Entity Relationship (ER) Data Model

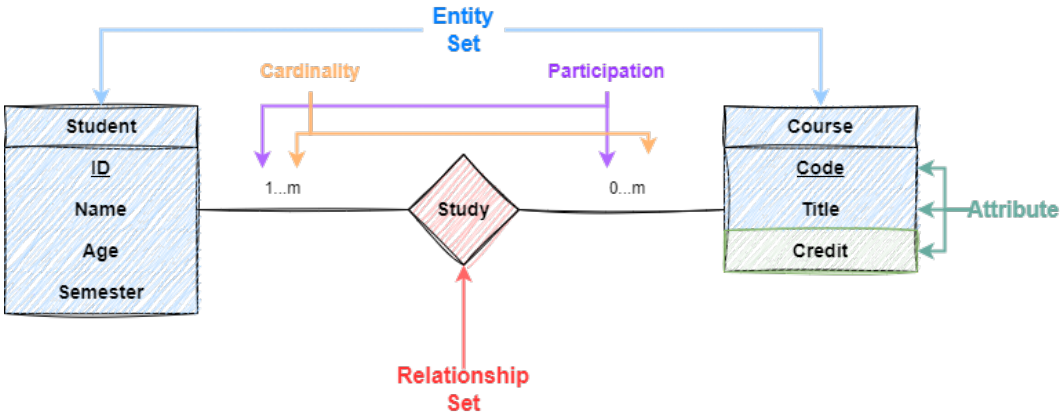
CSE 4308
CSE 4174
DATABASE MANAGEMENT SYSTEMS LAB

OCTOBER 15, 2024

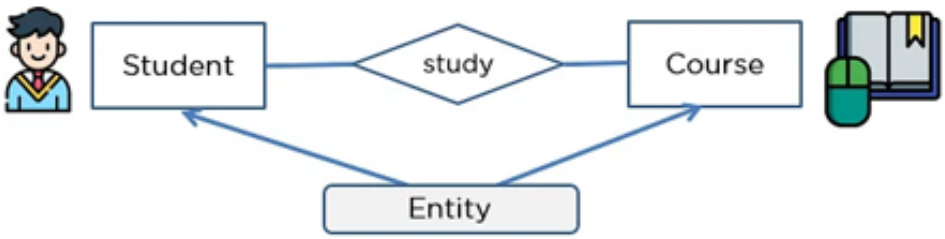
Contents

1 ER MODEL

ER Model or the Entity Relationship Model is the most popular modelling scheme in database design. The architecture of a database is initially planned and designed by using ER diagrams, and then the final design is implemented. Similar to software design, a database designer has to talk to appropriate individuals to get the list of requirements needed for a project. When describing these projects, it is easier to explain things by using diagrams instead of using technical terms like tables, attributes and so on as most people are not familiar with the terms used in computer science. Once all the requirements have been clearly conceptualized using the ER Model, a database designer can proceed to the actual implementation of the project.
Let's look at the things that make up an ER model and we can get a clearer picture on the whole concept



1.1 Entity & Entity Set



An entity is a "thing" or "object" in the real world that is distinguishable from all other objects. For example, each student in a university is an entity.
Entity set is a generic representation or description of an entity. In other words, entity set or entity type is the heading or schema, while entity is an instance of that entity set. Let's consider the example,

Entity set: *Student(ID, Name, Age, Semester)*
Entity: (10101, 'Sam', 24, 'Final')

1.2 Attribute

Attributes are nouns that describe the entities. For example, in the Student entity, the attributes are ID, Name, Age and Semester. Attributes can be of different types depending on the design of the database.

1.2.1 Composite VS Simple-valued Attribute

Composite attributes are those that can be split into multiple parts. For example, the Name attribute can be split into First Name, Middle Name and Last Name. Simple-valued attributes cannot be split further to produce more attributes. For example, the attribute 'First Name' cannot be split further. From this we understand that a composite attribute is essentially a combination of simple attributes.

1.2.2 Single-valued VS Multi-valued Attribute

Some attributes can have multiple values. For example, the attribute 'Phone Number' may have multiple values for an individual user. However, the attribute 'Age' can store only a single value for a particular user. So, Age is a single-valued attribute.

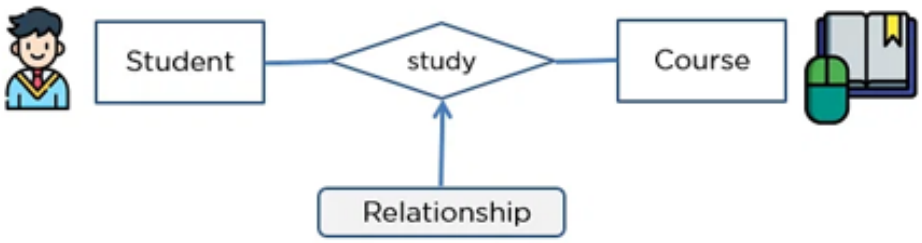
1.2.3 Stored VS Derived Attribute

Sometimes, some attributes may need to be derived from other attributes. Consider the following scenario: You have an attribute called 'DOB' that stores the date of birth of a person. Now, for this DOB, you can calculate that person's Age. Then, Age essentially becomes a derived attribute. On the other hand, stored attributes are basically all other attributes which we store directly in the database. So, DOB is a stored attribute.

1.2.4 Complex Attributes

An attribute can be a combination of different types. For example, the attribute 'Address' can be a composite and multi-valued attribute simultaneously. These attributes are called complex attributes.

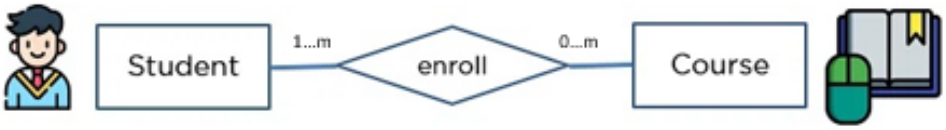
1.3 Relationship & Relationship Set



A relationship is an association among entities. As a database designer, after completing your requirement analysis phase, you will have a description of the requirements at your disposal. The verbs in the description will usually give an overview of the relationship or association that exists among the entities of your project.
Similar to Entity Sets, Relationship Sets are generic representation or description of Relationships. That is, each Relationship is an instance of a Relationship Set.

1.3.1 Degree of Relationship

Degree of a relationship denotes the number of entities participating in a relationship. In our scenario, two entities, Student and Course participate in the relationship. Hence, its degree is 2 i.e. it's a binary relationship.



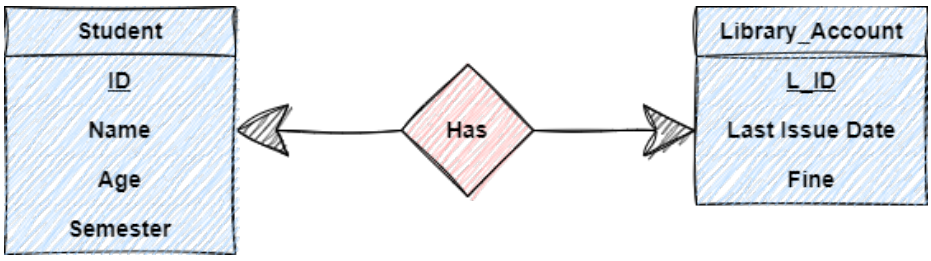
1.3.2 Cardinality

Cardinality denotes the maximum number of relationships an entity can participate in. Let's look at our example scenario to get a clearer picture. Each of the students can take multiple courses. So, the cardinality of STUDENT is many(m). On the other hand, under a single course, there can have multiple students. So, the cardinality of COURSE can be denoted as m(more than 1).
Based on cardinality, we can divide relationships into three categories.

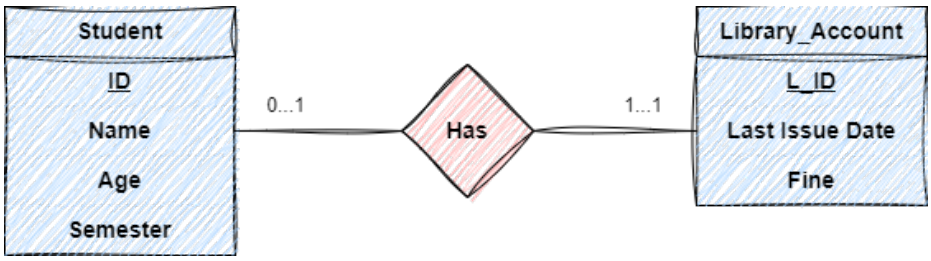
1.3.2.1 One to One Relationship

Let us imagine that we are working with a scenario where we have two entities: Student and Library_Account. 1 Student can have only one library_account and one library_account belongs to only one student. So, there is a one-to-one relationship between Student and Library_Account. In ERD we can represent it in two ways. One is arrow/line notation where the arrow indicates one and the line refers to multiple associations. Or we can follow min...max notation.

Arrow/Line notation:



Min...Max notation:



Generally, for implementing one to one relationship in DDL, we use foreign key in the total participation part. for example for this scenario, the schema for the DDL will be,

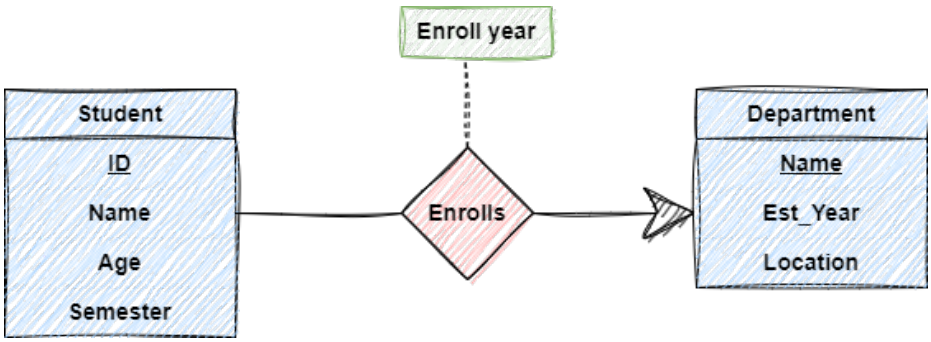
```
Student(ID(PK), Name, Age, Semester)
Library_Account(L_ID(PK), LastIssuseDate, Fine, ID(FK|Student))
```

1.3.2.2 One to Many Relationship

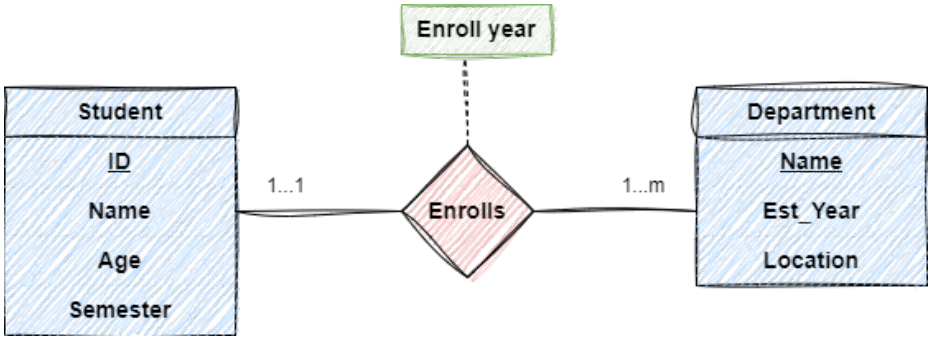
Let us imagine another scenario where we have two entities: Student and Department. A student can enroll under a single department, whereas a department can have multiple students. Here, the

relationship between Student and Department is one-to-many. Let us consider that additionally, we have to store the information of enrollment year too.

Arrow/Line notation:



Min...Max notation:



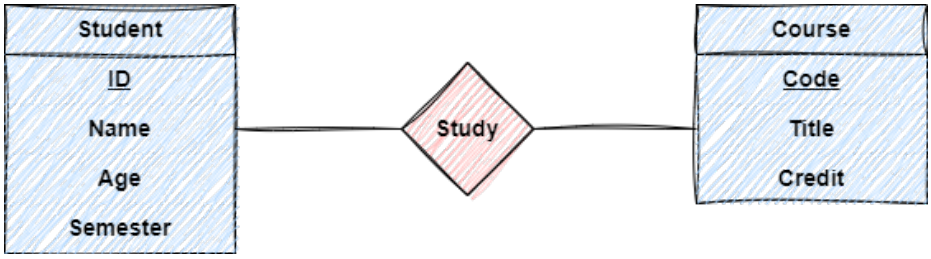
For implementing one to many relationship in DDL, the Foreign key is used in the many entity. for example in this scenario, the schema for the DDL will be,

```
Student(ID(PK), Name, Age, Semester)
Department(Name(PK), Est_Year, Location, ID(FK|Student))
```

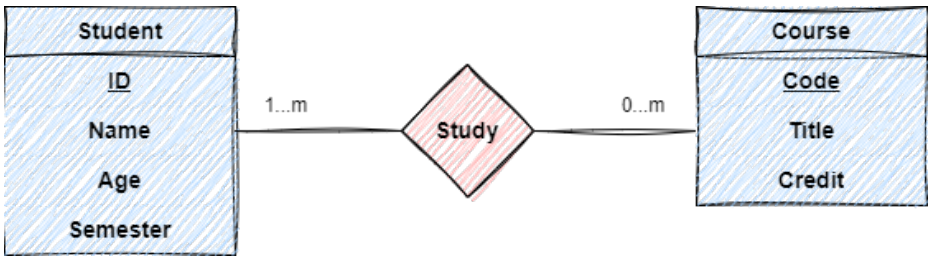
1.3.2.3 Many to Many Relationship

Let us consider a final example where we have two entities: Student and Course. A student can take multiple courses and multiple students can be assigned to a single course. As the maximum cardinality of both Student and Course is M, we can say that there is a many-to-many relationship between them.

Arrow/Line notation:



Min...Max notation:



Unlike the other two relationships, in case of many to many relationship in DDL, there will be a junction table for linking information from both tables,

```
Student(ID(PK), Name, Age, Semester)
Course(Code(PK), Title, Credit)
Study((ID(PK && FK|student), (CODE(PK && FK|student))
```

1.3.3 Participation

Participation denotes the minimum number of relationships an entity can participate in. It is also sometimes referred to as minimum cardinality. In our example scenario, each student must have to take at least one course. So, the participation of Student is 1. However, a course may or may not have any student. So, the participation of COURSE is 0. Hence, the participation of STUDENT in the relationship set is a total participation, whereas the participation of COURSE in the relationship is a partial participation. In Arrow/Line notation we have to draw a double line to represent the total participation whereas in min...max notation min is for describing participation. For example:
Arrow/Line notation: