RDBMS - Database Design:

What is Database Design?

It can be generally defined as a collection of tasks or processes that enhance the designing, development,implementation,and maintenance of an enterprise data management system.

Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space.

Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

Why Database Design is Important?

1) It provides the blueprints of how the data is going to be stored in a system. A proper design of a database highly affects the overall performance of any application.

2) The designing principles defined for a database give a clear idea of the behavior of any application and how the requests are processed.

3) Another instance to emphasize the database design is that a proper database design meets all the requirements of users.

4) Lastly, the processing time of an application is greatly reduced and the constraints of designing a highly efficient database are properly implemented.

A good database design starts with a list of the data that you want to include in your database and what you want to be able to do with the database later on.

This can all be written in your own language, without any SQL.

In this stage you must try not to think in tables or columns, but just think : “What do I need to know?” Don’t take this too lightly, because if you find out later that you forgot something, usually you need to start all over. Adding things to your database is mostly a lot of work.

1) Identifying the Entities:

The types of information that are saved in the database are called **“Entities”.**

These entities exist in four kinds: **people, things, events, and locations.**

Everything you could want to put in a database fits into one of these categories.

If the information you want to include doesn't fit into these categories then it is probably not an entity but a property of an entity , an **attribute.**

For Example:

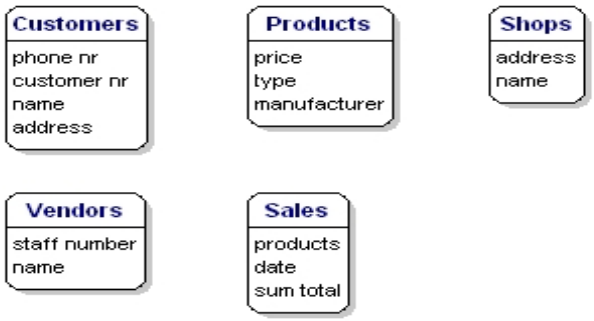
Imagine that you are creating a website for a shop, what kind of information do you have to deal with?

In a shop you sell your products to customers. The **"Shop"** is a location; **"Sale"** is an event; **"Products"** are things; and **"Customers"** are people. These are all entities that need to be included in your database.

But what other things are happening when selling a product? A customer comes into the shop , approaches the vendor, asks a question and gets an answer. **"Vendors"** also participate, and because vendors are people , we need a vendor's entity.

2) Identifying Attributes:

The data elements that you want to save for each entity are called attributes.

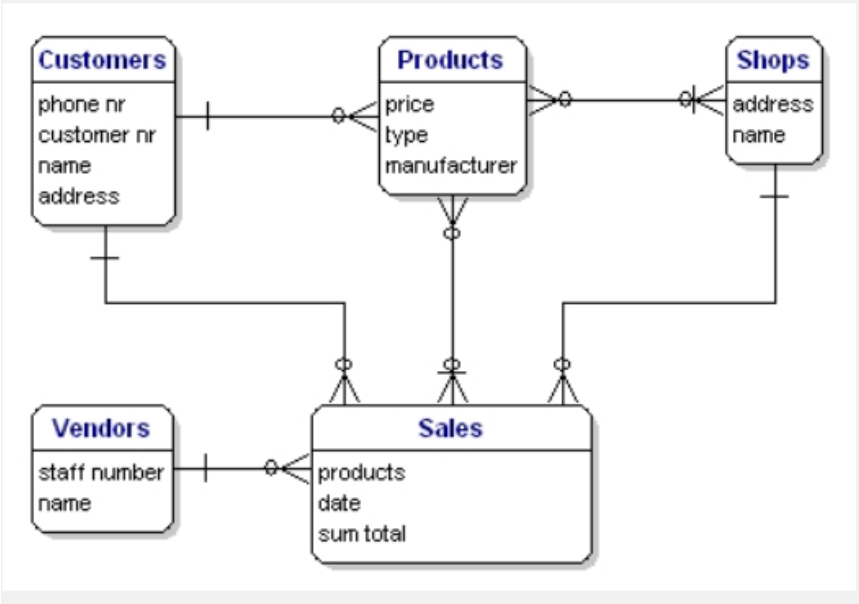


3) Identifying the Relationships:

The next step is to determine the relationship between the entities and to determine the cardinality of each relationship.

The relationship is the connection between the entities, just like in the real world: what does one entity do with the other, how do they relate to each other?

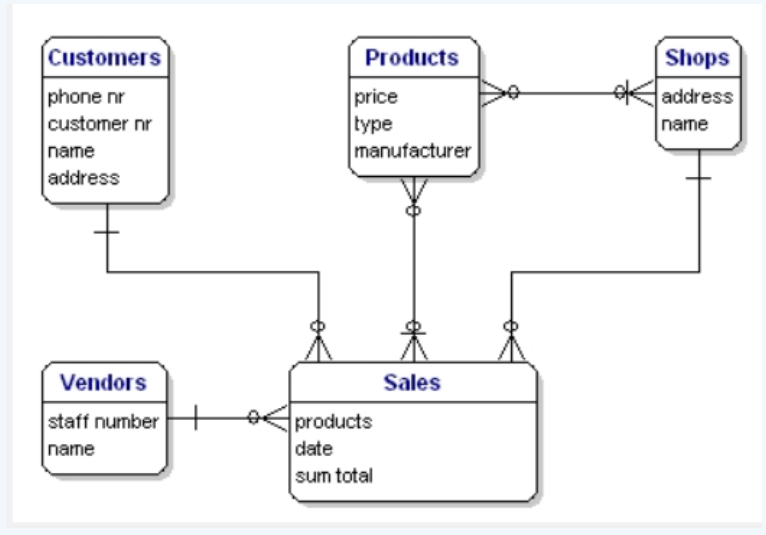
For example, customers buy products, products are sold to customers, a sale comprises products, a sale happens in a shop.



4) Removing the Redundant Relationships

Sometimes in your model you will get a ‘redundant relationship’. These are relationships that are already indicated by other relationships, although not directly.

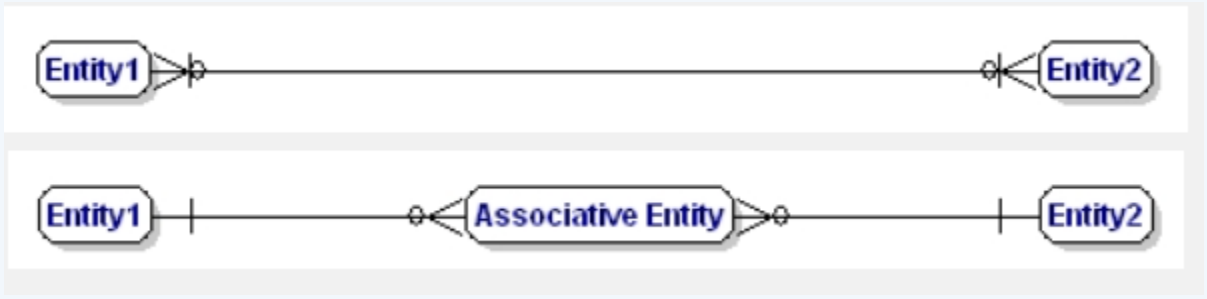
In the case of our example there is a direct relationship between customers and products. But there are also relationships from customers to sales and from sales to products, so indirectly there already is a relationship between customers and products through sales. The relationship 'Customers <----> Products' is made twice, and one of them is therefore redundant. In this case, products are only purchased through a sale, so the relationships 'Customers <----> Products' can be deleted.



5) Solving Many - to - Many Relationships

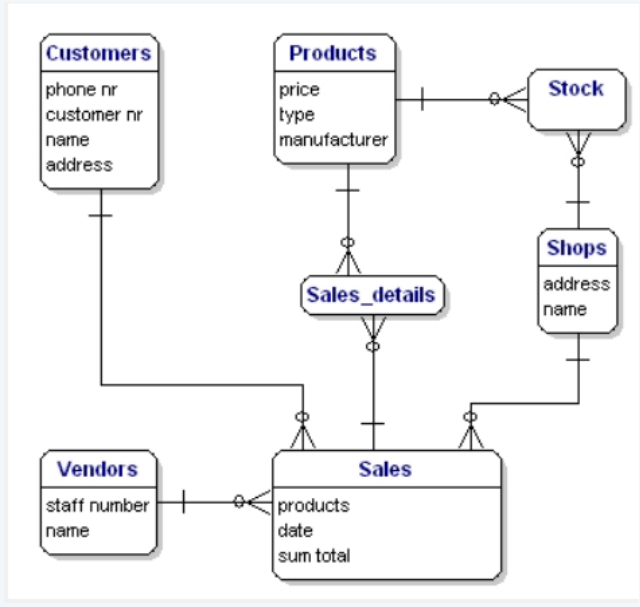
Many-to-many relationships (M:N) are not directly possible in a database. What a M:N relationship says is that a number of records from one table belongs to a number of records from another table. Somewhere you need to save which records these are and the solution is to split the relationship up in two one-to-many relationships.

This can be done by creating a new entity that is in between the related entities. In our example, there is a many-to-many relationship between sales and products. This can be solved by creating a new entity: sales-products. This entity has a many-to-one relationship with Sales, and a many-to-one relationship with Products. In logical models this is called an associative entity and in physical database terms this is called a link table, intersection table or junction table.



In the example there are two many-to-many relationships that need to be solved: 'Products <----> Sales', and 'Products <----> Shops'. For both situations there needs to be created a new entity, but what is that entity?

For the Products <----> Sales relationship, every sale includes more products. The relationship shows the content of the sale. In other words, it gives details about the sale. So the entity is called 'Sales details'. You could also name it 'sold products'.
The Products <----> Shops relationship shows which products are available in which the shops, also known as 'stock'.



7) Defining the Attributes Data Types