sections should be separated from courses. Since sections are not truly independent, however, it makes sense to model them as a weak entity owned by a course (no section exists independent of a course). See figure 4 for the final (for now) model.

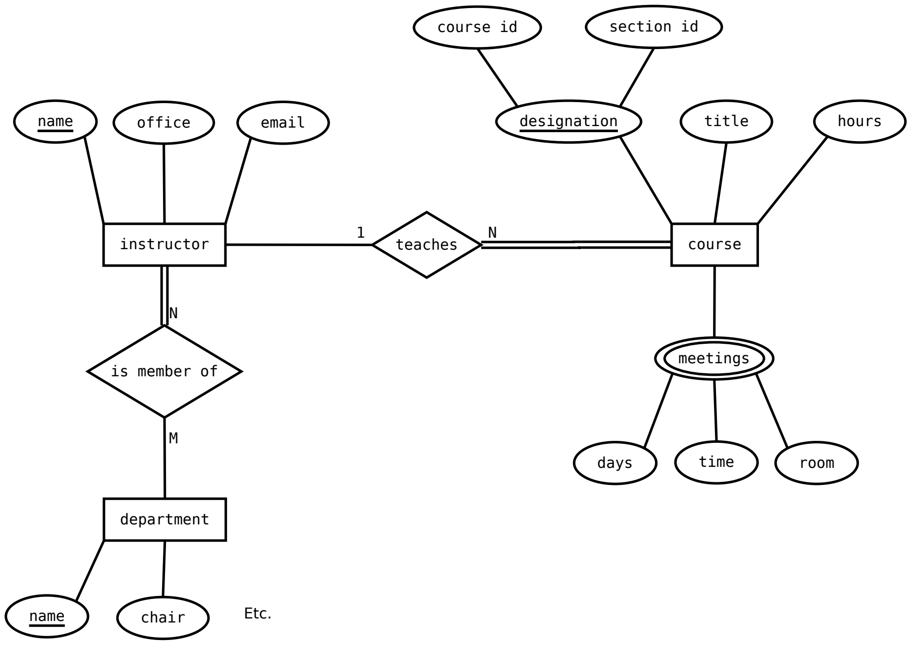


Figure 3: A complete example for a Mines courses database

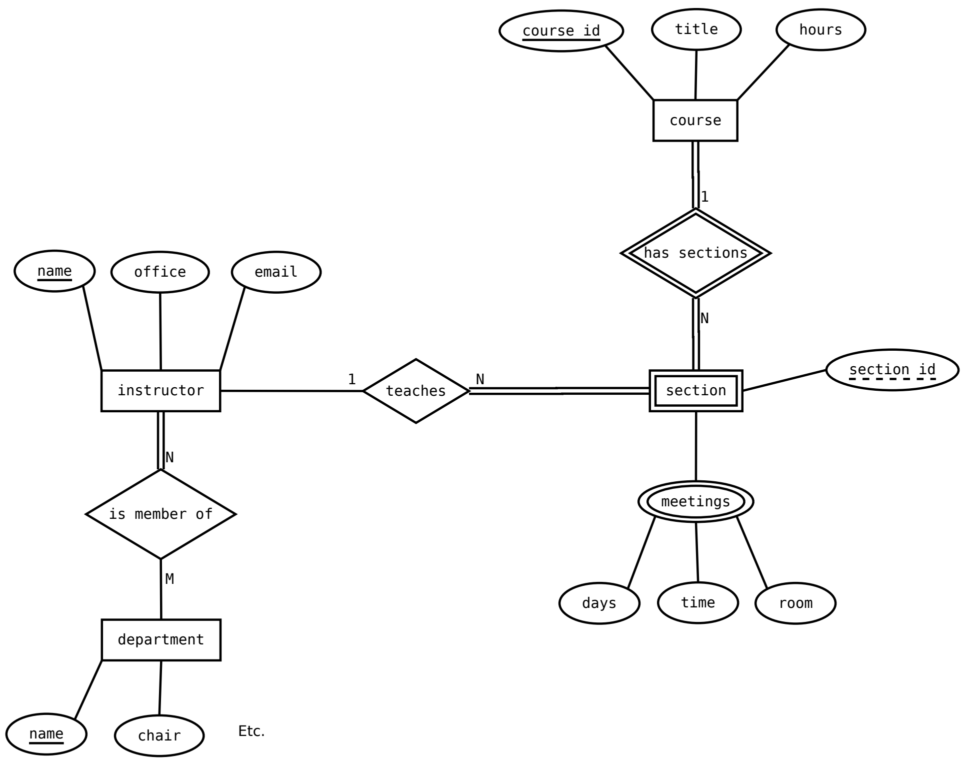


Figure 4: A refined model for the Mines courses database

Final Notes

One thing we didn't address above was the possibility that a relationship can be recursive; that is, an entity can be related to itself. An example from the book is that of an employee entity which also tracks the supervisor information about each employee. Since supervisors are themselves employees, we model this situation with 1:N relationship from the employee table to itself. For further clarification, the lines on each side of the relationship can be annotated with a description of the nature of the relationship, e.g., "Supervisor" and "Supervisee".

Also not covered (except to mention its existence) is the possibility of higher-order relationships, also known as n-ary relationships. These are somewhat rare, although certainly not impossible. An example of this is a relationship between three entities: vendor, part, and project. The relationship represents the situation that a particular part may be supplied by a particular vendor for a particular project at a company. See the book for more discussion of this topic.