ENTITY TYPE HEIRARCHIES AND "ISA"

COMP 1140: Database and SQL | Brendan Shea, PhD (Brendan.Shea@rctc.edu)

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

1. What is the **enhanced entity-relationship model (EERM)?**
2. What is the **isA** relationship between **subtypes** and **supertypes**?
3. How do you draw and interpret **enhanced entity-relationship diagrams (EERDs)?**
4. What are the types of constraints that can be placed on parent entities?

In previous lessons, you learned about the Entity-Relationship (E-R) model of data, and the closely related Entity-Relationship Diagrams (ERD). This sort of modeling is very useful in the early stages of databases design, involving everything from communications with clients (for example, to help discover and clarify business rules) to determining the best choice of DBMS to the final implementation of this database. In this lesson, we'll be taking a look at the **enhanced entity-relationship model (EERM)** and the related **enhanced entity-relationship diagrams (EERDs),** which provide ways to capture additional "semantic content" (that is, they help us capture more about what is *meaningful* about our data).

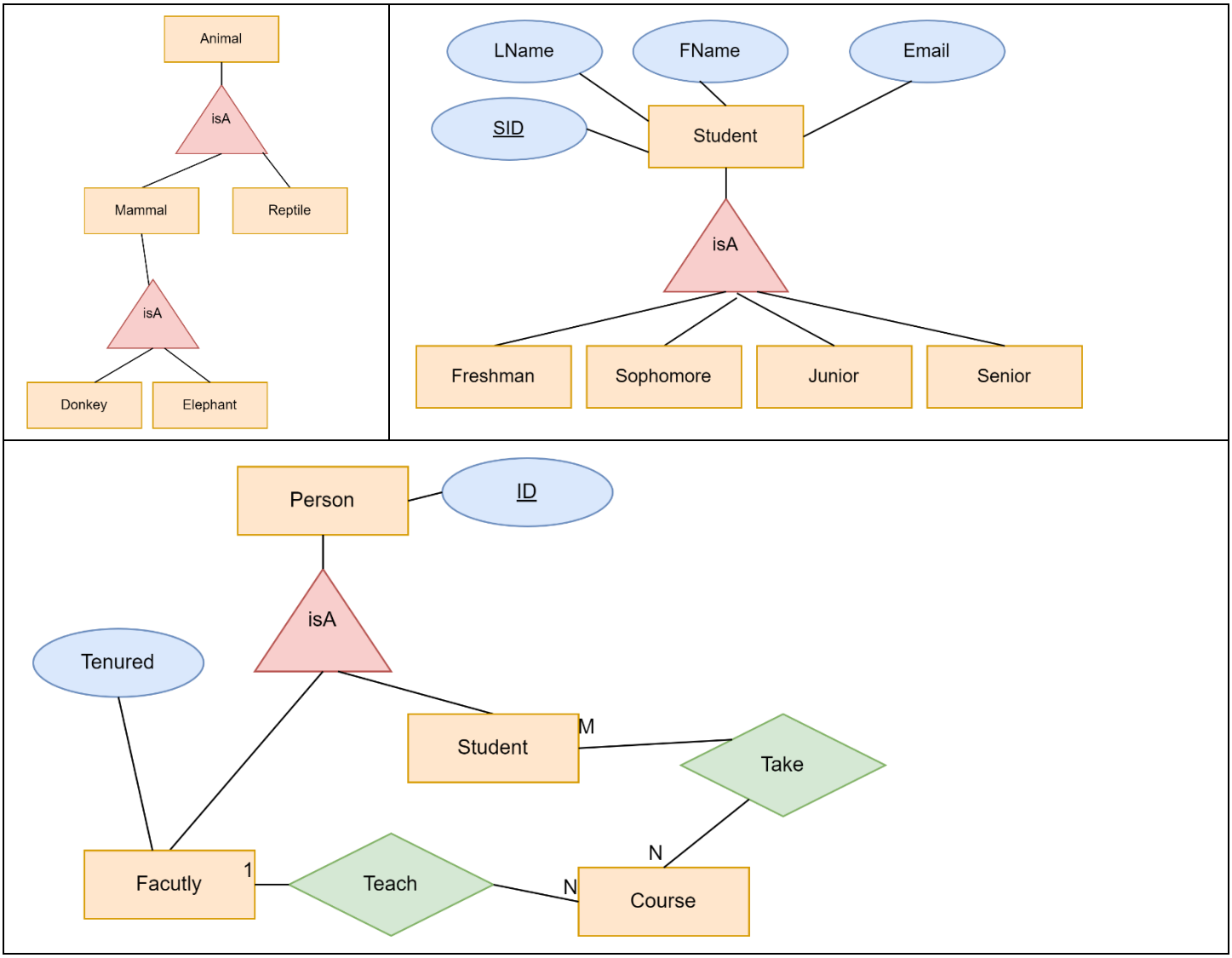
We'll specifically be focusing on the ways that the EERM allows us to capture and describe "specialization hierarchies" and "inheritance" relationships between different entity types. This is closely related to both the object-oriented data model (for databases) and the object-oriented programming paradigm that underlines languages like Java or C++ (so, some of you may have seen similar ideas before!).

INTRODUCING ISA AND SPECILIZATION HEIRARCHIES

What is isA? The important extension of the EERM is the explicit inclusion of the **isA** relationship, which allows us to capture the logical relationships between **subtypes**("children") and **supertypes** ("parents")**.** The isA relationship has a few important characteristics:

1. It has the form "subtype isA supertype." For example, "Elephant isA Mammal" or "Mammal isA Animal." As you might guess, isA is **transitive,** meaning that entities not only inherit properties from the direct "parents" but from the parents of these parents. So, we can infer that "Elephant isA Animal."
2. A subtype **inherits** all of the attributes and relationships of the supertype. So, for example, if we define "Freshmen" and "Sophomores" to be the subtypes of the more general class "Student", and all students have the attribute "Email address" and a relationship with a "Guardian" then Freshmen and Sophomores will have these attributes/relationships as well.
    a. Subtypes ALWAYS inherit their primary key from their supertype. So, if Student has a primary key of StudentID (or something of the type), this will also be the primary key for any subtypes of students.
    b. At implementation level, subtypes are in a "one-to-one" relationship with their parent entity. This means, for example, that any given row in the supertype table (for example, we have an entry for a Person named "Mario") should correspond to at most *one* row in a subtype table (for example, in the Contractor table, which is a subtype of Parent, Mario should appear at most once, if he is a in fact a contractor).
3. The parent entity will have **subtype discriminator** identifying subtype of instance. So, for example, if our supertype is Animal, it may have an attribute named Species. The *value* of this attribute will then serve to identify the relevant subtype. That is, if Species = "Elephant," this indicates that this is of the subtype Elephant.
4. A subtype may ALSO have attributes and relationships that are unique to it (and are not inherited from its parent). So, for example, if a university database defines "Faculty" to a subtype of "People" it may nevertheless be that Faculty have unique attributes or relationships (for example, they are in a "Teach" relationship with Courses, and have a "tenured" attribute).

Example of EERM Diagrams (Chen Style)



SPECIALIZATION HEIRARCHIES: WHY AND HOW?

A **specialization hierarchy** is the name we give to a collection of supertypes and subtypes (such as those illustrated above). We create such hierarchies for the following reasons:

1. "Supertypes" allow us to avoid duplicating information when possible, including metadata. So, for example, if want to store first and last name for all Persons, it makes sense to have some sort of common database table, rather than recreating these exact same attributes for each and every "type" of person.
2. "Subtypes" allow to avoid the **proliferation of nulls** that would occur if some attributes of Person (or whatever the supertype happens) to be are only relevant to certain subtypes. So, for example, it would be wasteful to create *an entirely new entity type* for Faculty (they are just People, after all!). However, Faculty have at least *some* attributes or relationships (such as Teaches or Tenured) which we don't want to encode for each and every Person, at these will be irrelevant to most people (so, they would be "null" for every person that wasn't a faculty).

In keeping with this, we should only create subtypes when we have good reason to: namely, that two or more entities share *some but not all* of the relevant attributes or relationships that we'd like to capture in our data model. We can then model the shared attributes/relationships as belonging the supertype, and the unique attributes/relationships as belonging to the subtype.

Entity Clusters. In the design phase (before we actually go about trying to implement the database on a particular DBMS), we may also include **entity cluster**, which is an abstract/virtual object meant to "stand in" for many subtypes. The entity cluster won't actually be entered in the database (just the subtypes) will, but it can make the modeling/communication process much easier.

Building Hierarchies: Generalization vs Specification. In order to build a specialization hierarchy, one can adopt several methods:

- **Generalization** proceeds from the "lowest", most specific entity types, and then generalizes on these by identifying the *shared attributes and relationships* among these entity types. So, for example, you might "generalize" by creating the entity types for Student and Professor and then generalize to a Person super-type that includes their shared attributes/relationships (such as email address, first name, last name, etc.).
- **Specification** proceeds from the "highest", most general entity type, and then makes this more specific by determining which attributes/relationships are *unique* to different subtypes. This is essentially the inverse of generalization. So, we might BEGIN with a Person entity, and then create the subtypes Student and Professor by identifying the ways in which these differ from one another (e.g., Professor have Salaries, Students have GradePointAvg).

In the specialization hierarchy, a **subtype discriminator** is the attribute of the supertype entity that determines which subtype (if any!) this particular entity is.

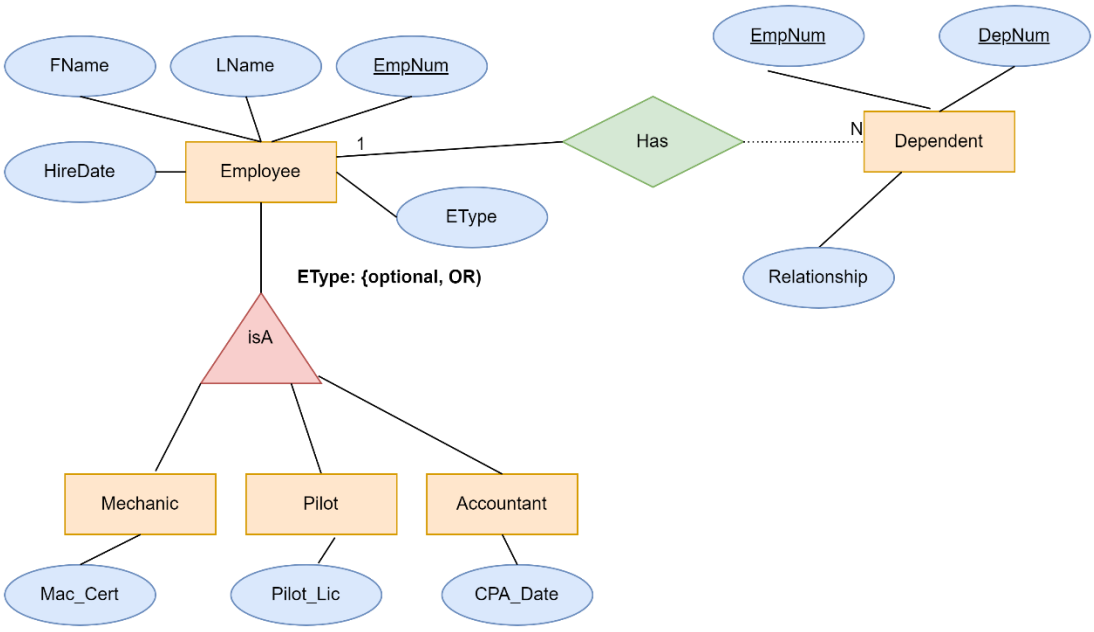
CONSTRAINTS: OR, AND, & (IN)COMPLETENESS

We said earlier that a *subtype discriminator* is an attribute of the supertype that serves to identify which subtype, if any, this entity is. As it turns out, depending on what sorts of *constraints* we place on this subtype discriminator, we can more fully capture the "nature" of the relationship between the supertype and subtype. The most important distinctions are as follows:

Disjoint (OR) vs Overlapping (AND) Subtypes. We say that a subtype discriminator is disjoint (or "OR") if each entity can only belong to ONE subtype. For example, each Animal can belong to only ONE Species. It is either a Lion or an Elephant but not both! By contrast, an Overlapping (AND) subtype discriminator allows for multiple subtypes. For example a single Person might have multiple types of Jobs (the subtype discriminator).

Partial completeness ("Optional") vs Total completeness ("Mandatory") Subtypes. Some supertypes require that an entity have an identified subtype, which cannot be left NULL (for example, you can't have an Animal with no Species!). By contrast, some supertypes do NOT require an entry for subtype. For example, you can be a Person without having a Job.

Here is an example an EERD showing a "partially complete, disjoint" supertype-subtype relationship. In this case, the parent entity is Employee, and the attribute that serves as a subtype discriminator is EType.



REVIEW QUESTIONS

1. In your words, describe the following concepts: (a) supertypes, (b) subtypes, (c) isA relationship, (d) specialization hierarchy, (e) generalization/specification, (f) inheritance, (g) subtype discriminator.
2. Draw the EERD for a specialization hierarchy with at least ONE supertype and TWO subtypes. You should also include:

    a. At least two attributes for the supertype, including the subtype discriminator.
    b. At least one attribute per subtype which are DIFFERENT from either the supertype (and from each other).
3. Give an original example of a supertype / subtype combination with the following characters:
    a. Overlapping/Complete (must have at least one subtype, but can have multiple)
    b. Disjoint/Complete (must have exactly one subtype)
    c. Overlapping/Partial (may or may not have a subtype, can have many subtypes)
    d. Disjoint/Partial (may or may not have a subtype but if it does, has only one).
4. Draw the EERDs for your answers for 3. Include relevant attributes and relationships.