RDBMS

Analyse the requirement ? - w3h - only what and how
logical Model - Err diagram
Physical Model -
Testing
production

Entity
Attribute
Normalization form
Single value

Indirect relationship

It is a program used to create,update,and manage relational databases

Prostgre Sql

Types of management system
Fms   -file management
Dbms   -database
Rdbms -relational database

Dbms   - no relation
Rdbms - relation - using primary key

What is table?
As per RDBMS table is an object
Table is collection of related data
Table is the most common and simplest form of data storage
Primary key - is not null -blank

Fields?

Field - column
Every table is broken up into smaller entities called fields
Field is a column in a table that is designed to maintain specific information about every record in the table.

Ex: our CUSTOMERS table consists of different fields like ID , name , salary

What is Record or row?

A record is also called as a row of data is each individual entry that exists in a table,
For ex there are 3 records in the above CUSTOMERS table
Single row of data or record in the CUSTOMERS table

Id
1

What is a column ?

A column is a vertical entity in a table that contains all information associated with a specific field in a table .
For ex, out Customers table have a different columns to represent ID, Name , Age , Salary

What is Null Value?

A Null value in a table is a value in a field that appears to be blank, which means a field with a NULL value is a field with no value .

It is very important to understand that a NULL value is different from a zero value or a field that contains space .

The one that has been left blank during a record creation

What are Constraints?

Constraints - condition

Constraints are the rules enforced on data columns on a table . These are used to limit the type of data that can go into a table . these ensures accuracy and reliability of the data in the database

Constraints can either be column level or table level , column level constraints are applied only one column whereas, table level constraints are applied to the entire table

Type of constraints

NOT NULL
Ensures that a column cannot have a NULL value

DEFAULT
Provides a default value for a column when none is specified.

UNIQUE key
Ensures that all the values in a column are different

PRIMARYkey
Uniquely identifies the each row/record in any another database table

FOREIGNkey

CHECK

INDEX

What is Normalization?

It is the process of efficiency organizing data in a database

There are two reasons of this normalization process

1) Eliminating redundant data , for example , sorting the same data in more than one table.
2) Ensuring data

Compulsory normal forms

First Normal First

1NF sets the basic rules to organize the data in a database.
A database is said to be in first normal form if it satisfies the following conditions

Rule1(Atomic values)
Every column of a table should contain only atomic values is a value that cannot be divided further
Rule2(No Repeating Groups)
There are no repeating groups of data

Second Normal Form
It states that it should meet all the rules for 1NF

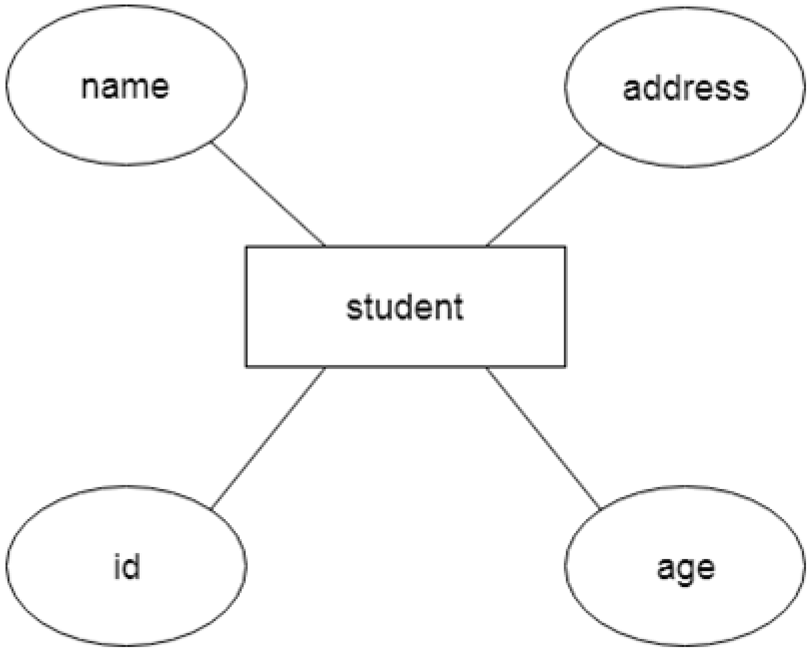
Third Normal form
Int d.

It is used to reduce data duplication. It is also used to achieve the data

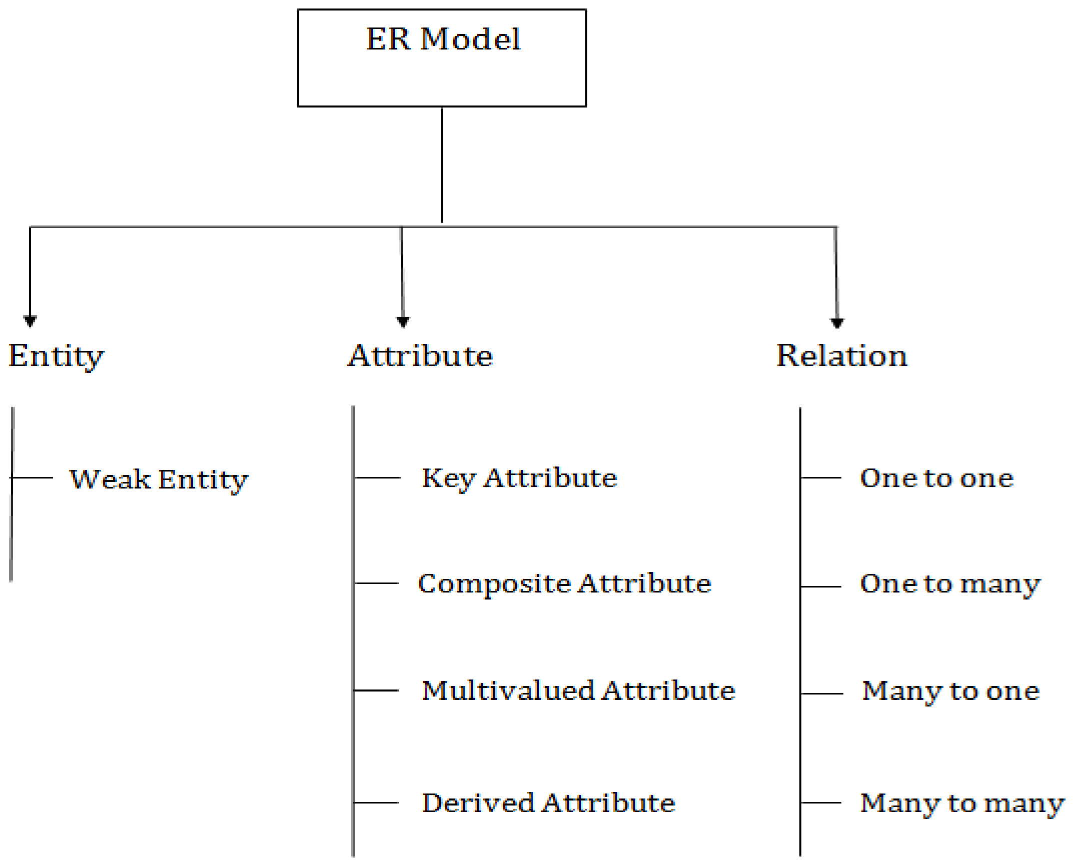
ER (Entity Relationship) Diagram in DBMS

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



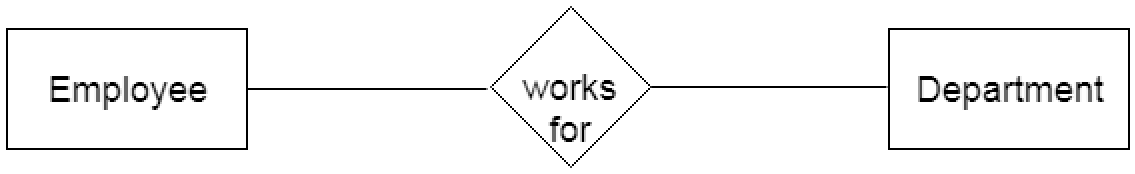
Component of ER Diagram



1. Entity:

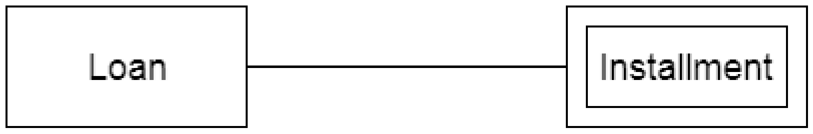
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

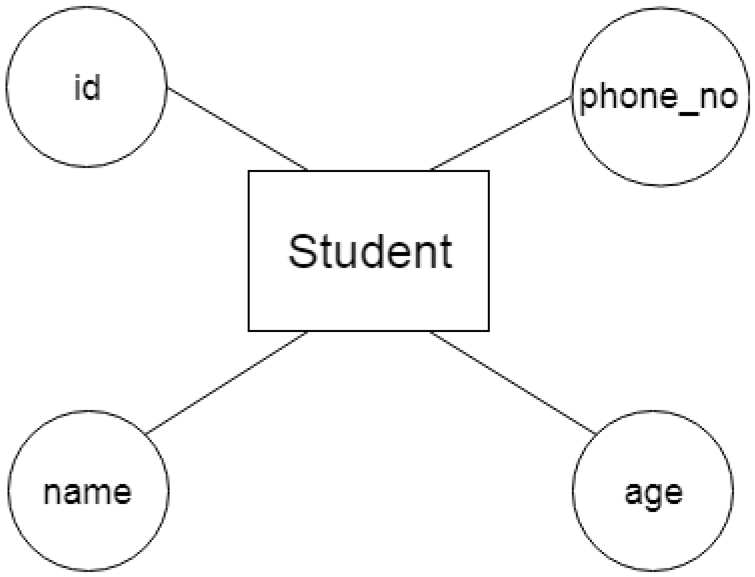
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

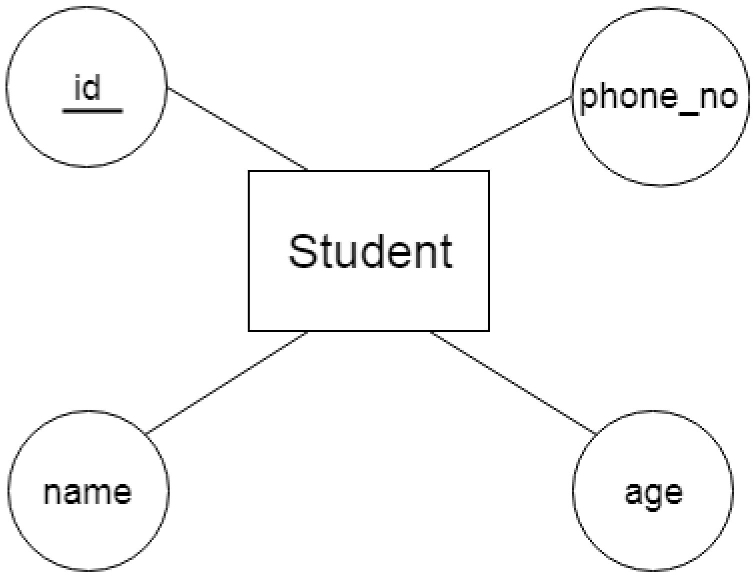
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



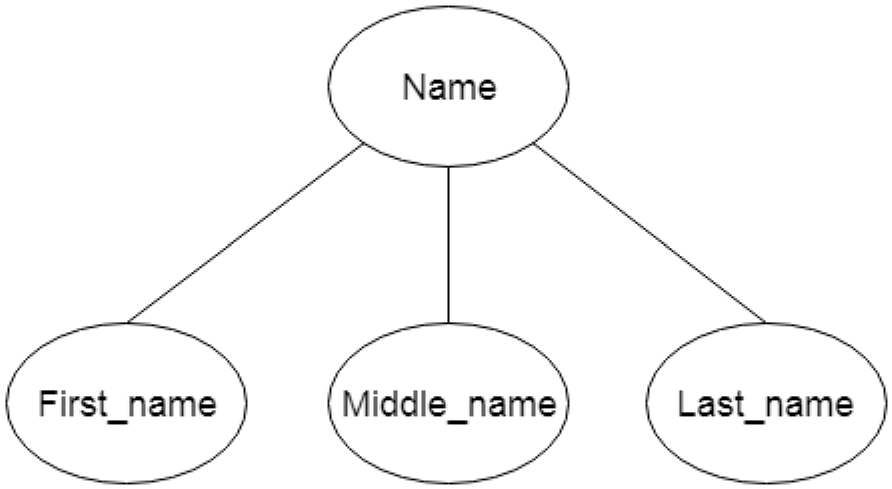
a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

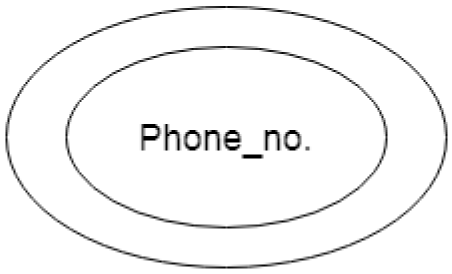
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent a multivalued attribute.

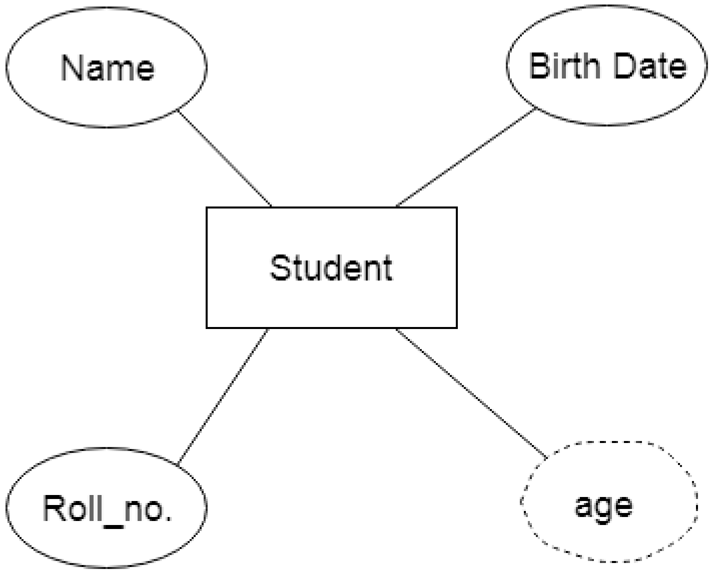
For example, a student can have more than one phone number.



d. Derived Attribute

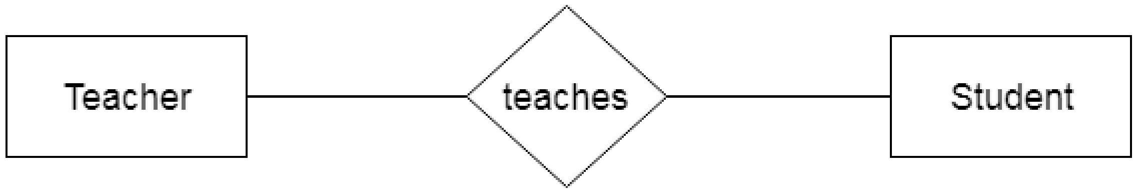
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

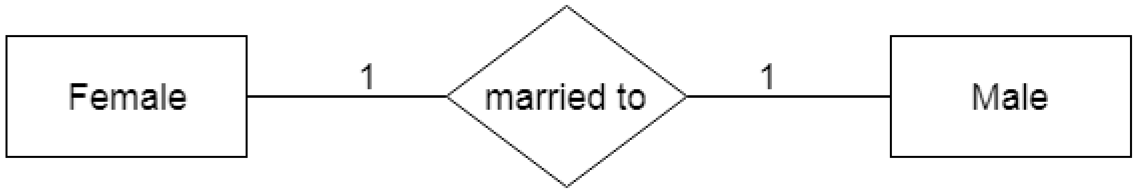


Types of relationship are as follows:

a. One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

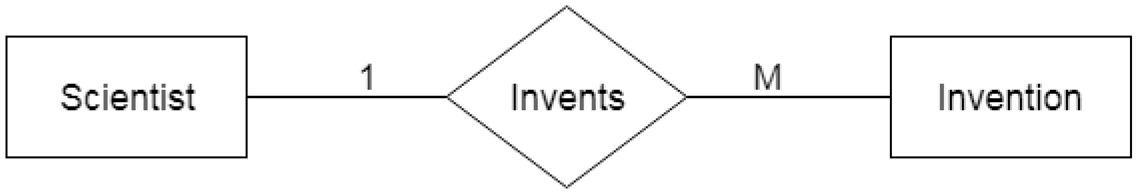
For example, A female can marry to one male, and a male can marry to one female.



b. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

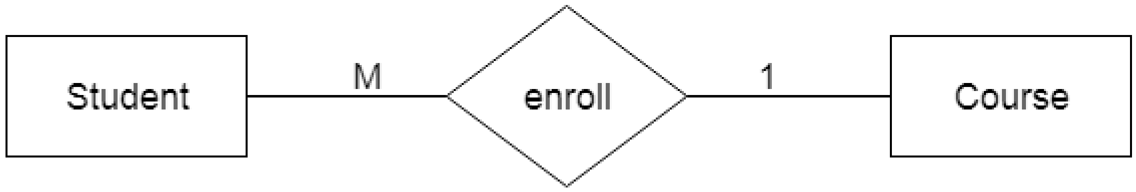
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

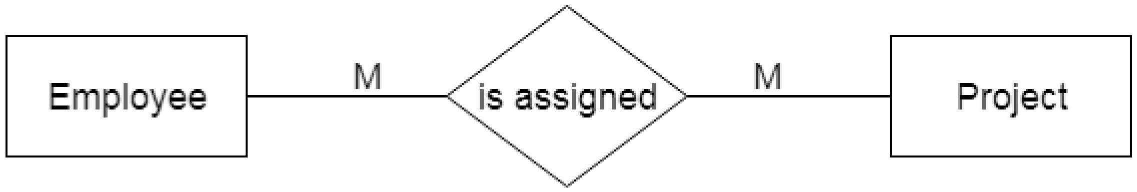
For example, Student enrolls for only one course, but a course can have many students.



d. Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.



What is database Design ?

Database design can be generally defined as a collection of tasks or processes that enhance the designing, development,implementation,and maintenance of enterprise data management system.
Designing a proper database reduces the maintenance=ance 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?

Database designs provide 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.

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

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

lastly , the processing time of an application is greatly reduced if 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 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 - table**

   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 those 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.**

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

Ina 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 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 vendors entity

2. **Identifying attributes**
   The data elements that you want to save for each entity are called ’attributes’.

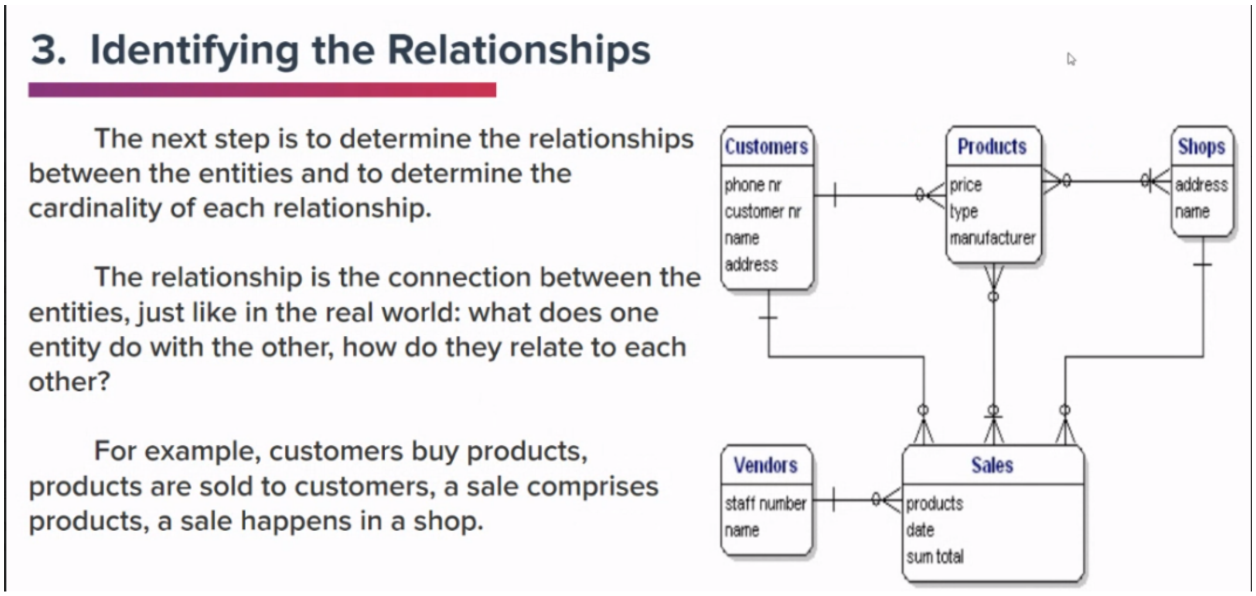
Customers	**products**	**shops**
Phone no	price	address
Customer nr	type	name
Name	manufacturer	
address		

Vendors	**sales**
Staff number	products
Name	date
	Sum total

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.



3. Identifying the Relationships

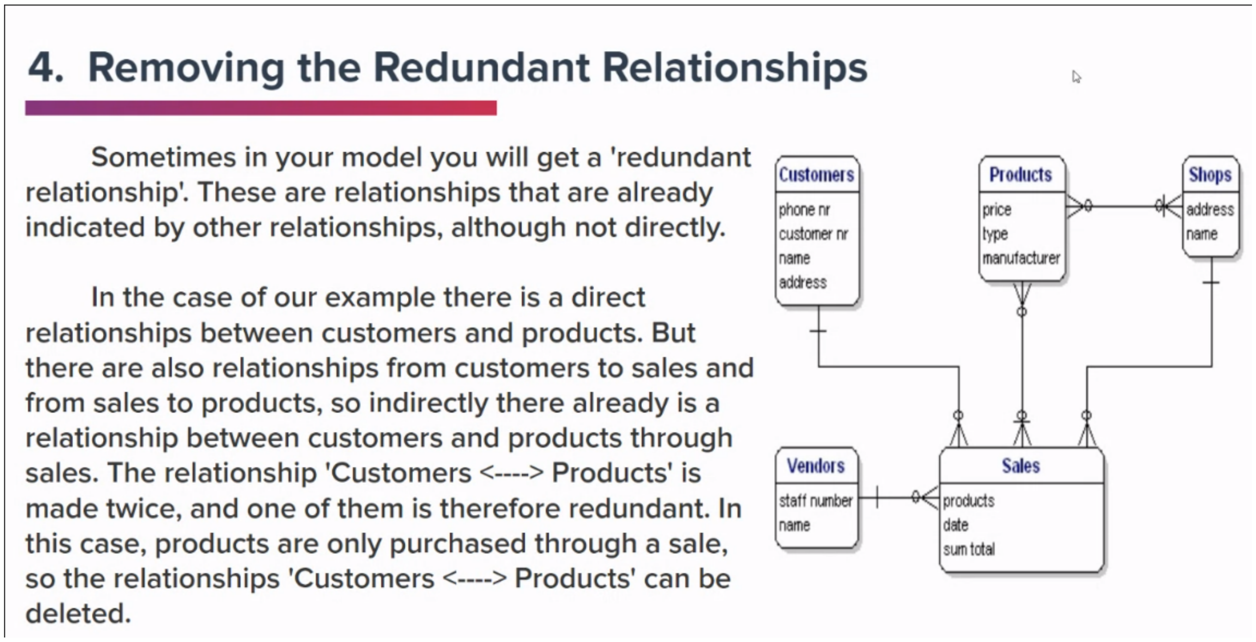
The next step is to determine the relationships between the entities and to determine the cardinality of each relationship.

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 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.



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 relationships 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 fro 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

6.

   In the example there are two many-to-many relationships that need to be solved:'Products<- - - - >Sales','Products <- - - - > Shops'. For both situations these 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 attributes data types**