CSCI 466/566

Databases - ER Diagrams Part 2

Jon Lehuta

Northern Illinois University

September 8, 2017

Entity-Relationship Diagrams, cont.

- Quick Review
- Inheritance
- Things to Watch Out For

Northern Illinois University

Quick Review

We'll start with a review of what we did last time.

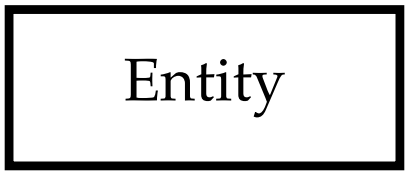
Northern Illinois University

ER Diagram

A way to graphically show the conceptual model of your database design.

Entities

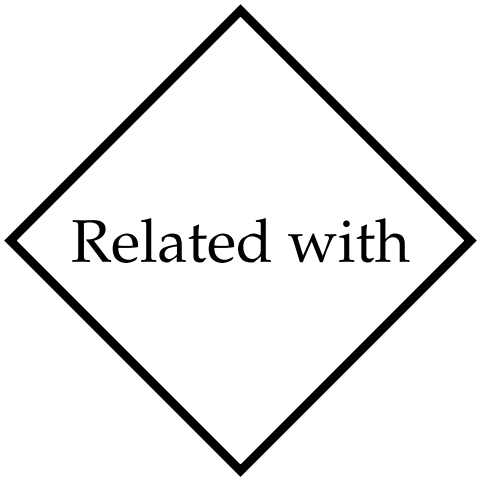
- The principle objects your database is concerned with will generally become entities.
- Shown as rectangles with the name of the entity in them.
- Note once more that this is a container for the concept of the entity, not a single instance of it. If you have an entity called Person, it would encompass all of the people that you need to store data on.



Northern Illinois University

Relationships

- Relationships show how entities work together.
- The relationship is shown as a diamond.
- You cannot have a relationship without at least one entity.



Degree of a Relationship

The degree of a relationship is defined as how many entities it associates. If one entity is associated more than once (such as with a recursive relationship), then the degree counts each time it is referenced.

Connectivity of Relationship

The connectivity of a relationship is written on its "legs". When you connect the relationship to an entity you define the minimum and maximum number of that entity that belongs in the relationship as a whole.

Northern Illinois University

Cardinality of Relationship

Cardinality of a relationship is one of

- one-to-one
- one-to-many
- many-to-many (can be optional or mandatory depending on minimum values)

Attributes

Attributes indicate data to be stored about a given entity or relationship.

- Attributes on an entity will have data for every occurrence of that entity.
- Attributes on a relationship will have data when all of the appropriate entities come together to make the relationship happen. This is called intersection data.

Recursive Relationships

It is possible for an entity to have a relationship with itself. There are two general forms this will take:

- many-to-many - this will represent the connections between members of the entity in a network or graph form
- one-to-many - this will represent the connections between members of the entity in a tree, or hierarchical form

Northern Illinois University

Inheritance

Two types of inheritance available

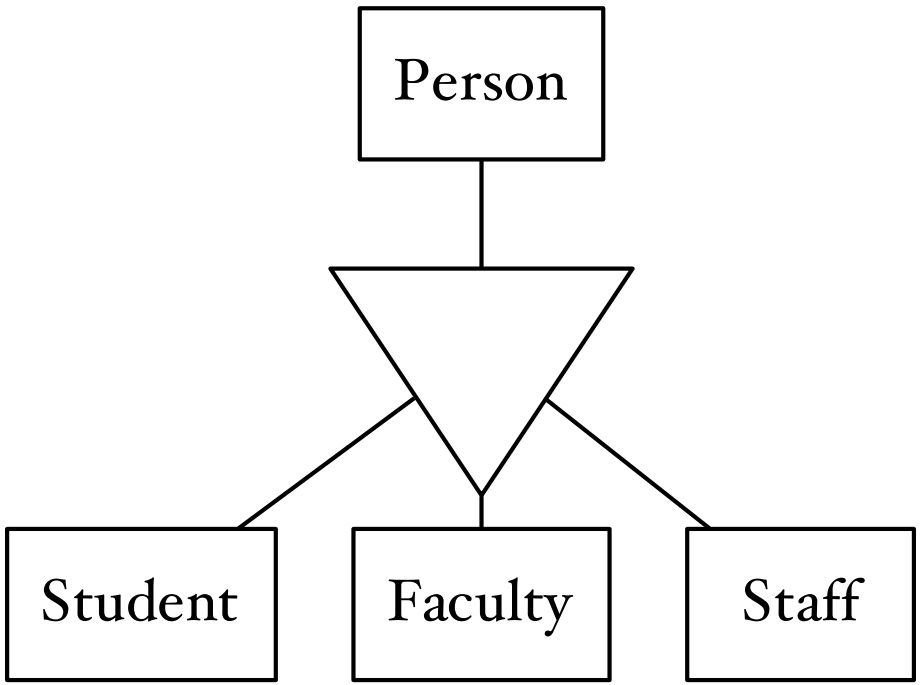
- **"is a"** inheritance. This shows that the subtype IS a member of the supertype.
- **"is part of"** inheritance. This shows that the supertype contains, or is made up of members of the subtypes.

All attributes of the supertype entity are inherited by the subtype entities. The identifier of the subtypes will be the same as the supertype.

Northern Illinois University

IS A Inheritance

- This type of inheritance happens when you have a **supertype** and one or more **subtypes** that are members of the supertype.
- Denoted by an upside-down triangle, with the supertype on top, and the subtypes coming out the bottom.



Defining IS-A inheritance

There are two things you need to choose when using IS-A inheritance:

- Generalization vs. specialization - can the supertype occur without being a member of the specified subtypes?
- Overlapped vs. disjoint subtypes - is it possible for a single occurrence of the supertype to be a member of more than one subtype?

They are mutually exclusive so you need to pick one of each, ie. GO, GD, SO, SD

IS-A inheritance - Generalization

Generalization:

- Supertype is the **union** of all of the subtypes.
- This means that an instance of the supertype **CANNOT EXIST** without belonging to at least one subtype.

IS-A inheritance - Specialization

- The subtype entities *specialize* the supertype.
- This means that an instance of the supertype **CAN** exist without being related to any of the subtypes.

IS-A inheritance - Overlapping Subtypes

Overlapping Subtype Entities

- It is possible for an instance of the supertype to be related to more than one of the subtypes.
- In our example, this would mean that a Person can be a Faculty member and a Student, or a Student and a Staff member, or all 3.

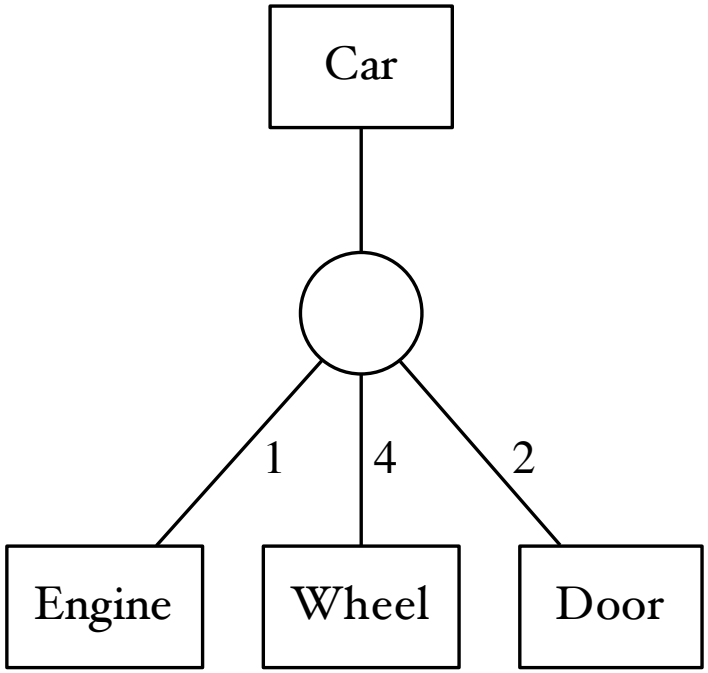
IS-A inheritance - Disjoint Subtypes

Disjoint Subtype Entities

- the subtype entities are mutually exclusive
- it is **not possible** for an instance of the supertype to be related to more than one subtype.
- in the above example, this would mean a Person could be Student, Faculty, or Staff, but **not** any combination of two or more of these.

IS-PART-OF Inheritance

- **"Is part of"** inheritance indicates that the supertype is constructed from instances of the subtypes. It is shown on an ER diagram as a circle, with the supertype on the top, and subtypes on the bottom.
- The diagram to the right shows that a Car is made up of 1 engine, 4 wheels, and 2 doors.



Warning about IS-PART-OF

- The IS PART OF inheritance operator does have its uses, but it is not very commonly used.
- If you see something involving a certain number of things being present, there are several possibilities
  - Sometimes a number is specified that isn't actually important for what we are modeling. This won't even be represented on an ER Diagram. This is the case when changing the number wouldn't have any effect on the necessary structure of a database.
  - If you need a certain number of items for a relationship to hold, you should explore using the connectivity of the relationship to express that.
  - Finally, this IS PART OF inheritance might be useful. It is almost never *necessary*, however.

Are you actually representing what you want to?

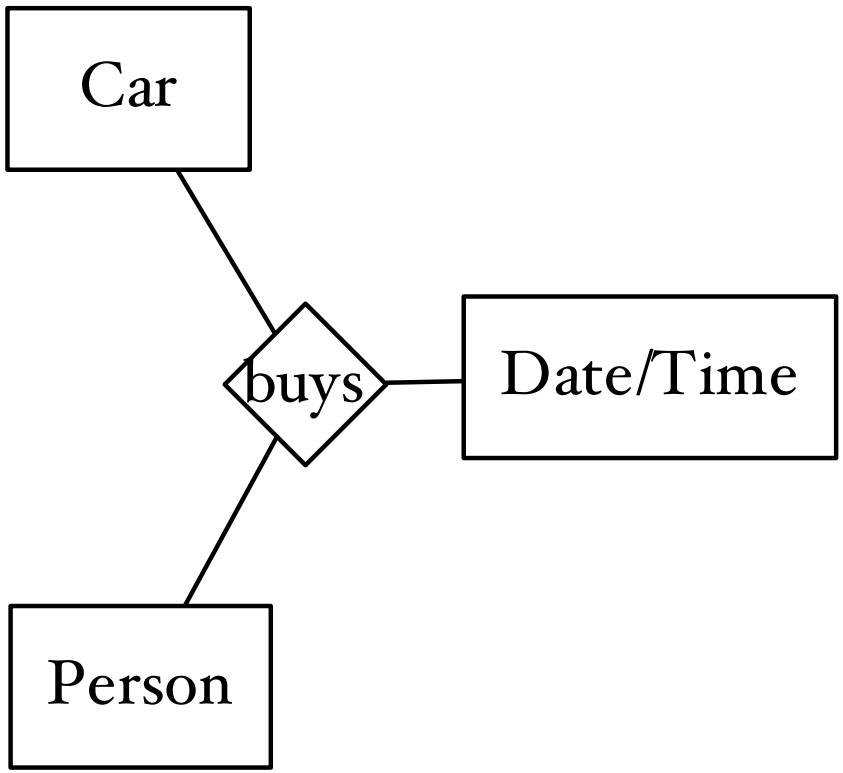
Let's say you're running a business selling used cars. A simple ER diagram for the sales might look like the following:

The resulting database would have one entry for each time a specific person buys a specific car. If the same person buys the same car more than once (obviously selling it to someone else at some point), this model would no longer be appropriate.

Northern Illinois University

Are you actually representing what you want to?

Adding a new entity to the relationship for the date/time of the purchase can fix this problem.



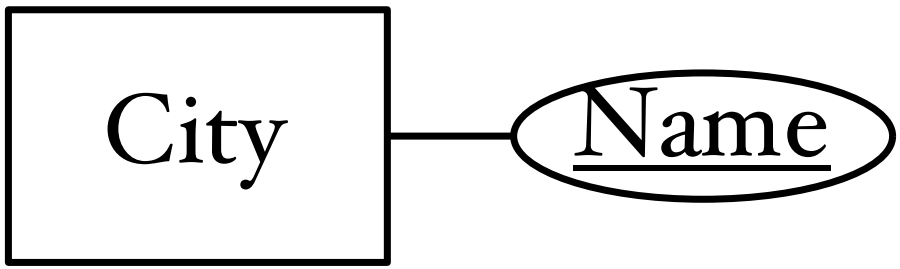
Notice that the connectivities can change when you add new entities to the relationship.

Weak Entities - Introduction

So far, all of the entities we have used have been things that stand on their own. There are some situations where we are modeling an object for which we certainly need to store data, but the items exist only in the context of some other entity. Many of these examples can occur, we will deal with one example on the next page.

Weak Entity - Example Situation

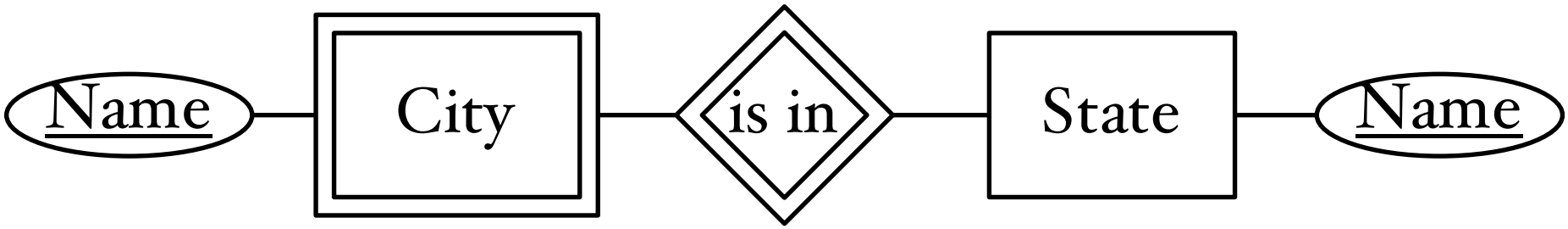
One example of a time that an entity depends on another would be the idea of a city. Within a state, we can generally be assured that cities will have unique names. If we were working only at that level, the City could be an entity as we saw above. A good identifier for it would be the name of the city, so we would see the following:



In some situations, this would be valid. The Name attribute can serve, in those circumstances, as an appropriate identifier.

Weak Entity - Example Diagram

To indicate this sort of dependency, we can make the dependent entity a "weak" entity. This is drawn with a double-edged rectangle, shown below.



Notice that the City entity is now drawn as a weak entity, with a double border. The relationship between the weak entity and the strong entity is also drawn with a double border. The relationship is not weak, per se, but it is used to indicate which strong entity the weak entity depends upon.