Database Design

Database design is the process of producing a detailed data model of a database, this data model which can then be used to create a database.

The term database design can be used to describe many different parts of the design , it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views.

The Design Process

The design process consists of the following steps:

1. Determine the purpose of your database - This helps prepare you for the remaining steps.
2. Find and organize the information required - Gather all of the types of information you might want to record in the database, such as product name and order number.
3. Divide the information into tables - Divide your information items into major entities or subjects, such as Products or Orders. Each subject then becomes a table.
4. Turn information items into columns - Decide what information you want to store in each table. Each item becomes a field, and is displayed as a column in the table. For example, an Employees table might include fields such as Last Name and Hire Date.
5. Specify primary keys - Choose each table's primary key. The primary key is a column that is used to uniquely identify each row. An example might be Product ID or Order ID.
6. Set up the table relationships - Look at each table and decide how the data in one table is related to the data in other tables. Add fields to tables or create new tables to clarify the relationships, as necessary.

7. Refine your design - Analyze your design for errors. Create the tables and add a few records of sample data. See if you can get the results you want from your tables. Make adjustments to the design, as needed.
8. Apply the normalization rules - Apply the data normalization rules to see if your tables are structured correctly. Make adjustments to the tables, as needed.

Database Cardinality

In data modeling, the cardinality of one data table with respect to another data table is a critical aspect of database design. Relationships between data tables define cardinality when explaining how each table links to another.

In the relational model, tables can be related as any of:

- **many-to-many**
- **many-to-one** (rev. **one-to-many**)
- **one-to-one**

This is said to be the **cardinality** of a given table in relation to another.
For example, considering a database designed to keep track of hospital records. Such a database could have many tables like:

- a ***Doctor*** table full of doctor information
- a ***Patient*** table with patient information
- and a ***Department*** table with an entry for each department of the hospital.

In that model

- There is a **many-to-many** relationship between the records in the doctor table and records in the patient table (Doctors have many patients, and a patient could have several doctors);
- a **one-to-many** relation between the department table and the doctor table (each doctor works for one department, but one department could have many doctors).
- **one-to-one** relationship is mostly used to split a table in two in order to optimize access or limit the visibility of some information. In the hospital example, such a relationship could be used to keep apart doctor's personal or administrative information.

Example**:** A chemical factory producing chemical materials, each material identified by a name and a formula.

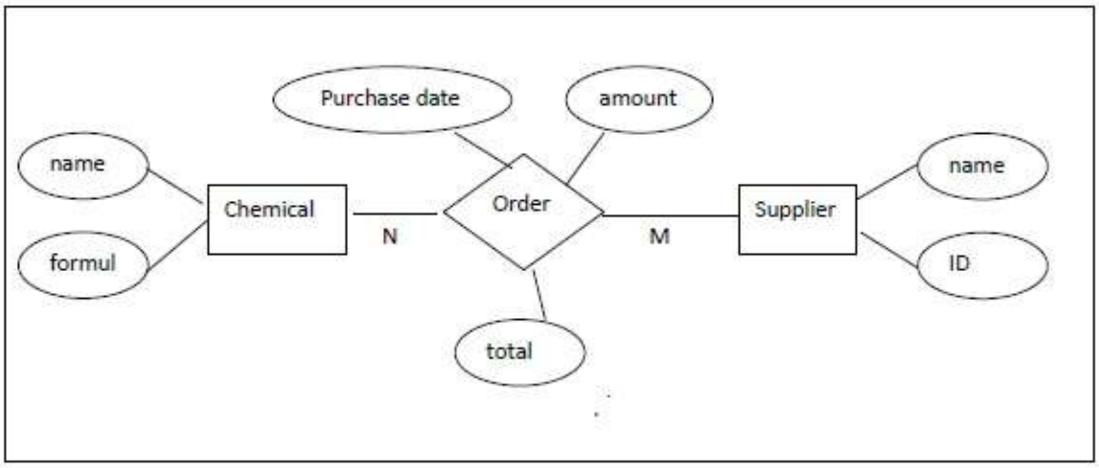
The supplier, identified by his name and his ID, purchase from the factory by an order. The order has date, amount and total.

Example 1 : A chemical factory producing chemical materials , each material identified by a name and a formula .

The supplier , identified by his name and his ID , purchase from the factory by an order . The order has date , amount and total.

To draw the ER model

Each supplier can have any material , and any material can go to any supplier . The relation is of type **many-to-many ,** as shown in figure below

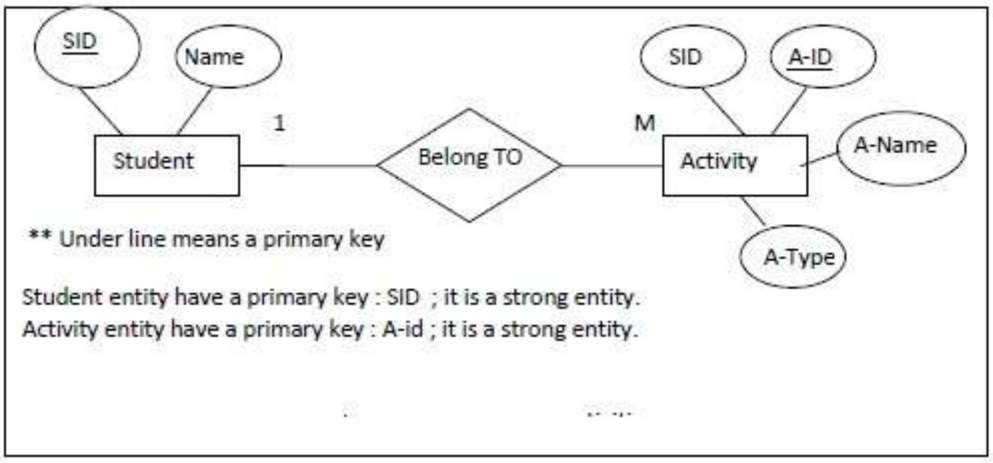


Weak Entity

An entity may not have sufficient attributes to form a primary key. Such an entity is termed a

weak entity* .** An entity that has a primary key is termed a ***strong entity.

Consider the relationship ***Have*** which links a student with his activates as shown in figure below

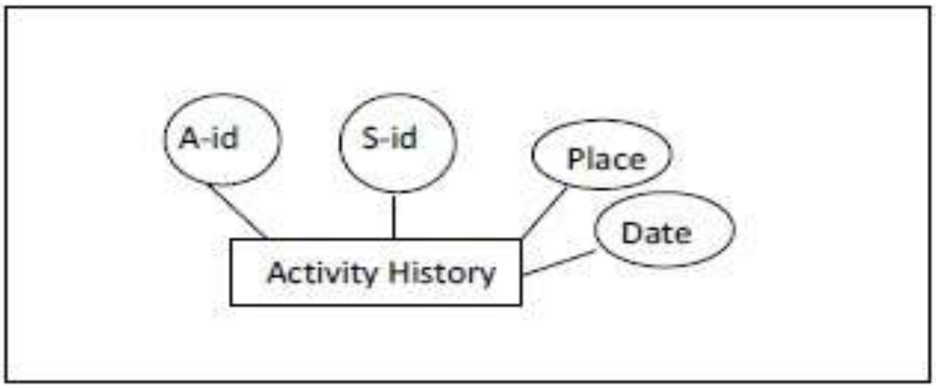


** Under line means a primary key

Student entity have a primary key : SID ; it is a strong entity.
Activity entity have a primary key : A-id ; it is a strong entity.

Each student have many activities

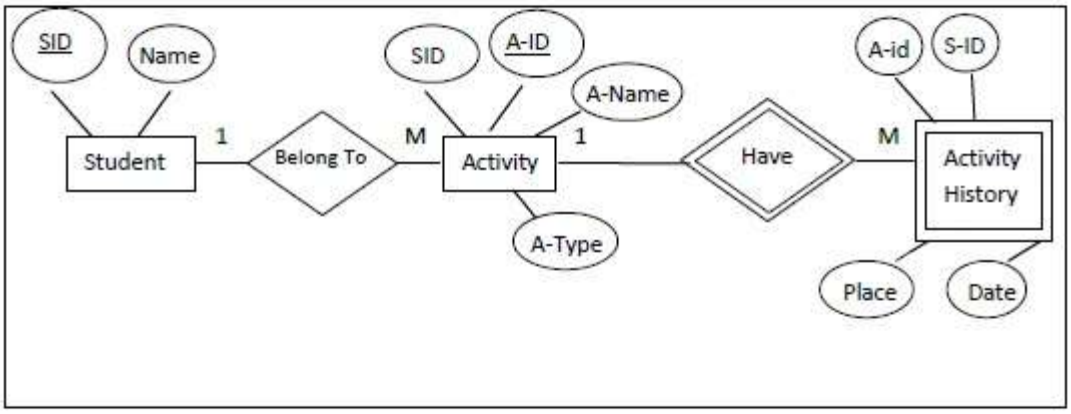
If more details is needed about all the activities the student have (history) , another entity will be added as shown if figure below



Activity History entity

This new entity have S-ID and A-ID as a foreign keys to recognize each record belong to what student and what activity. This mean for each activity a student have, he/she can carry it many times.

Figure below shows how the new entity, Activity History, is linked with activity entity



Linking the three entities

In the Activity History, there is no primary key, so it is a weak entity.
The weak entity to be meaningful, it must be part of a one-to-many relationship.
The weak entity is depends on the strong entity , that means if the strong entity is delete then the weak entity must be deleted.

In the example of figure above if the activity is deleted then all the history for that activity must be deleted.

But if the history is deleted , the activity no need to be deleted.

The DOMINANT entity is the strong entity.

The SUBORDINATE entity is the weak entity.

The weak entity in the ER diagram is indicated by a double outline box, and the corresponding relationship by a double outline diamond.

More Designs Considerations

In this section , we consider how a database designer may select from the wide range of alternatives . Among the decision to be made are the following:

• weather to use an attribute or an entity to represent an object :

consider the entity ***employee*** with attribute **employee-name** .

if we want to add telephone number to the employee:

case 1 : another attribute ,**telephone-number** , is added to the entity.

Case2 : the ***telephone*** can be consider as an entity in its own with an attribute :

Telephone-number and **location**, ant the two entities are connected with some relation.

In the first case every employee has one telephone number.

In the second case , each employee has several telephone numbers

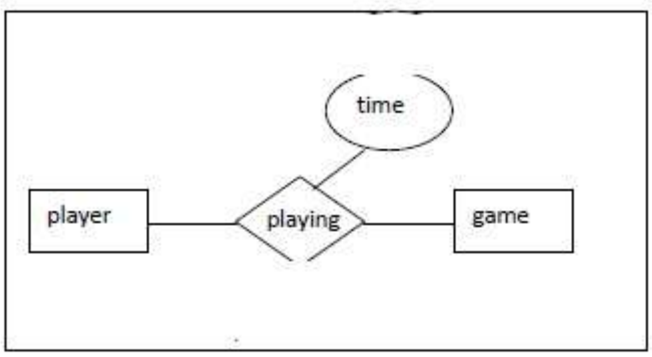
• weather a real world concept is expressed by an entity or by a relationship .

For example a ***playing*** can be modeled as an entity, or it can be a relation between

player and ***game,*** with playing-time attribute.

If every game is played by one player then playing is better be a relationship as in figure below

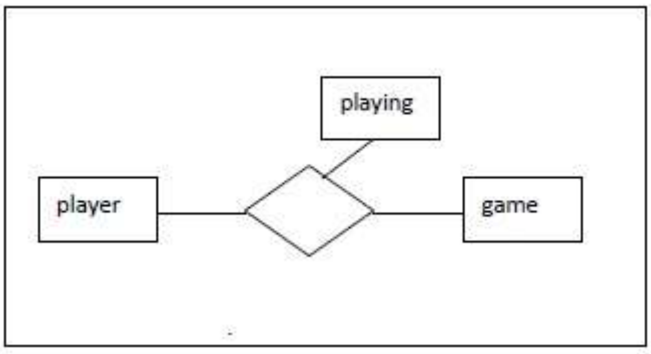
If several players playing the same game then we must replicate the playing information (time) for each player, and the problems of repetition and updating are arise.



Playing is a relationship

If the information of the playing is updated then this update must be done to each player.

Playing can be an entity rather than a relation. In this case there will be only one existence of the playing information, and the player and the games is connected with the playing by keys as shown in figure below



Playing is an entity