DATA MODELS AS LOGICAL STRUCTURES

In this lesson, we'll be answering the following questions:

1. What is a data model, and why is data modeling so important?
2. What the basic concepts of data modeling?
3. What sorts of data models have been used? Which are the most popular?
4. What are the basic ideas beyond the **relational** model, the **entity-relationship** data model, and the **object-oriented data model?**

In some very general sense, every database has two components: a collection of *data,* and some logical *structures* that are used to organize this data. If either one of these components is missing or flawed, the database's ability to turn raw facts into information and knowledge will be compromised. So, for example, the best database in the world won't yield good results if it is filled with false or poorly selected data. This is, unfortunately, often beyond the ability of database designers and administrators to fully correct on their own, as the collection of data is often dependent on people such as business or scientific researchers. By contrast, database designers can have a significant influence on how the data is *structured.* A good structure can help users make the best of incomplete information, and can even help determine how to go about collecting new data. By contrast, bad structure can make it difficult draw conclusions from even the best data.

The process of creating such structures is called **data modeling.** Here, a **model** is simply a simplified (and often visual) representation of some aspect of the world, intended to how different "parts" fit together. Models are intended to help us humans understand how things "hang together." They are NOT intended to catalogue each and every fact we know and are thus always incomplete. As the famous statistician George Box apparently said, "All models are wrong, but some are useful." It's important to note that data models are NOT pieces of computer code. Instead, a good data model should be built *before* any code is written. Data modeling is also a process, where early highly abstract models are slowly turned into more specific and **implementation-ready** models that are ready to be entered into a database.

DATA MODELING AT HOGWARTS

Let's suppose that Hogwarts (the fictional school attended by the characters of the *Harry Potter* novels) wishes to create a database. This database would, among other things, allow for better tracking of which students are in which classes, which instructor is teaching them, the membership of different "houses", and so on. Before making decisions on which database software to use, and entering the data into this software, they would need to construct a data model.

Basic Concepts: Entity, Attribute, Relationship. Every data model (and every database model built on top of it) can be characterized in terms of entities, attributes, and relationships.

1. An **entity** is a person, place, object, evet, or transaction for which you want to store and process data.
   a. At Hogwarts, entities are "nouns" such as Student, Instructor, Course, Room, or House.
2. An **attribute** (also called a **field** or **column)** is a characteristic property of an entity.
   a. Hogwarts's students might have attributes such as first_name, last_name, id, email, etc.
   b. A course might have attributes such as course_title, room, credits, etc.
3. A **relationship** is an association between entities. It is usually expressed by a verb.
   a. Hogwarts students are *enrolled* in classes, while instructors *teach* them. The relations are Enrolled(Student, Class) and Teach(Instructor, Class).
   b. Entity A has a **one-to-many** relationship with entity B if and only entity A can have a relationship with MANY items in B, but each item in B can have this relationship with only ONE item in A. For example, Teaches is one-to-many relationship between instructors and courses. Each instructor can teach MULTIPLE courses, but each course can only have ONE instructor (at least for this example).
   c. Relationships can also be **many-to-one** (the same as one-to-many, with B and A reversed), **one-to-one** (just like it sounds), or **many-to-many** (with entities in A being related to an arbitrary number of items in B and vice versa). So, for example, Enrolled is a many-to-many relationship between Student and Courses: each student can be enrolled in multiple courses, and each course might have multiple students.
   d. In the relational model, most/all relationships should be one-to-many. If there is a one-to-one relationship between A and B, then *either* A or B should get a table, and the other should become an attribute of that table (otherwise, we'd be wasting space!). The relational model can't "naturally" handle many-to-many relationships between A and B. Instead, you'll need to create a **linking** table C that is "between" A and B. In the Hogwarts example below, this is the "enrolled" table.

Business Rules as Constraints on Data Modeling. In order to decide precisely *which* entities, attributes, and relations to include in the Hogwarts data model, the database designers will need to know the **business rules,** which are precise and unambiguous statements of policies, procedures, or rules that say how the different parts of the business (or scientific research project, or video game, etc.) relate to one another. The rules can sometimes be found by reading company handbooks or talking to administrators (in this case, Dumbledore), but will often require looking at "how things are done," and attempting to formulate this as a business rule. Some simple business rules for Hogwarts might be things like "Students take classes, while Teaches receive salaries" or "A House can have many members, but a student can only belong to one House." In constructing data models from business rules, database designers will find that part of their job is *descriptive* (trying to formally capture the way an organization already logically organizes it data), while part is *prescriptive* (they have to decide how to organize data that is currently unorganized).

Question: Think of a topic you might want to make a database for. What are some "business rules" you'd want to take account of? What are some entities that you might want to include in a database? What are the attributes of these entities? The relationships between them?

THE RELATIONAL MODEL

Most modern DBMSs are based on the relational model, and we'll be focusing most of our attention this model. Here are the basic ideas of this model:

1. Each type of entity is expressed as its own **relation** (or **table),** which can be visualized as 2-dimensional collection of "columns" and "rows**.**" The **rows** (or **tuples)** of this table correspond to the different entities of the relevant type about which we have stored data, while the columns represent the values of the attributes. A database **record** (one row) describes one entity and all of its attributes.
2. The tables are **related** to one another via their attributes. So, for example, the Student and Teacher tables may both contain an attribute of course_id, which in turn references the id attribute int the Course table. This allows us to answer questions like "Which students are enrolled in Professor Snape's class?"
3. One-to-one relationships and many-to-one relationships can be captured using attributes. Many-to-many relations will need their own table.

Again, don't worry about the details at this point. The point is just to get a sense of how the relational model "looks"

SELECT * FROM Student;						
id	**last_name**	**first_name**	**dob**	**year_in_school**	**Email**	**house_id**
1	Diggory	Cedric	5/25/1978	5	diggory.dog@gmail.com	2
2	Chang	Cho	3/11/1979	4	Chang23@hogwarts.edu	3
3	Potter	Harry	7/31/1980	3	harry.potter@gmail.com	1
4	Granger	Hermione	9/19/1979	3	hgrangerr@hogwarts.edu	1
5	Malfoy	Draco	10/31/1979	3	DracoM79@hogwarts.edu	4

Here, we have stored information about the student's id numbers, name, email, and house. "id" is the **primary key** of this table, which means (among other things) that each row in the table must have a *unique* id. The attribute house_id serves to *relate* this table to the House table, as pictured below:

SELECT * FROM House;		
id	**name**	**History**
1	Gryffindor	Gryffindor was one of the four Houses of Hogwarts School of Witchcraft and Wizardry and was founded by Godric Gryffindor...
2	Hufflepuff	Hufflepuff was founded by Helga Hufflepuff, and is the most inclusive among the four houses, valuing hard work, dedication, patience, loyalty, and fair play rather than a particular aptitude in its members
3	Ravenclaw	Ravenclaw was founded by Rowena Ravenclaw. Members of this house were characterised by their wit, learning, and wisdom...
4	Slytherin	Slytherin was founded by Salazar Slytherin. House characteristics included cunning, resourcefulness, leadership, and ambition...

Finally, we have tables for Course and Enrolled (I've left the Instructor table out). Enrolled is a **linking table** (or **bridge entity)** between Course and Student.

SELECT * FROM Course;		
id	**credits**	**name**
1	3	Potions
2	3	History of Magic
3	4	Transmutation
4	4	Defense Against Dark Arts

SELECT * FROM Enrolled;	
course_id	**student_id**
1	1
3	1
1	2
2	2

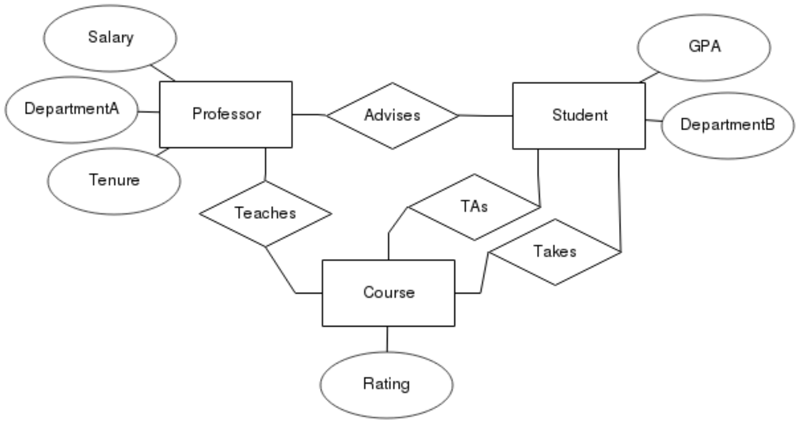
Question: Which students are enrolled in which classes?

E-R DIAGRAMS AS A "FIRST STEP"

The **Entity-Relationship Model (ERM)** is a common model in databases used for high-level design, and is often used as a *precursor* to translating into the relational model just described (though it can also be used with other models). It generally involves the use of visual diagrams (called **Entity-Relation Diagrams, or ERDs)**, with the symbols as follows:

1. Entities (actually, types of entities) are expressed by rectangles/squares, while attributes of these entities are represented by circles.
2. Relationships (which connect two or more entities) are represented by diamonds.
3. Different types of lines are often used to **connect** entities to these relationships, depending on what sort of relationship it (one-to-one, many-to-one, or many-to-many).

Here is a simple E-R diagram from a paper by Hayes et. al[1]:



Here the entities are Professor, Course, and Student. Each entity has associated attributes. For example, a Professor's attributes include tenure (do they have it or not?), the department they are part of, and their salary. Finally, the entities bear different relationships to one another. Students, for example, might *take* a class, or *ta* (serve as "teaching assistant") for that class. Professors, by contrast can *teach* classes. This diagram does NOT contain the sorts of lines necessary to express different sorts of relationships.

Think of one attribute, entity, or relationship that might be added to above diagram. How could add it?

REVIEW QUESTIONS AND ACITIVITIES

1. Define the following concepts in your own words. Do your best NOT to look at the handout while doing so: **data model, business rule, entity,** and **relationship.** Try to give an example of each concept.
2. What are the basic components of the **relational database model?**
3. Business rules must be formulated as clear, short, and unambiguous sentences. What are THREE possible business rules for a school data model (such as the Hogwarts example).
4. Suppose that you were designing a database that contained information about music (for example, about different songs, artists, albums, etc.). Do the following::
   a. Identify at least THREE different entities about which you'd like to store data.
   b. Identify at least TWO attributes for each of your entities.
   c. Describe at least TWO relationships between these entities (bonus: what type of relationships are these?).
   d. Draw an E-R diagram based on your data model
5. Describe the difference between the following types of relationships. Then, give an original example of entities X and Y that fulfill them.
   a. A one-to-one relationship between X and Y.
   b. A many-to-one relationship between X and Y.
   c. A many-to-many relationship between X and Y.

[1]Hayes, Alexander & Das, Mayukh & Odom, Phillip & Natarajan, Sriraam. (2017). User Friendly Automatic Construction of Background Knowledge: Mode Construction from ER Diagrams. 1-8. 10.1145/3148011.3148027.