**Chapter 7: Database Design and the E-R Model**

Table of Contents

[7.1 Overview of The Design Process 3](#_Toc49499061)

[7.2 Entity Relationship Model 4](#_Toc49499062)

[7.3 Constraints 5](#_Toc49499063)

[Mapping Cardinalities 5](#_Toc49499064)

[7.5 Entity-Relationship Diagrams 7](#_Toc49499065)

[Complex Attributes 7](#_Toc49499066)

[Redundant Attributes 7](#_Toc49499067)

[Weak and Strong Entity Sets 7](#_Toc49499068)

[Total and Partial Participation 9](#_Toc49499069)

[Minimum and Maximum Values 9](#_Toc49499070)

[Strong and Weak Entity Sets 10](#_Toc49499071)

[7.7 Entity-Relationship Design Issues 11](#_Toc49499072)

[Entity Sets Versus Attributes 11](#_Toc49499073)

[7.8 Extended E-R Features 12](#_Toc49499074)

[Generalization, Specialization and Aggregation 12](#_Toc49499075)

[Constraints on Generalization 13](#_Toc49499076)

[Aggregation 13](#_Toc49499077)

[Representation of Generalization 14](#_Toc49499078)

## 7.1 Overview of The Design Process

To design a database, we must first consider the data needs of the prospective users. Then we can choose a data model, and by applying the concepts of the model, create a schema of the database. A fully developed model would also include the operations that would be performed on the data.

## 7.2 Entity Relationship Model

This model designs the database as relationships between entities, which are individual objects, easily distinguishable from each other. An example of an entity could be a specific person, or a specific company. A set of entities would thus contain entities of the same type that share the same properties (attributes).

A relationship is the association between entities, such as a teacher having the relationship of advisor with a student. A relationship set is the set of relations between any number of entities, but this is usually limited to 2 entities. This number is called the degree of the relationship. An attribute could be associated with a relationship. For example, the advisor relationship could have the start date of the relationship tagged onto it.

## 7.3 Constraints

### Mapping Cardinalities

We can create relationships between two entities in different sets in a few ways. Each of these is known as a cardinality. We could have a 1:1 cardinality, where a single entity from one set has a relationship with a single entity from another set, 1:many cardinality, where a single entity from one set has a relationship with a multiple entities from another set or many:many cardinality, where multiple entities from one set have relationships with multiple entities from another set.

First, consider a 1:many cardinality. This is the most common type, and is done with a simple foreign key. Consider a single entity in the department table, say CSE. This entity could be the foreign key for multiple members of the student table, through the use of a foreign key.

For a many:1 relationship, like between phone numbers and customers (a customer can have many phone numbers), we would need one table that holds customer details, with a primary key like customerID, another table that uses customerID as a foreign key and holds multiple phone numbers per customer. We needed to create another table since we would be unable to hold multiple values for phone numbers in the same table as the customer details. Since the second table uses customerID as a foreign key, the same ID can be repeated multiple times. If we set the phone number itself as the primary key in the second table, we would be able to ensure that no two customers were related to the same phone number.

Next, consider a many:many relationship. This expands upon the idea of using an extra table, but adds to it. Say we have an entity set class and another entity set teacher. Each class has multiple teachers, and each teacher teaches multiple classes. Thus, we could create a new table, called the junction table, that takes both classId and teacherID as foreign keys. In this way, in that new table, we would be able to have multiple techerIDs related to the same classID and vice versa. To ensure no single class-teacher relation is repeated, we would need to create a composite primary key using the two foreign keys.

Finally, consider a 1:1 relationship, like between NID and Passport Numbers. A single NID can have only one passport number associated with it. Thus, we would have to use a foreign key for the passport number with the added constraint that it has to be unique.

## 7.5 Entity-Relationship Diagrams

### Complex Attributes

Attributes can be divided into many categories. There are simple attributes, which are the normal ones, and composite attributes, which contain parts, such as an address which can contain parts within it. The parts are called component attributes. There are single-values attributes and multi-values attributes, with the latter being for things like phone numbers, which can have multiple values. There are also derived attributes, which contain data calculated from other data. Age is an example of a derived attribute, since it can be calculated from a given date of birth.

### Redundant Attributes

Some attributes may appear multiple times over different tables. For example, we could have the department name present along with information about different students. However, since the department name is already part of another table containing department information, we should use it as a foreign key here. If we do not do so, it is considered a redundant attribute, since it is being repeated.

### Weak and Strong Entity Sets

There are some entity sets in which it is not possible to have a primary key. Such an entity set is linked to another entity set that does have a primary key, and is identified through that entity set. The entity set that has a primary key is called a strong entity set, while the one without a primary key is called a weak entity set.

The entity with the primary key is also called the identifying entity for the weak entity. The identification is done with the help of the primary key of the identifying entity, along with some attribute of the weak entity. This attribute is called the discriminator.

ERDs describe the relationships between entities. Entities are shown with rectangles with attributes listed inside them. The primary key is underlined. Relationships are shown using diamonds. Inside an entity, a multivalued attribute is written as { phone\_number } and a derived attribute is written as age().



Sometimes, relationships themselves can have attributes. For example, the above relationship could have an attribute that defines the date at which advising began. This attribute is not a property of the student or the instructor. It is a part of the relationship itself. These are show with a dotted line, like this:



Roles: - Figure out what these are and let me know please.

Cardinality is shown with arrows from the relationship to the entity. A single line means a many relationship, and a line with an arrow at the top means a one relationship. For example, this diagram means that there is one instructor advising every student, but there are many students being advised by a single instructor.



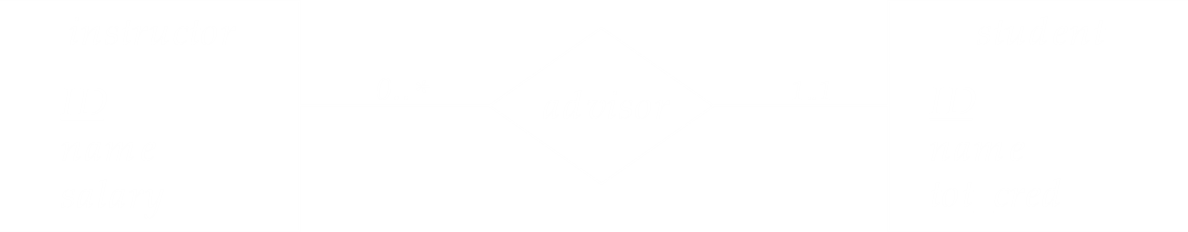
### Total and Partial Participation

Total participation refers to the fact that each and every member of an entity set is taking part in a relationship, as opposed to partial participation, in which every member is not. Single lines like we have seen above indicate partial participation. Total participation is shown using double lines like this:



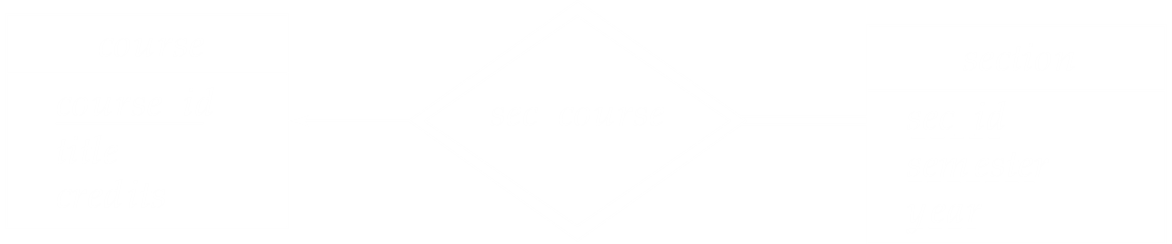
### Minimum and Maximum Values

Minimum and maximum values are shown in the format like this:



### Strong and Weak Entity Sets

Weak entity sets are shown using a double rectangle. The discriminator is underlined with a dotted line. The relationship between a weak entity set and its identifying entity set is shown with a double diamond.



## 7.7 Entity-Relationship Design Issues

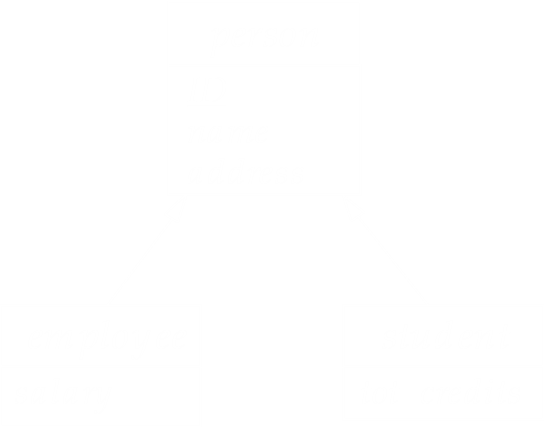
### Entity Sets Versus Attributes

Say we have an instructor entity set with an attribute phone\_number. It can easily be argued that phone\_number should be modelled as an entity set itself and linked using a foreign key. If we do this, then phone\_number would have its own attributes such as whether it is a home phone, a work phone or a mobile phone and the phone number. The key difference in these two approaches is if phone\_number is used as an attribute, then we cannot have more information associated with it. If we want to have more information associated with phone\_number, such as its location, we should model it as a separate entity set. The latter approach is more general than the prior. In contrast, there are attributes that should not be modelled as entity sets, such as name or blood\_group. Modelling these attributes as entity sets would make little to no sense.

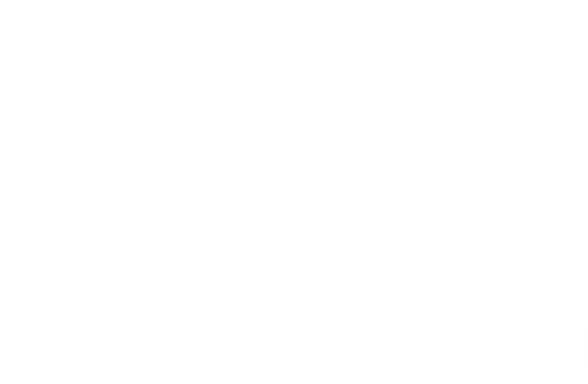
## 7.8 Extended E-R Features

### Generalization, Specialization and Aggregation

It is possible to derive entity sets from other entity sets. For example, we could have a people entity and an employee entity. The employee entity is said to be a more specialized form while the people entity is said to be a more generalized form. This means we could have information in the people entity, such as name, phone etc. that also apply to the employee entity, but is not necessarily a part of that entity. The employee entity has additional attributes. This is similar to inheritance from the C++ language. Generalization is shown with empty headed arrows like this:



We could also have another type of generalization called disjoint generalization. Here, the more specialized forms have no members in common. This is shown like this:

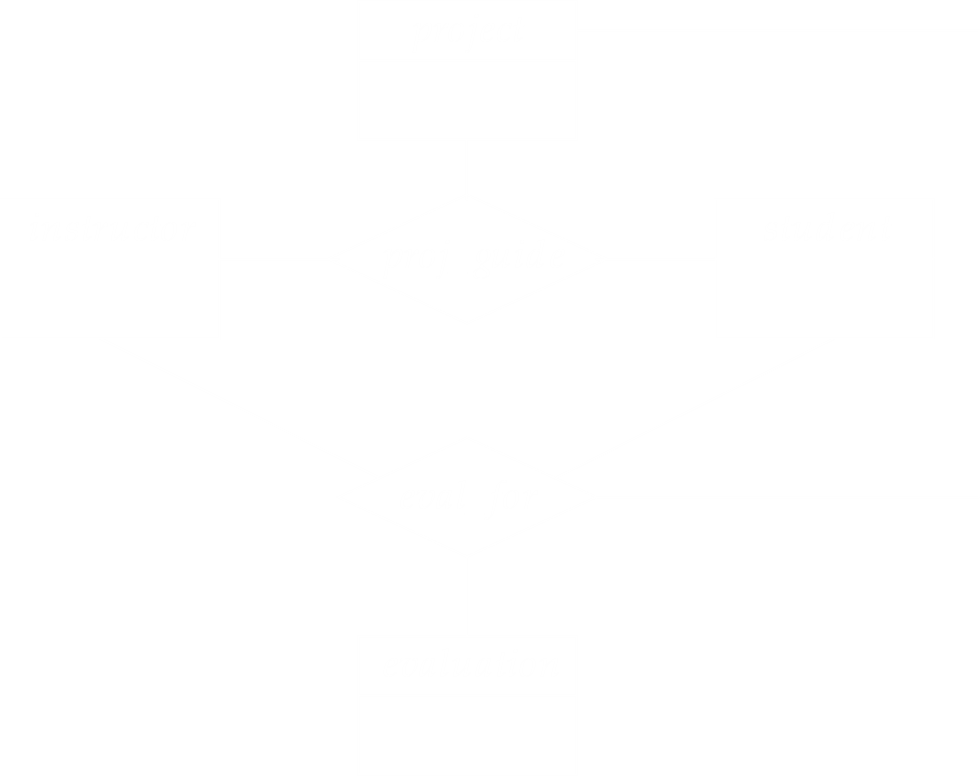


Constraints on Generalization:

* **Condition Defined** – For lower level entity sets, membership is evaluated based on the value of some particular attribute. For example, from the people entity set, if age is over 60, entities go to the senior entity set.
* **User Defined** – The user creates some operation that divides entities based on some decision. The entities are assigned to entity sets on a case-by-case basis.
* **Disjoint** – There are no common members among lower level entity sets.
* **Overlapping** – There are common members among lower level entity sets.
* **Total Specialization** – All members from the higher level go into some entity set in the lower level.
* **Partial Specialization** – Not all members from the higher level are present in the lower level.

### Aggregation

Aggregation is a property similar to generalization. We take a group of entity and relationship sets, and we create a relationship between this entire group and another entity set.



### Representation of Generalization

Entity sets can be made following two methods. First, we could have the common attributes of lower level entity sets in one common higher-level entity set and just link to that. For example, the people entity could contain attributes like name, phone\_num etc. while lower level, specialized entities have specific attributes only like employee\_id­. The second method is to have all attributes present in every lower level entity. This method can only be used if the entities are disjoint and follow total specialization. In this case, the higher-level entity is not created at all.