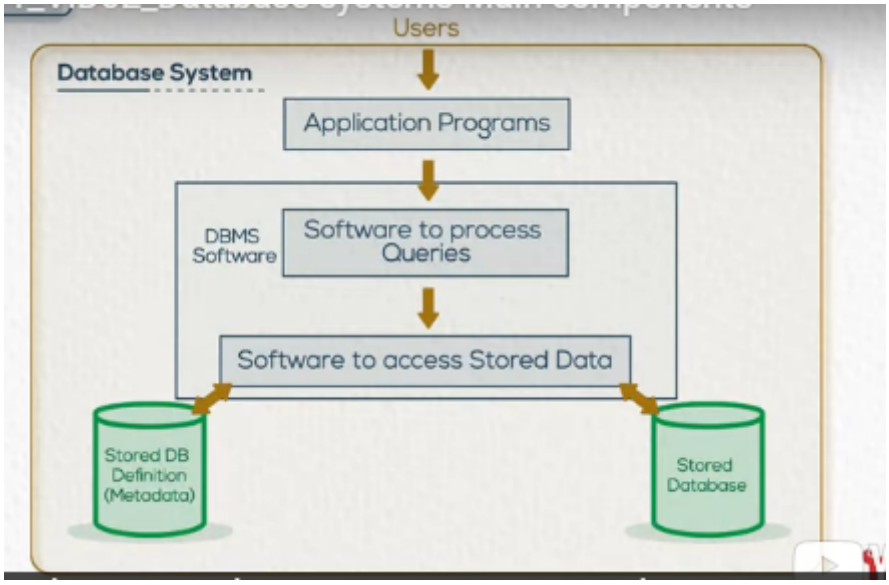
Intro To DataBase (Mahara Tech)

The Main Components Of Database System:

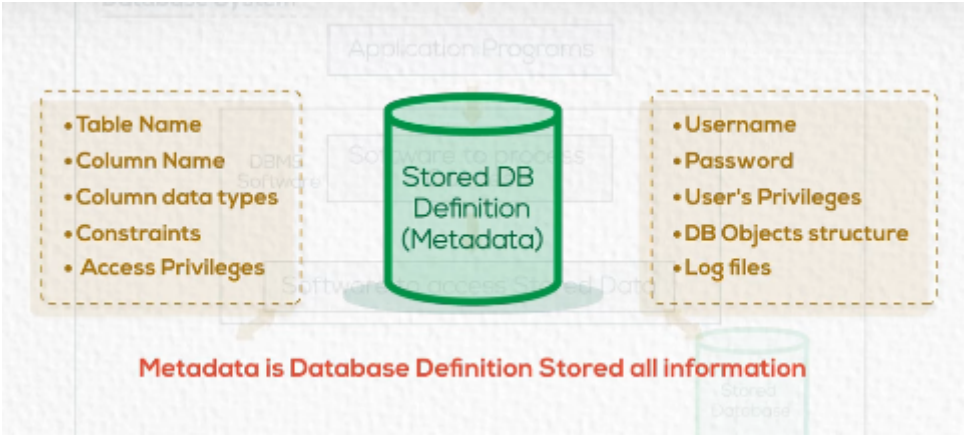


1-The DBMS is divided into two parts

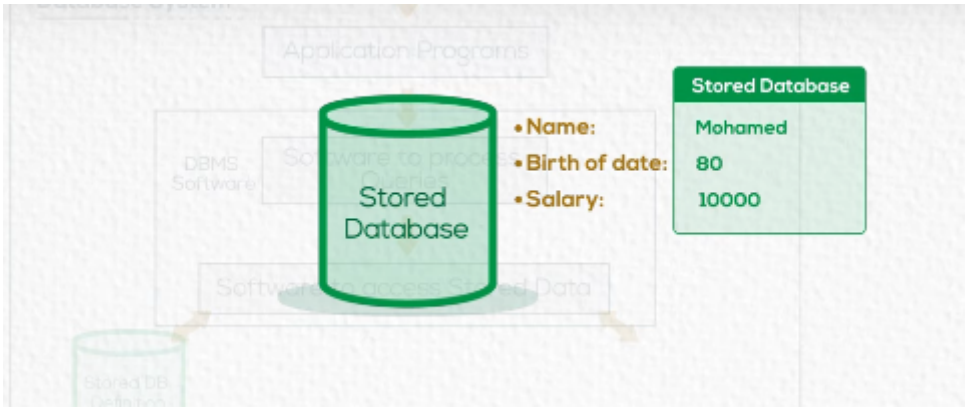
- the first part processes the queries that came from the application program
- the second part accesses the database itself and return, add, edit the data

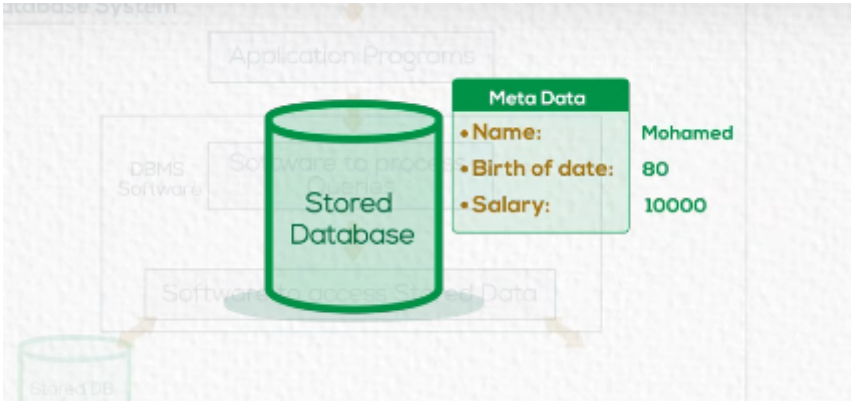
2-The Database is divided into two parts

- First **Database definition (metadata)** a set of information about the data for example the data types, column names, table name, constraints, access privileges



- Second **Stored Database** is the data itself for example Mohamed, 80, $10000

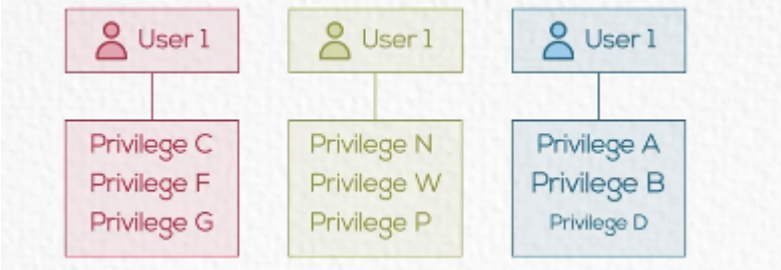




The Advantages Of Database System

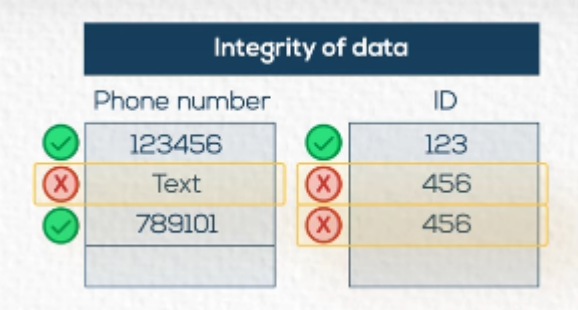
1- **Controlling Redundancy** (The data is found in one place and many users can access this data)

2- **Restricting Unauthorized Access** (By creating the users and give them the permissions and privileges on this database)



3- **Sharing data**

4- **Enforcing Integrity Constraints**(the data apply specific business rules and data logical rules)



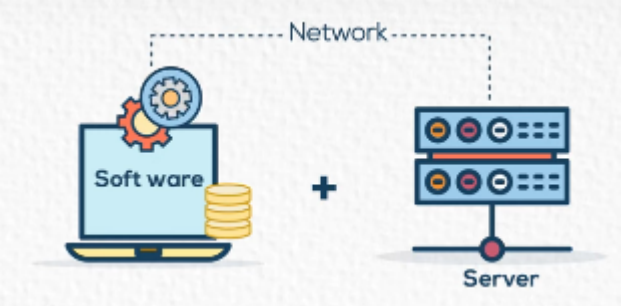
5- **Inconsistency can be avoided**(when data is updated we can all share the same updates at the same time)

6- **Providing Backup and Recovery**(the DBMS can recover the data and avoid losing it)

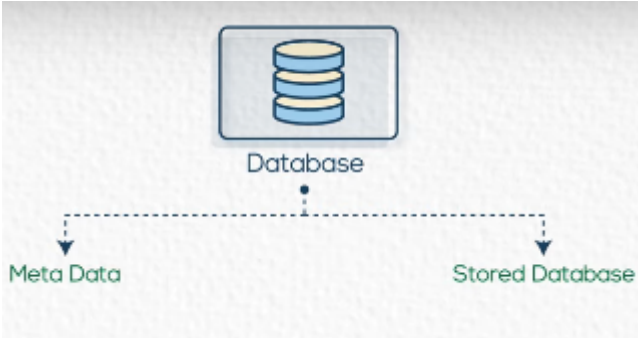
Database Systems Disadvantages

1- **Needs expertise to use**

2- **DBMS is expensive** (the cost of infrastructure and networks that I may need to add to store the database itself and can support the desired number of user)



3- **May be incompatible with any other available DBMS** (by using metadata)



there are some DBMS are not compatible with each other so we can't transfer the data easily but we solve this problem by using **third party tool** to facilitate the transfer between the DBMS

Database Users

The cycle of creating the database or having it works:

EXAMPLE:- A company needs to have a database system, let's know the steps of getting it done:

1-**Analysis and requirements gathering**

The system Analyst gathers the requirements from the stakeholders and understands why they want the DBMS and who will use it and have the access on it, the number of users, and the budget the stakeholders could afford.

2-**Database Design**

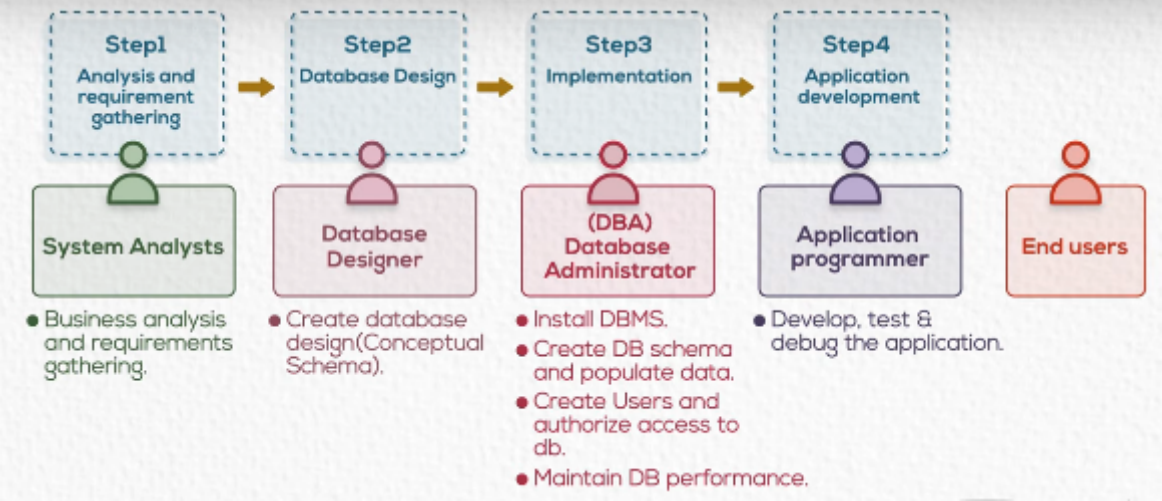
The database designer takes the requirements from the system analyst and then designs how the data will be and how the tables look like (conceptual schema)

3-**Implementation**

The DBA (database Administrator) install DBMS, Create DB schema and populate data, Create users and authorize access to DB, Maintain DB performance

4- **Application Development**

The application programmer develops, tests and debugs the application



DBMS Architecture, Data Models

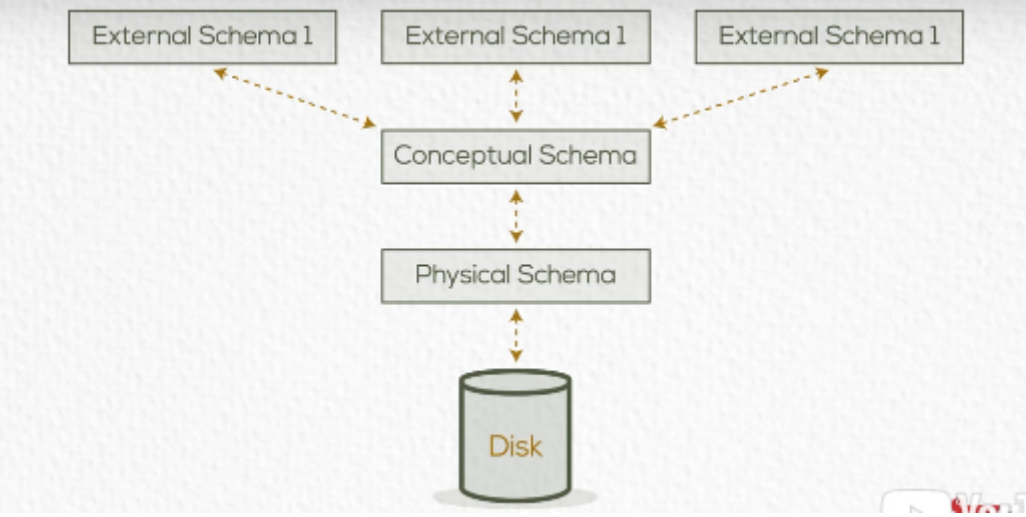
DBMS Architecture (Three Schema Architecture)

1- **External Schema** They are concerned with what data the user will see and how the data will be presented to the user (**What the user sees**)

There are many external schemas depends on the specific departments or users that have the access to use this DB

2- **Conceptual Schema** They are concerned with what is represented (**define database structures such as tables and constraints**)

3- **Physical Schema** How the data are represented in the database? how the data structures are implemented (where the data is allocated on the Hard disk, the spaces)



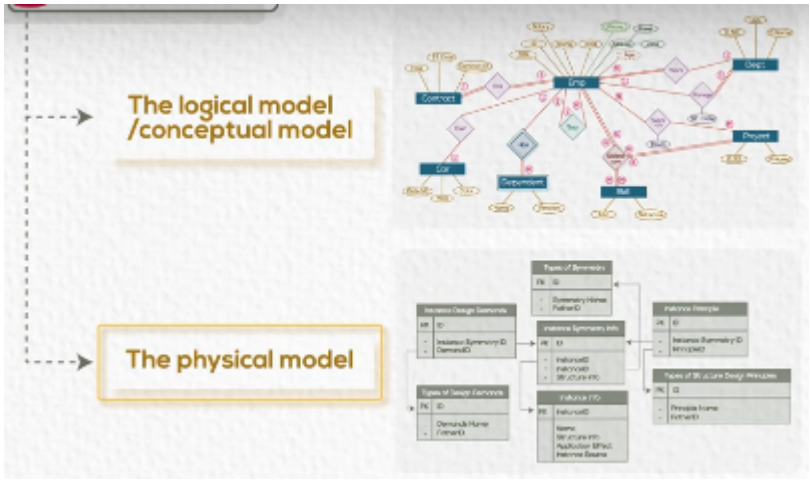
The Schemas are separated from each others because of the **data independencies** (The capacity when changing in a specific schema, it's not necessary that this change reflected on the schema in the higher level)

مثال: لو قمنا بوضع فايل جديد او تغير مكان حفظ الداتا في الفيزيكال سكيما مش بالضروروة ان ده يسمع في الاكسترنال سكيما ، و كمان مع الكونسيبشوال سكيما لو عملنا جدول جديد مش ضروري كل الاكسترنال سكيما يسمع فيها غير الاكسترنال سكيما اللي مسموحلها بالدخول علي الجدول ده طبعا

Data Models

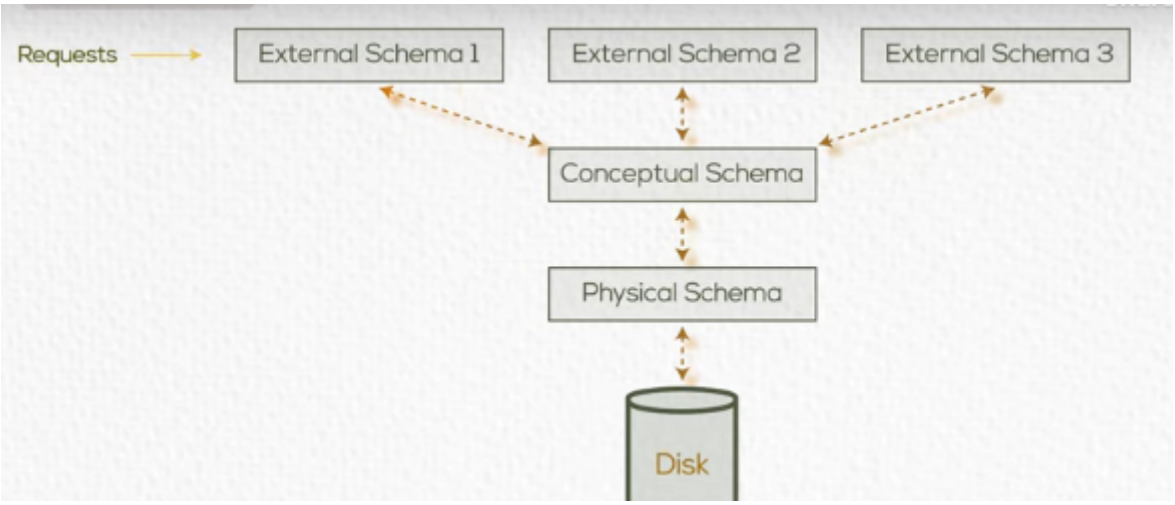
1- **The Conceptual Data Model** Provide the concepts that are close to the way many users perceive data, entities, attributes and relationships (**EX: ERD**)

2- **The Physical Model** Describes how data is stored in the computer and the access path needed to access and search for data



Mapping

The processes of transforming requests and results between levels.



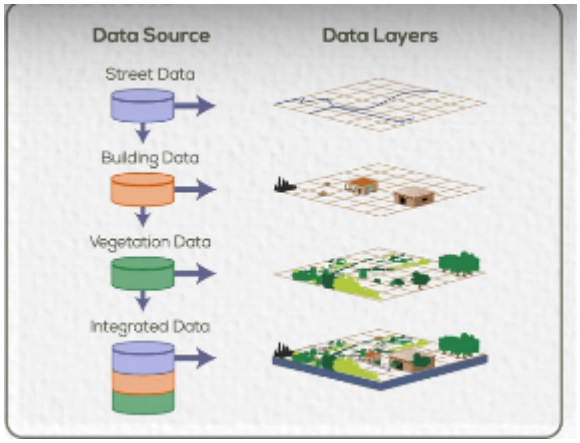
DBMS Other Functions

Previously the DBMS had functions to support/deal with Text/Number data only.

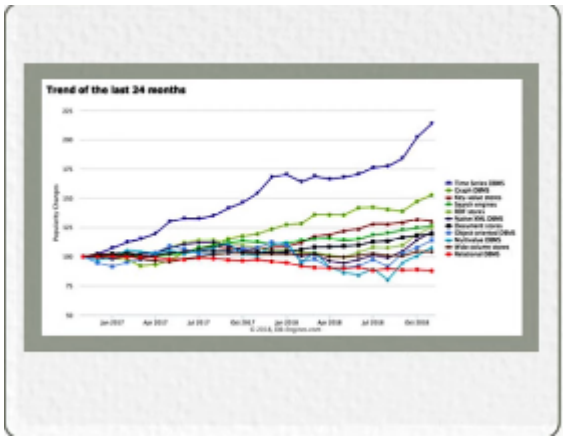
but then they've got new functions to support/deal with different types of data such as:-

1- Text/Images/Numbers/Audio/Video

2- Spatial Data (GIS and maps data)



3- Time Series



4- Data Mining

Centralized Database Environment

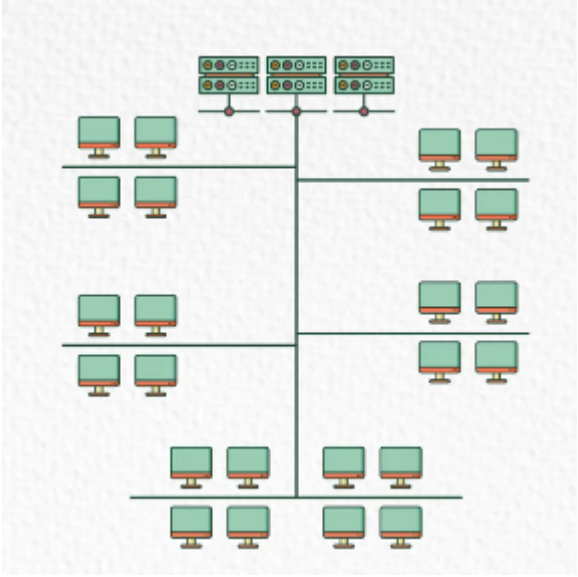
Centralized Data base

1- Mainframe Environment

There is a mainframe(contains database server and application server) and a group of connected monitors, all processing made on one machine and everyone connects to the mainframe using dummy terminal

Problems with this environment

- The processing depends on one server
- The performance is very slow
- Database and application layer has **Single Point Of Failure** (If the mainframe dropped then all the other computers would drop)



2- Client/Server Environment(Two tier environment)

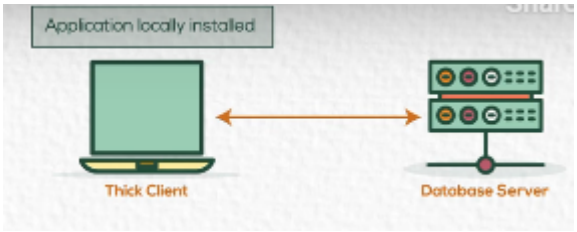
Each machine has the application locally installed on it

Problems with this environment

- Database is a single point of failure
- High cost for support

Advantages

- Application layer isn't a single point of failure



3- Internet Computing Environment(Three tier architecture)

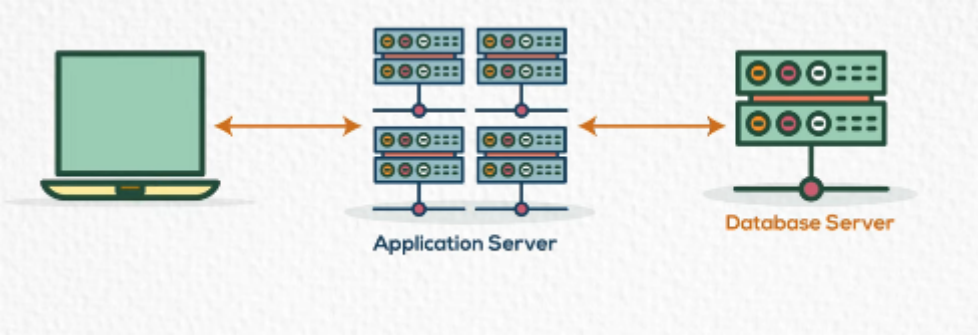
we have multiple tiers such as the database server and the application server and the Thin client

Problems with this environment

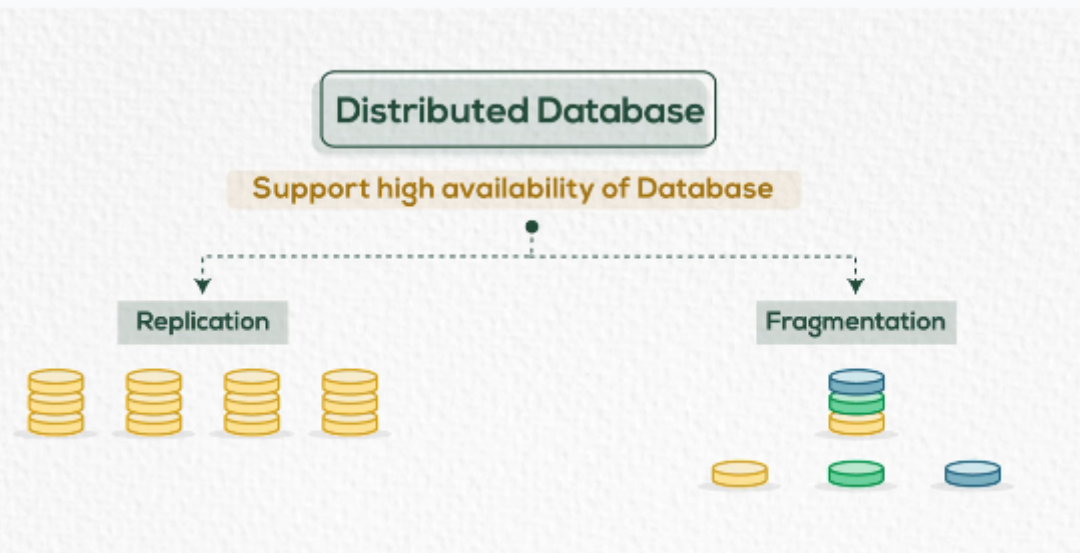
- Application server is a single point of failure
- Database is a single point of failure

Advantages

- Lower cost for support and maintenance



Distributed Database Environment



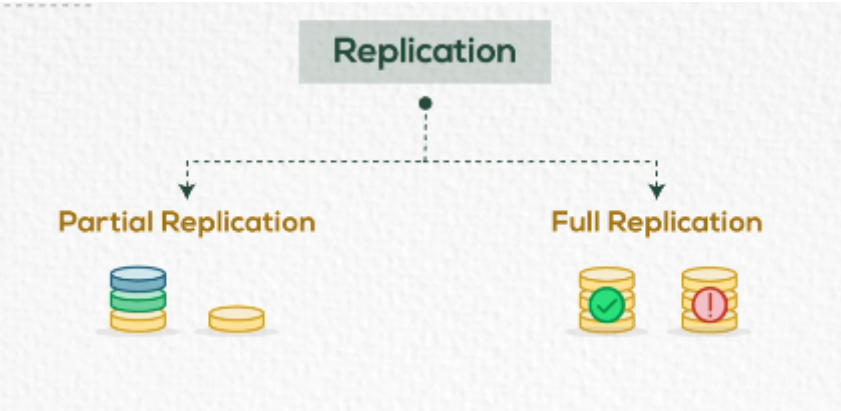
There are two method to create the database environment (**Support high availability of database if the database is critical**)

1- Replication (copy&paste for the database)

- **1.1) Partial Replication**

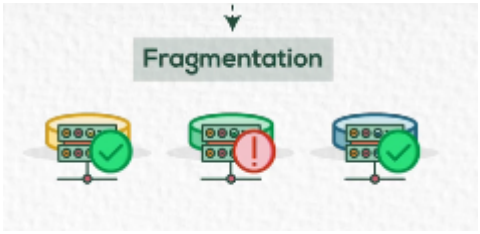
  make a copy of a specific part of this data to another server
- **1.2) Full Replication**

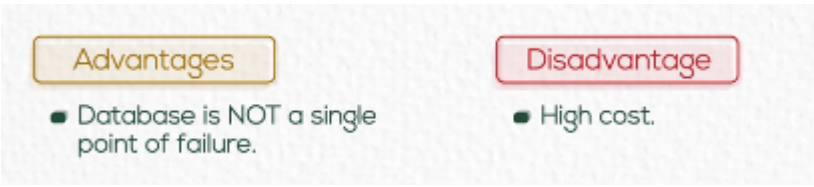
  make a copy of all the part of this data to another server



2- Fragmentation (cut&paste)

Distribute the database into fragments and put each one of them in a database server and make setup and restart and run (**It's important that all this fragment is connected through a specific network**)

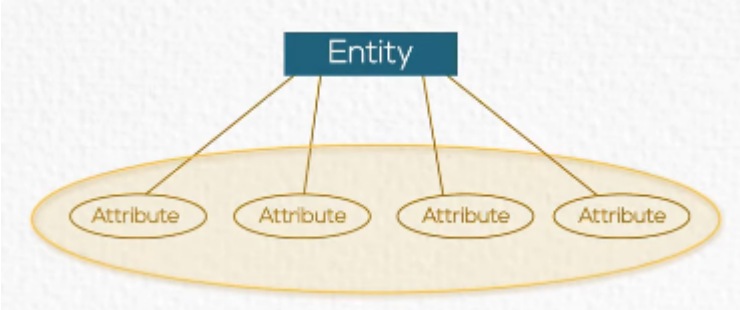


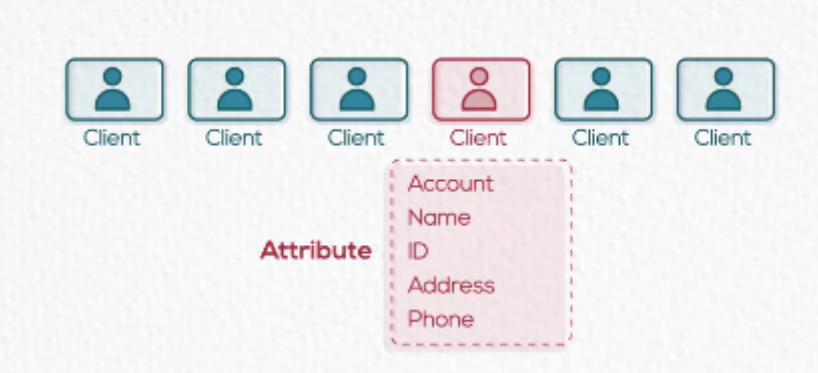


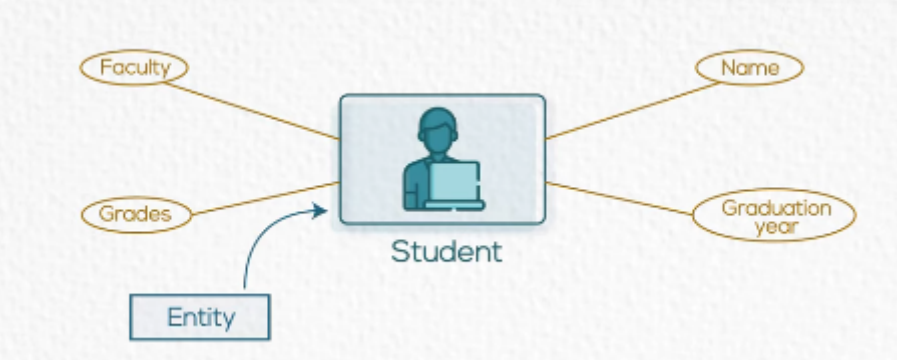
Chapter 02: Entity Relationship Modeling

Identifies information required by the business by displaying the relevant **Entities** and **Relationships** between them.

Entity is a thing in the real world with an independent existence (Physical OR conceptual existence) which I can describe as a set of characteristics or attributes.

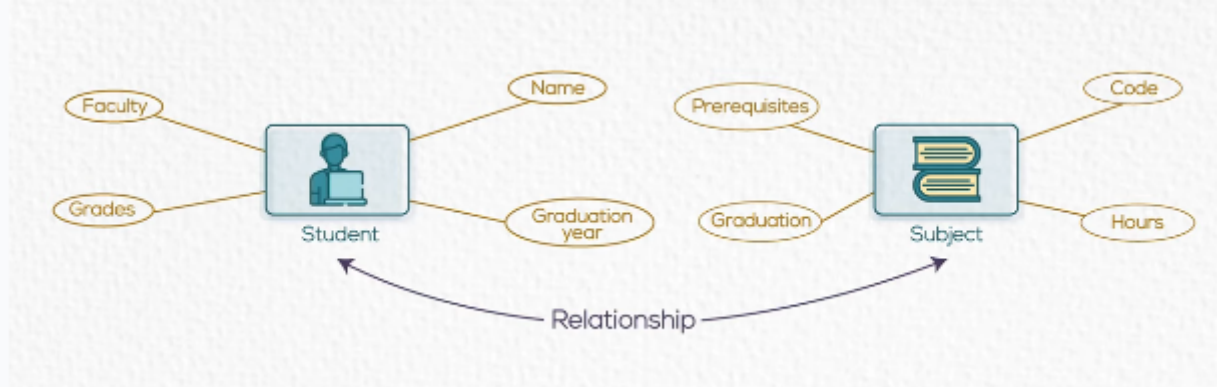






Examples for a relationship between the entities

A student studies subjects in a school
The student and the school are entities and there's a relationship between them.



In building a data model a number of questions must be addressed:

1- What entities need to be described in the model?

2- What characteristics or attributes of those entities need to be recorded?

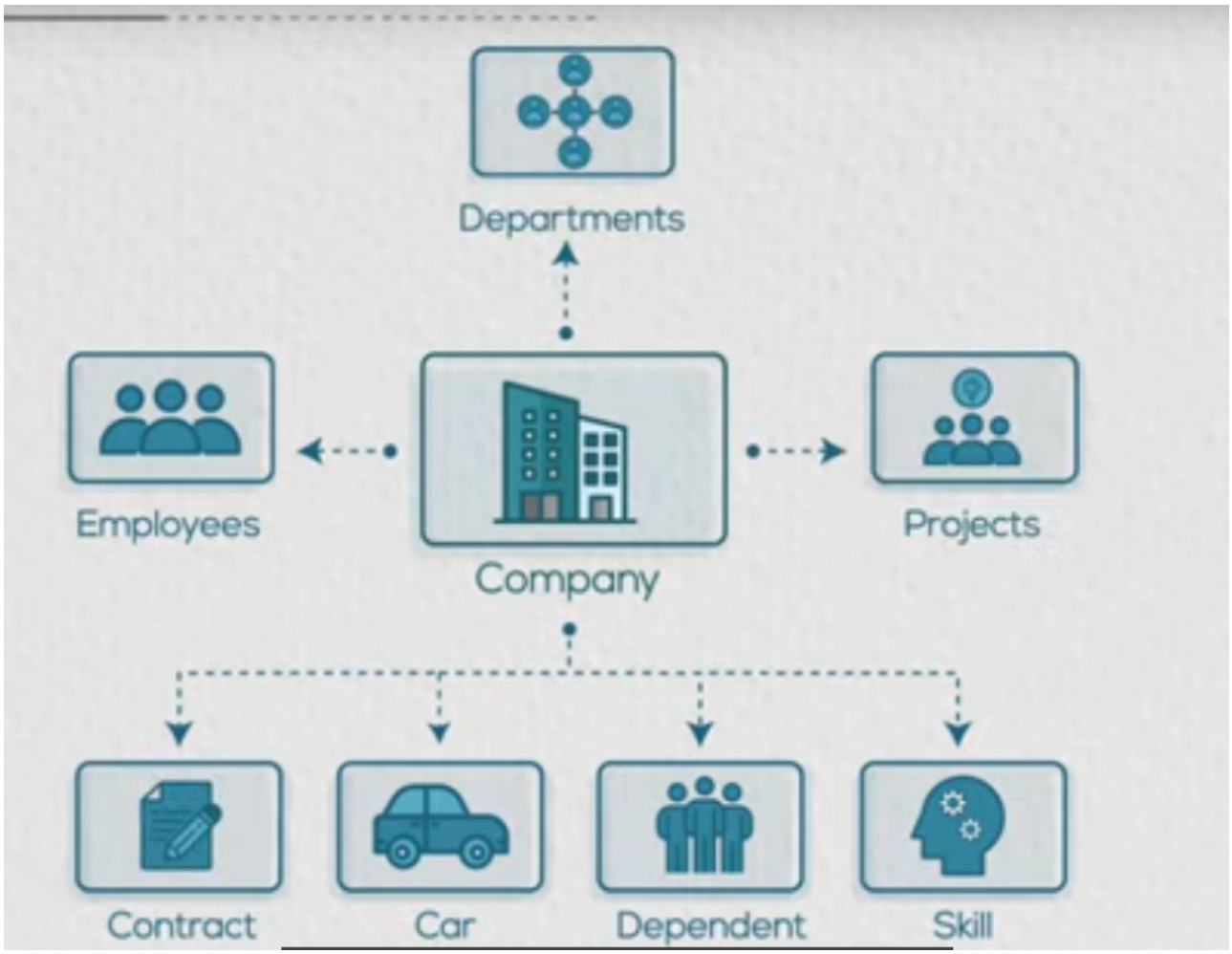
3- Can an attribute or a set of attributes be identified that will uniquely identifying one specific occurrence of an entity ?

هل فيه بيانات معينة ممكن اميز بيها ال entityعن بعضها ؟

4- What associations or relationships exist between entities?

Entities & Attributes

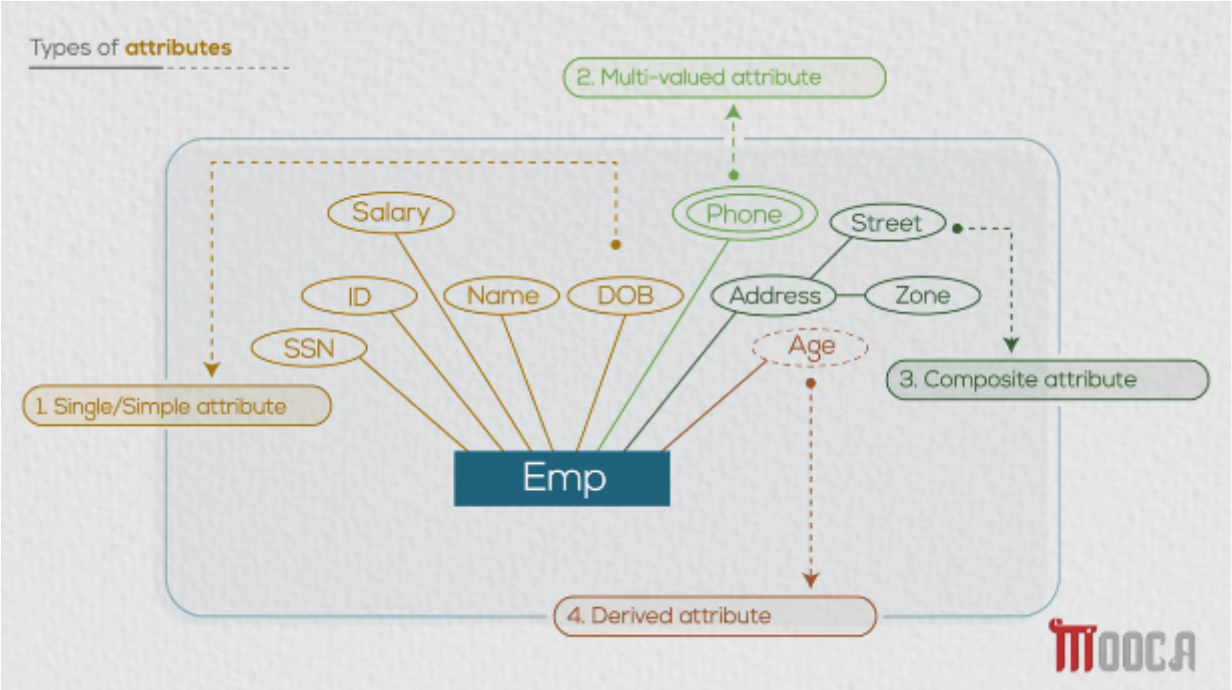
Example of a model:



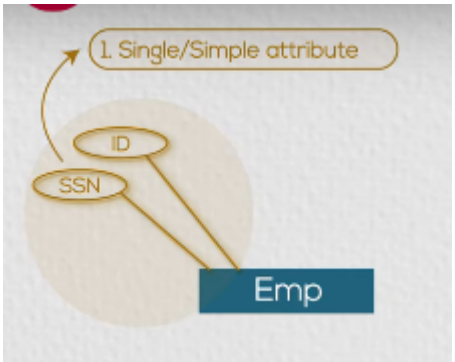
Entity described by a rectangular shape

Attribute A particular properties that describe the entity (oval shape)

Types of attributes:

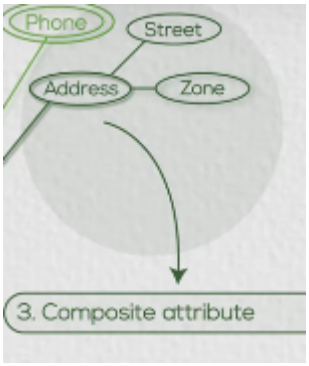


1- **Single/Simple attribute** (Not divisible and have a single value for a particular entity instance)

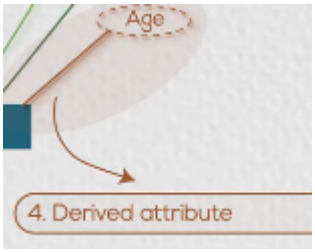


2- **Multivariable attribute** (Attributes that have multiple values for a particular entity instance)

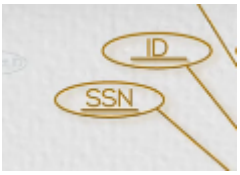
3- **Composite attribute** (Attribute that can be divided into smaller subparts)



4- **Derived attribute** (calculated from another attribute or entity)



Candidate Key When an entity type has more than one unique key. those are candidate keys.

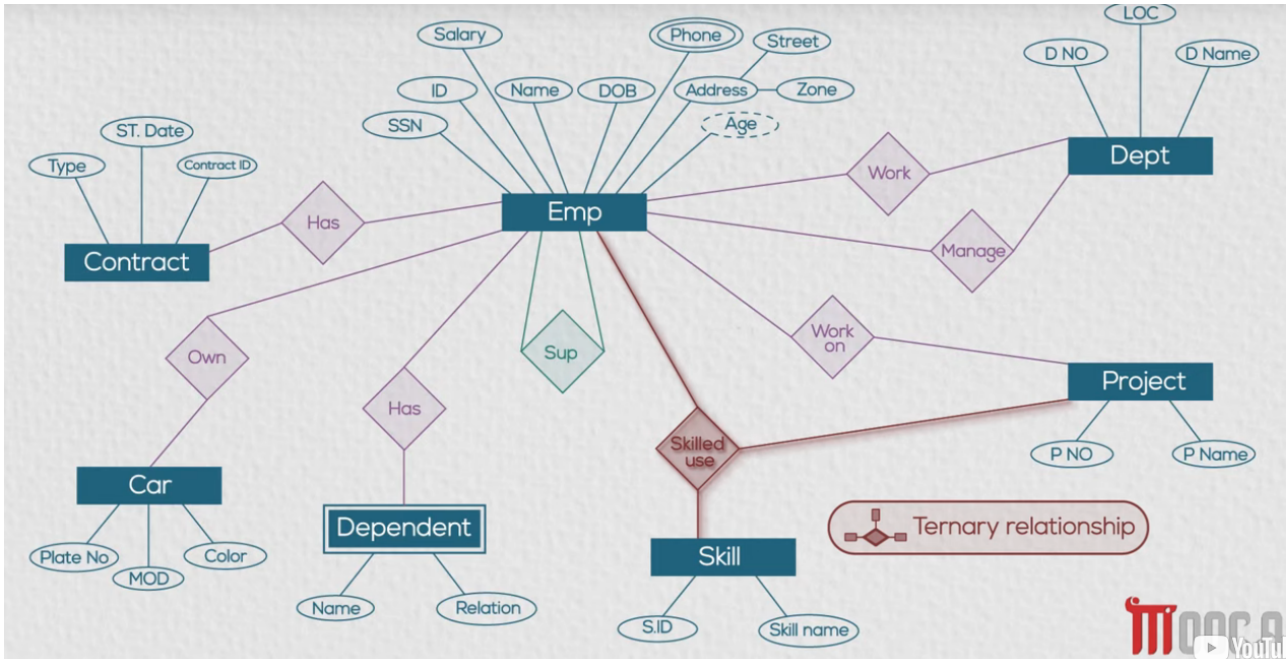


Types of Entities

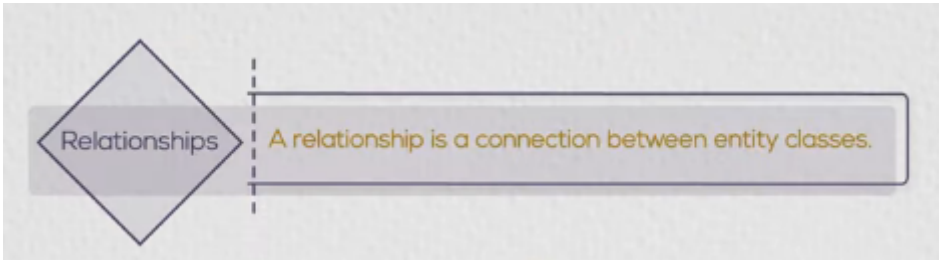
A) **Strong Entity** an entity that has a unique value

B) **Weak Entity** an entity that doesn't have a key attribute, must be fully dependent on another entity

Relationship-degree



A relationship is a connection between entity classes



For each relationship, I need to define three items

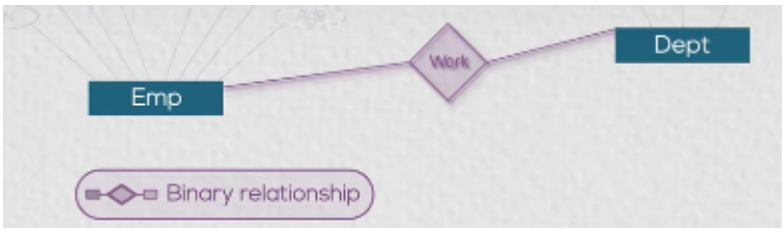
1- Degree of a relationship (# of participating entity)

2- Cardinality Ratio

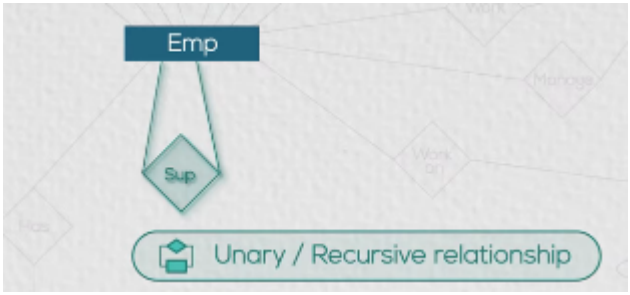
3- Participation

Types of relationship

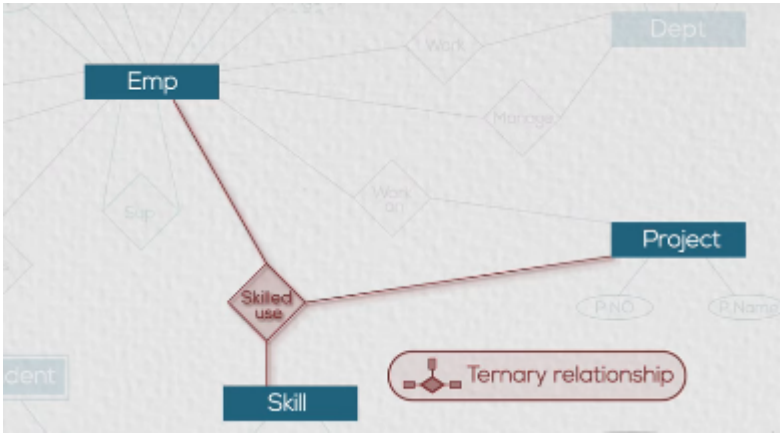
A) **Binary relationship** (which mean it has two entities)



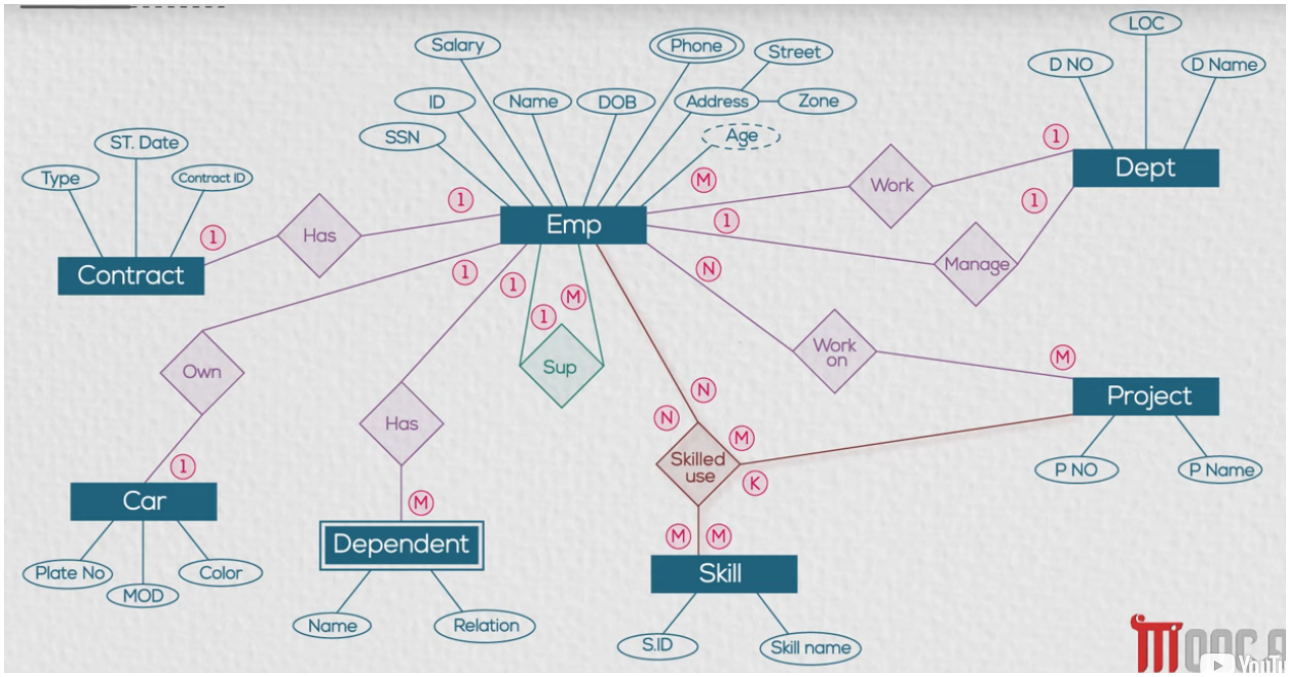
B) **Recursive/ Unary relationship** (The relationship between the entity and itself)



C) **Ternary Relationship**(A relationship includes three entities)



Relationship-Cardinality Ratio



Cardinality Ratio Specifies the maximum number of relationship

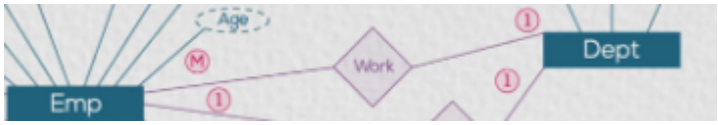
for example :- one employee to work on only one dept. or he can work in more than one dept.

So we write the ratio in the same way we asked the question

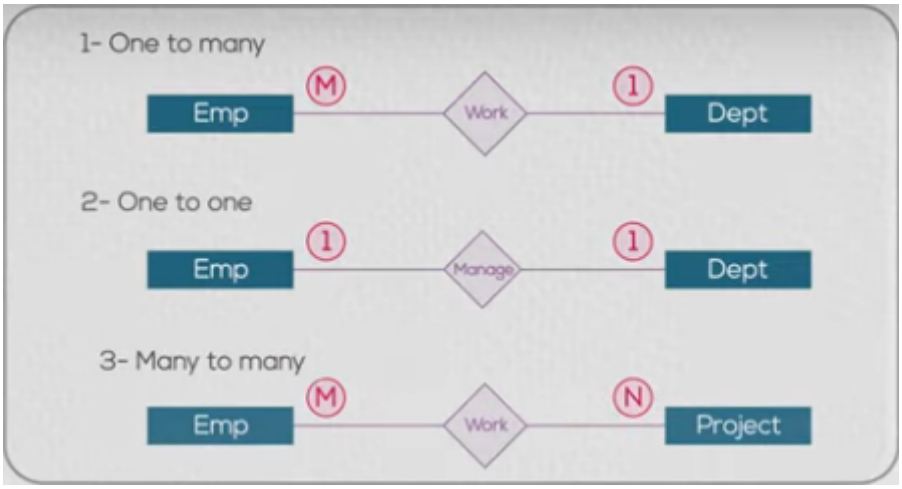
for example

the cardinality ratio between EMP and DEPT. We ask if there's one employee can work on one dept. or many dept.

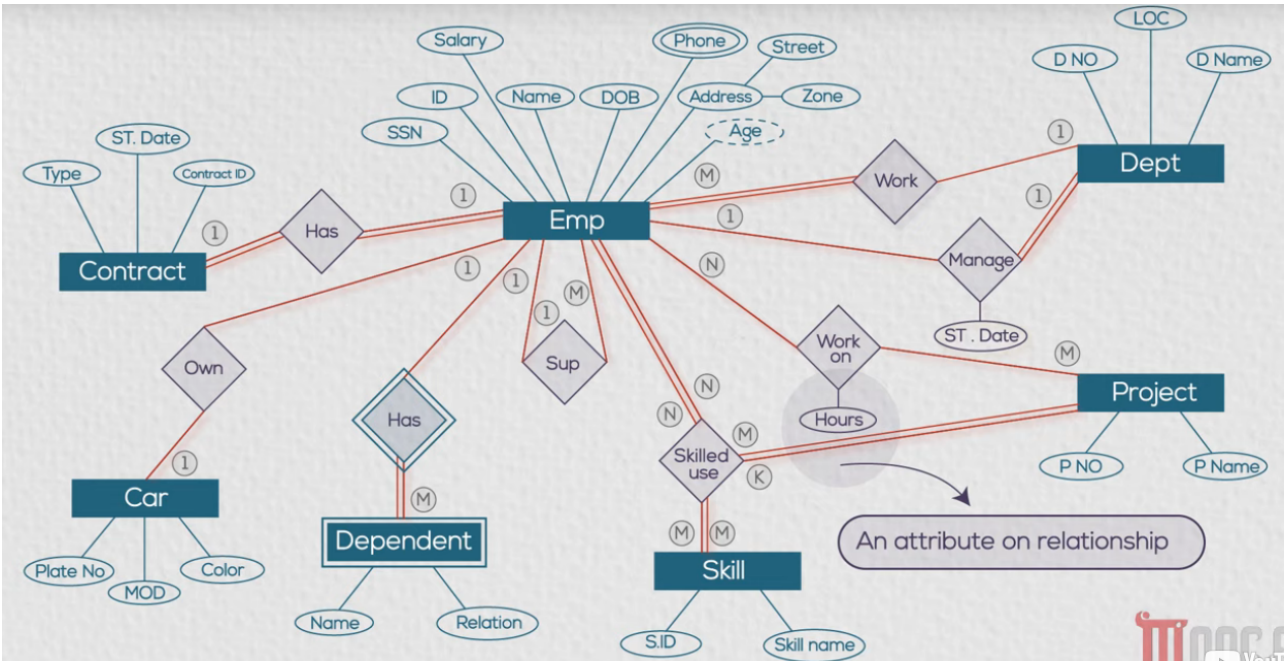
we find that each employee can work on one dept.
and then we revise the question each dept. has one employee to work on or many we find that each dept. has many employee



The types of cardinality ratios are:-



Relationship-Participation



Participation is opposite to cardinality which represents the minimum number of relationships that can occur with these instances

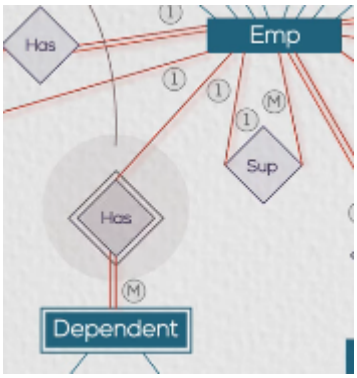
can be represented using **MUST** (double line) or **MAY** (single line)

So if this employee must work in a department so we need to put a double line to represents the "MUST"

Participation depends on the business scenario

NOTE

The relationship between the strong entity and weak entity is called <u>Identifying relationship</u> and it's represented in a double line



Chapter 03- ERD Mapping To Tables

Mapping strong and weak entities

Converting the conceptual design into a logical design representing the relationships we have.

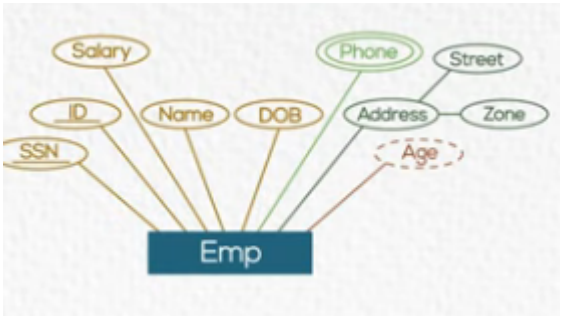
- relation database is a **set of tuples** or a **set of columns**
- the intersection between column and tuple called **DOMAIN**(single value)

Primary key It must contain a unique value for each row of data, It cannot contain Null values

Foreign key a value refers to a primary key in another table

Step 1: Mapping of regular entity types

create table for each entity type



Emp (ID, SSN, Salary, Name, DOB, Street, Zone)

Emp-phone(SSN, Phone)

كدة احنا حلينا مشكلة الكومبوزيت اتريبيوت بأننا خلينا ال

(SSN,Phone)

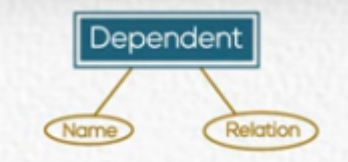
يبقوا هما البرايميري كي فكدة معناه انك مش هتقدر تكرر رقم بطاقة موظف والموبايل بتاعه

The derived attribute is not default to store it in database because it causes a headache for the database whenever I use this table it will calculate the employee's age so it delays the performance or the retrieval of database.

We store it in database when I always use the derived attribute for calculating a specific info or for referring to specific data or I retrieve the database based on the derived attribute.

Step 2: Mapping of weak entity types

Add foreign key that correspond to the owner entity type



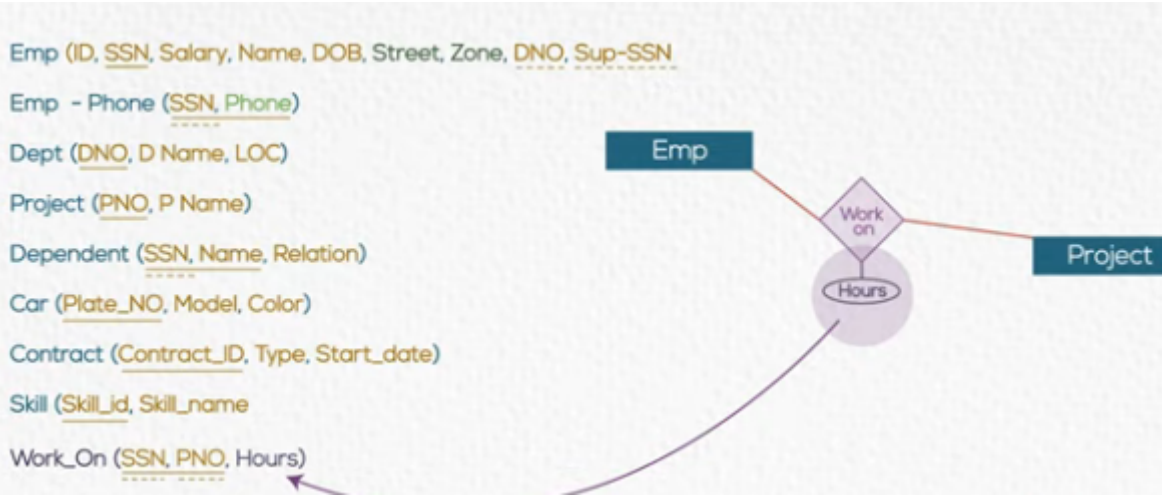
Mapping of Relationship Types

Step 3 : Mapping of relationship types Binary/ Unary 1:N

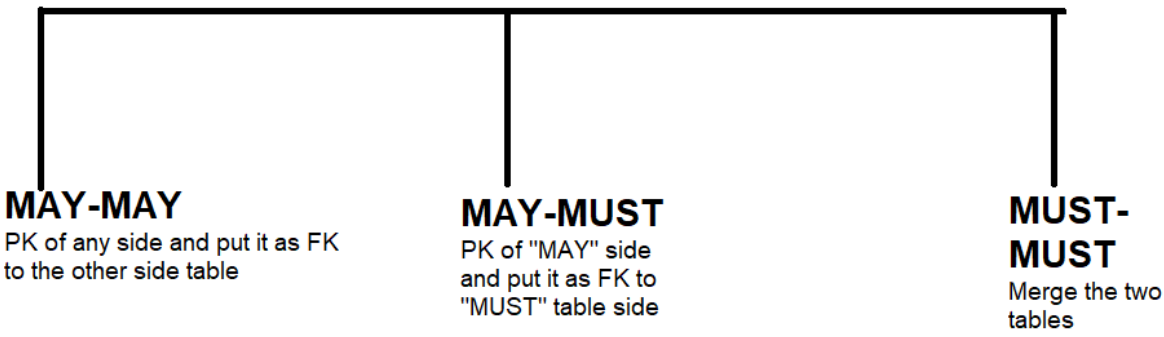
Take the **Primary key** of **one side** and put it to as a **Foreign key** to **N-side table**

Step 4 : Mapping of relationship types Binary/ Unary M:N

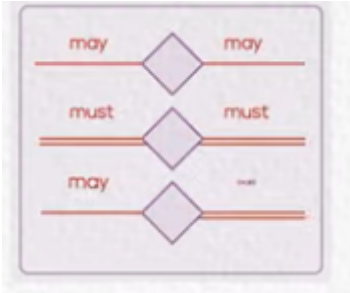
Take the **Primary keys** of participating entities and put it as **Foreign keys** to a new table



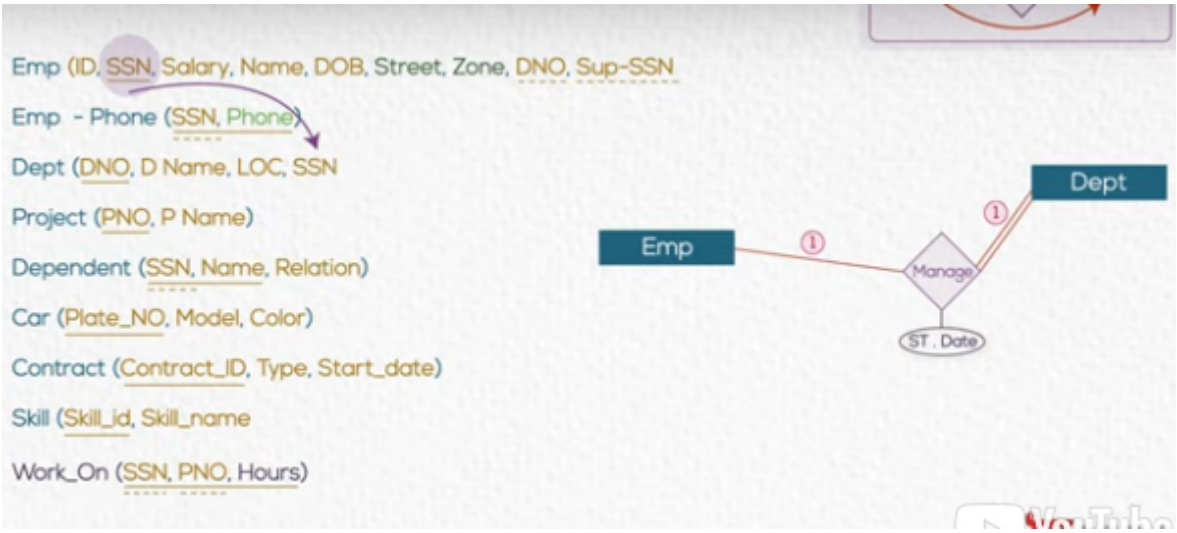
Step 5 : Mapping of relationship types Binary/ Unary 1:1



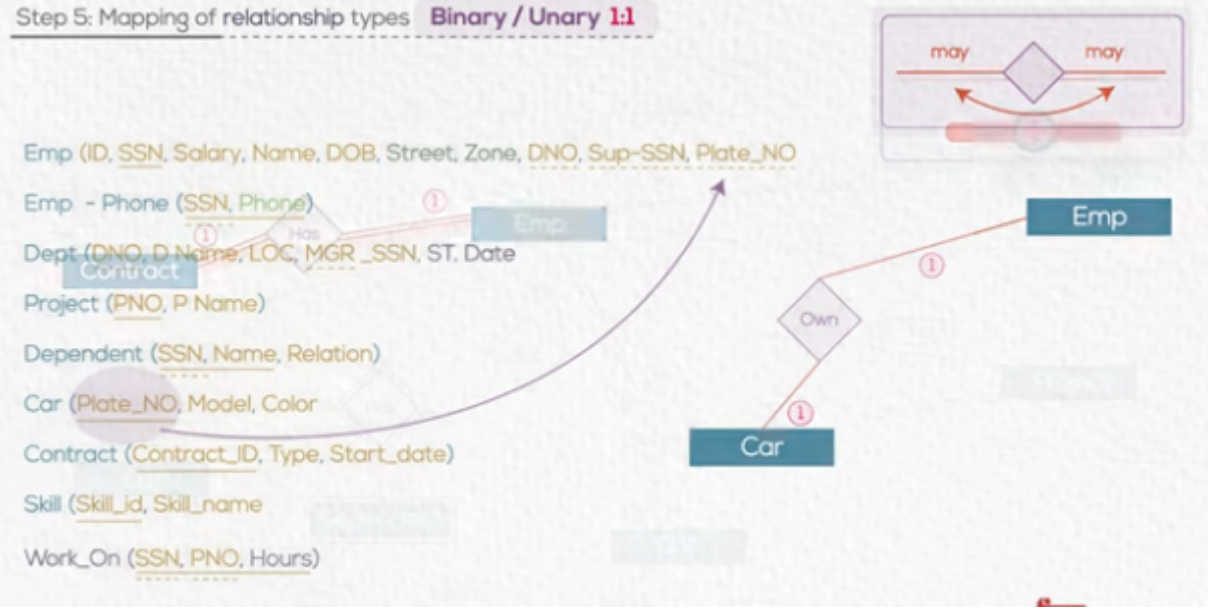
There are three cases **according to the participation**:



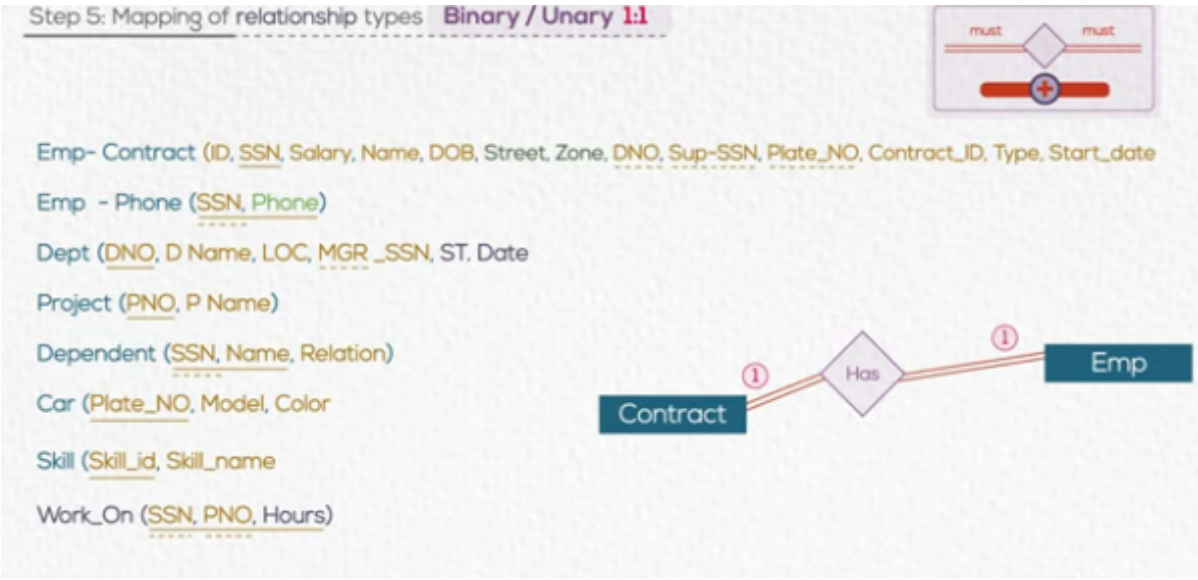
1) May-Must : Take the **Primary key** of "May" side and put it as a **Foreign key** to "Must" Table



2) May-May : Take the **Primary key** of any side of both and put it as a **Foreign key** to the table of the other side.

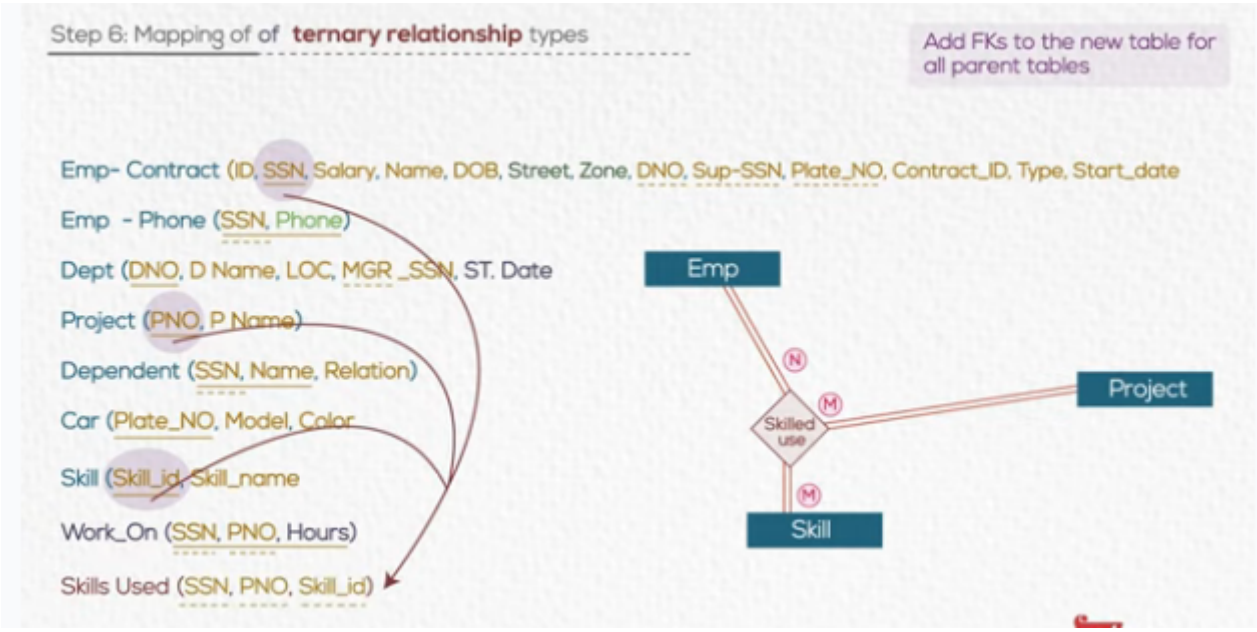


3) Must-Must : Merge between the two tables.



Step 6: Mapping of ternary relationship types

Take the **Primary keys** of all the parents and put it as a **Foreign keys** to a new table. (**Like M:N relationship**)



Chapter 04- Structured Query Language (SQL)

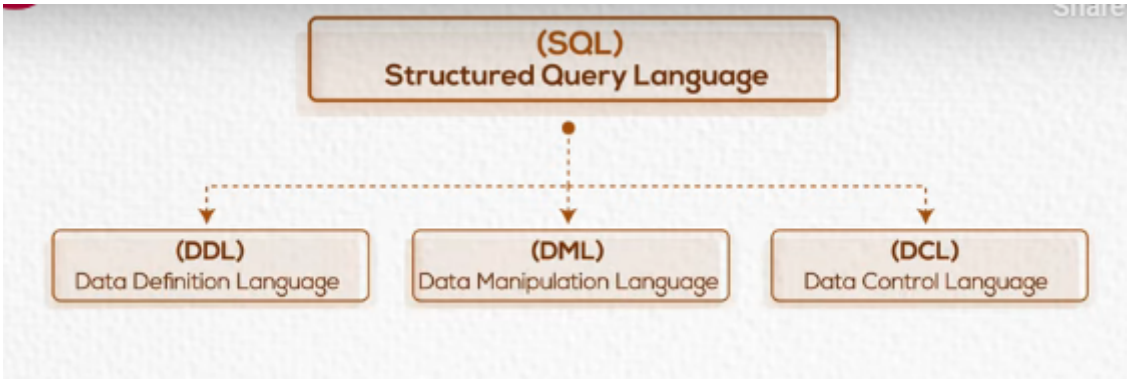
SQL is a programming language helps us to use database.

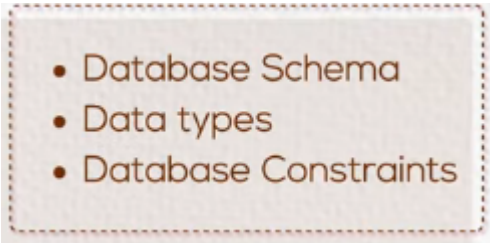
SQL is divided into 3 main categories:

1- **DDL** (Data Definition Language)

2- **DML** (Data Manipulation Language)

3- **DCL** (Data Control Language)





Database Schema :

- A group of related objects in the database
- There is one owner of a schema who has access to manipulate the structure of any object in the schema
- A schema doesn't represent a person, although the schema is associated with a user that doesn't represent him.

Data Types

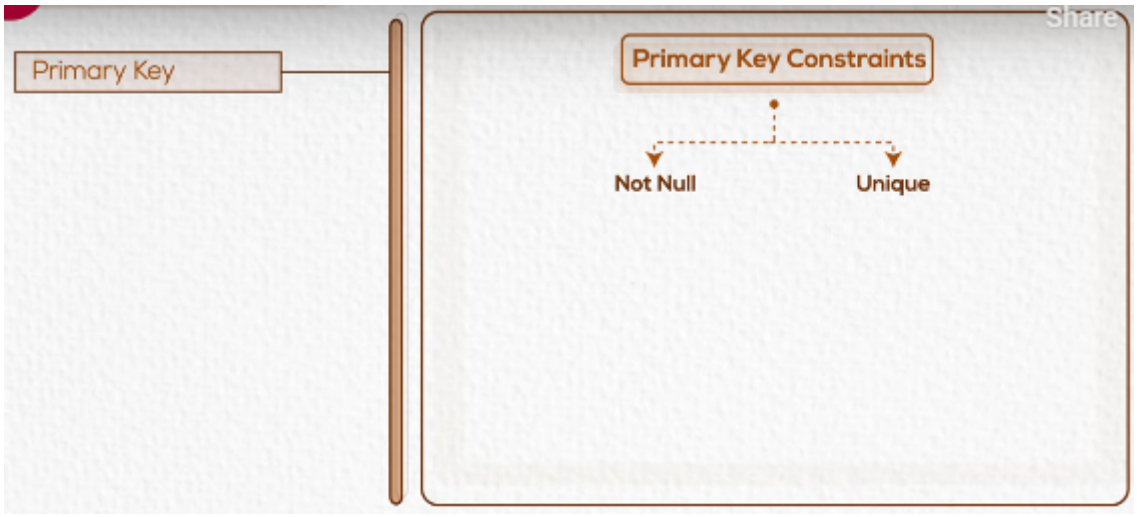
- A data type determines the type of data that can be stored in a database column
- The most commonly used data types are (Alphanumeric, Numeric, Date and Time, Characters, Integer, Float data type, etc..)

Database Constraints

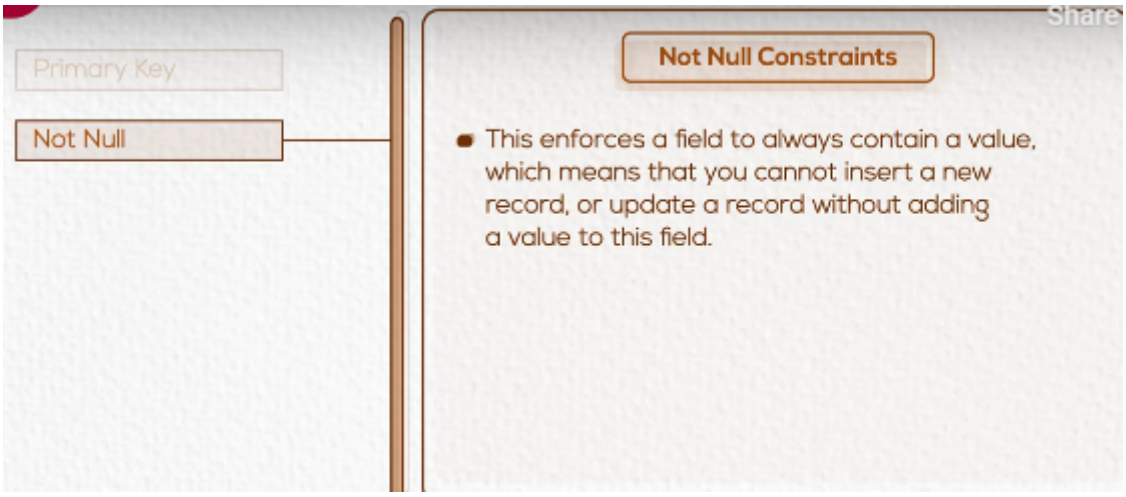
- Restrictions on database table r object to help **Maintain integrity of data**

For Example:-

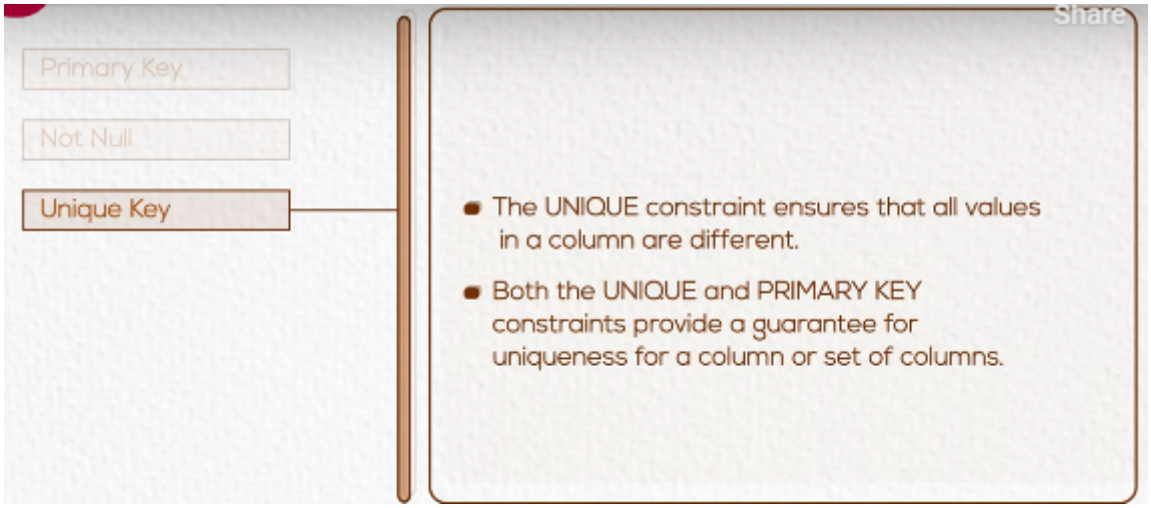
Primary Key



Not Null Constraints



Unique Key



Referential Integrity (FK)

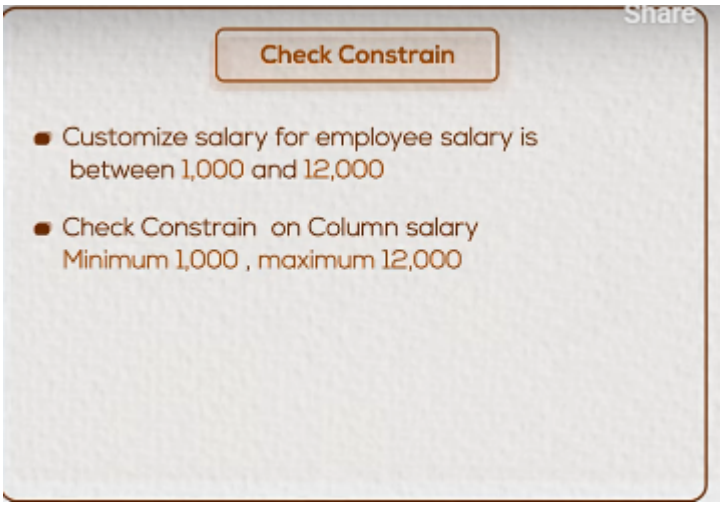
The referential integrity works when we:-

- Insert
- Delete

During insertion I insert in the parent record first, then I insert in child record

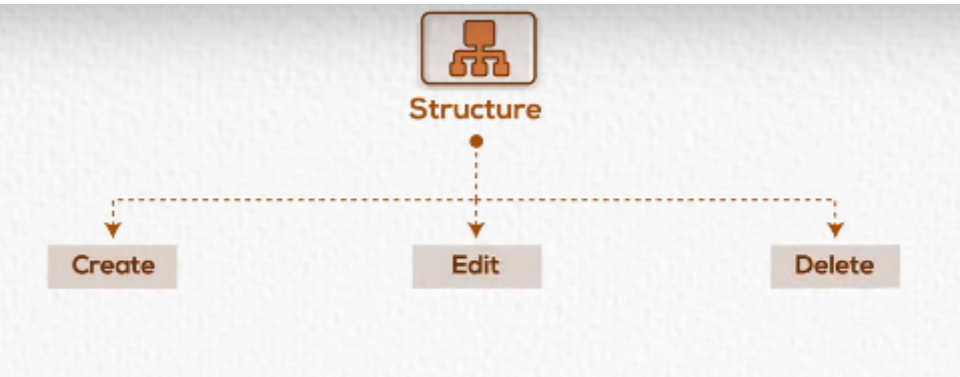
During deletion I delete from child record, then I delete from parent record.

Check Constraint



SQL - Data Definition Language (DDL)

They are commands responsible for the structure of database. This helps to Create, Edit, or Delete table **But not to manipulate the data itself**



The commands are:-

- **CREATE** command
- **ALTER** command
- **DROP** command
- **TRUNCATE** command

1. **CREATE**

```
CREATE TABLE Students
(ID PRIMARY KEY, First_Name CHAR(50)NOT NULL, Last_Name CHAR(50)NOT
NULL, Address CHAR(50), City CHAR(50), Country CHAR(25), Birth_Date
DATE)
```

To Add or remove a column in a table we use ALTER command then the query we want for example:-

2- **ALTER with ADD**

```
ALTER TABLE Students ADD Postal_Code
```

ALTER with REMOVE

```
ALTER TABLE Students REMOVE Postal_Code
```

3-**DROP** To remove the whole table

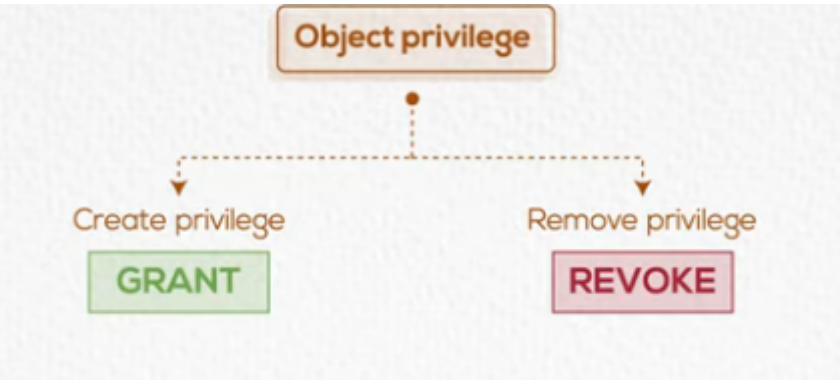
```
DROP TABLE Students
```

SQL- Data Control Language (GRANT, REVOKE)

We use the two commands to give the user a **Privilege** access to data

There are two types of Privileges

1. System Privileges
2. Object Privileges (**It includes the permissions you give the user for the database objects you have either it's a table or whatever the database**)



For Example:-

1. **GRANT command**

```
GRANT SELECT ON TABLE Employee TO Ahmed
```

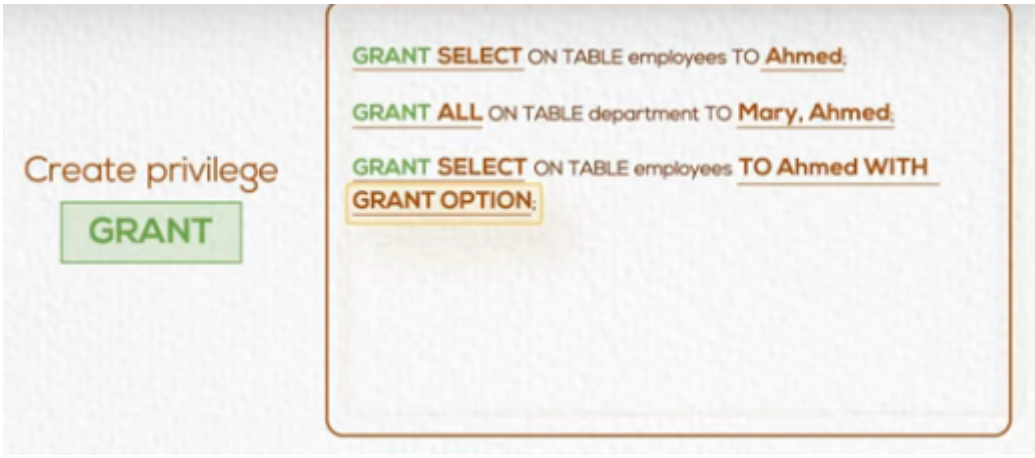
So I give the User who called Ahmed the access to table Employee and to only select/ view the table but he cannot add or remove data

```
GRANT ALL ON TABLE Employee TO Mary, Ahmed
```

IN the previous code we give Mary, Ahmed all the DMLs command which are (INSERT, UPDATE, DELETE, SELECT)

```
GRANT SELECT ON TABLE Employee TO Ahmed WITH GRANT OPTION
```

GRANT OPTION means that Ahmed has the ability to let anyone select/view the table Employee



2. **REVOKE** is responsible for removing the privileges from the users

```
REVOKE UPDATE ON TABLE department FROM Mary
```

Remove the update command from the user name Mary on department table

```
REVOKE ALL ON TABLE department FROM Mary, Ahmed
```

Remove all commands from the users name Mary and Ahmed on department table

SQL- Data Manipulation Language (DML)

1. INSERT

```
INSERT INTO Students(ID,First_Name,Last_Name,Address, City,
Country, Birth_Date)
VALUES(211,'Ahmed','Abdallah','20 el-haram
st','Giza','Egypt','20/11/2001')
```

OR

If you know the attributes of the table and there order

```
INSERT INTO Students
VALUES(211,'Ahmed','Abdallah','20 el-haram
st','Giza','Egypt','20/11/2001')
```

OR If you insert specific data to only specific columns

```
INSERT INTO Students(ID,First_Name,Last_Name)
VALUES(211,'Ahmed','Abdallah')
```

Insert multiple rows once

```
INSERT INTO Students(ID,First_Name,Last_Name,Address, City,
Country, Birth_Date)
VALUES(213,'Ahmed','Abed','19 el-ahram
st','Cairo','Egypt','10/01/2003');

INSERT INTO Students(ID,First_Name,Last_Name,Address, City,
Country, Birth_Date)
VALUES(214,'Saeed','Abdallah','1 el-areesh
st','Giza','Egypt','20/11/2001');

INSERT INTO Students(ID,First_Name,Last_Name,Address, City,
Country, Birth_Date)
VALUES(216,'Ahmed','Mohamed','20 el-hamra
st','Tanta','Egypt','20/12/2005');
```

2. UPDATE

```
UPDATE Students
SET Birth_Date = '20/11/2001'
WHERE ID = 211
```

We use WHERE to choose a specific record

3. DELETE

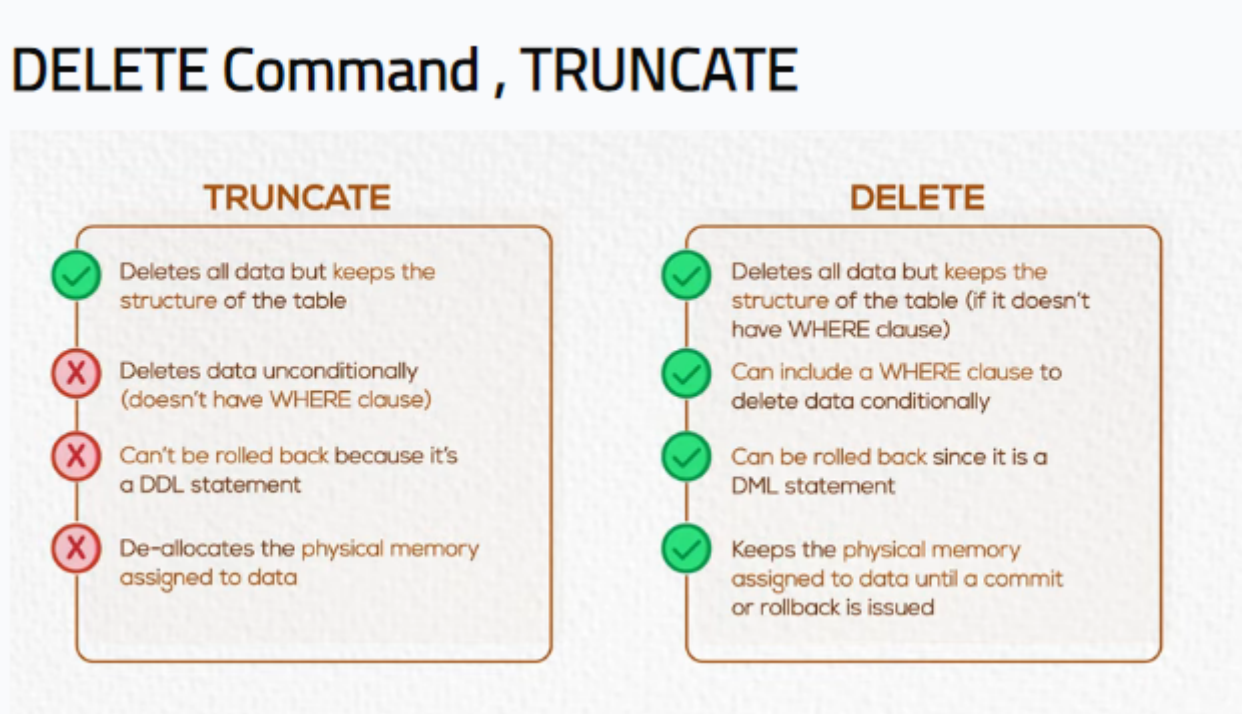
```
DELETE FROM Students
WHERE ID = 211
```

The difference between DELETE and TRUNCATE (DDL command)

TRUNCATE table is functionally identical to DELETE statement with no WHERE clause

```
TRUNCATE TABLE customer
```

remove all data for customer but keeps the structure of the table



When we use **TRUNCATE** it deletes the data and de-allocate the physical memory assigned to data so we cannot **roll-back (Undo)** the command,

On other hand the **DELETE** command deletes the data but keeps the physical memory assigned to data until we do **COMMIT** which saves the query we have issued or roll-back is issued

4. SELECT (View the data using it)

```
SELECT First_Name FROM Students
WHERE First_Name = 'Ahmed'
```

Output

First_Name
Ahmed
Ahmed
Ahmed

```
SELECT *
from Students
WHERE First_Name = 'Ahmed'
```

Output

ID	First_Name	Last_Name	Address	City	Country	Birth_Date
213	Ahmed	Abed	19 el-ahram st	Cairo	Egypt	10/01/2003
216	Ahmed	Mohamed	20 el-hamra	Tanta	Egypt	20/12/2005

```
SELECT ID, First_Name, Last_Name
FROM Students
```

Output

ID	First_Name	Last_Name
213	Ahmed	Abed
214	Saeed	Abdallah
216	Ahmed	Mohamed
211	Ahmed	Abdallah

Comparison & Logical operators

1. **WHERE**

We use ' = ' operator when we compare exact values

```
SELECT ID, First_Name, Last_Name, GPA
FROM Students
WHERE GPA > 2
```

Output

ID	First_Name	Last_Name	GPA
214	Saeed	Abdallah	2.3
216	Ahmed	Mohamed	2.22
211	Ahmed	Abdallah	2.4

```sql
SELECT ID, First_Name, Last_Name, GPA
FROM Students
WHERE GPA BETWEEN 2.23 AND 2.4


//OR by using single row operator

SELECT ID, First_Name, Last_Name, GPA
FROM Students
WHERE GPA >=2.23 AND GPA <= 2.4
```

Output

ID	First_Name	Last_Name	GPA
214	Saeed	Abdallah	2.3
211	Ahmed	Abdallah	2.4

```sql
SELECT ID, First_Name, Last_Name, GPA
FROM Students
WHERE GPA IN (2.22,2.4)

//OR by using single row operator

SELECT ID, First_Name, Last_Name, GPA
FROM Students
WHERE GPA =2.22 OR GPA = 2.4
```

Output

ID	First_Name	Last_Name	GPA
216	Ahmed	Mohamed	2.22
211	Ahmed	Abdallah	2.4

2. **LIKE**

We use it when we don't know the exact value we are looking for

for example we are looking for the student whose last name has "b" in the second place

```
SELECT *
FROM Students
WHERE Last_Name LIKE '?b*'
```

we are looking for the student whose la first name is "Ahmed" but we don't know the spelling

```
SELECT *
FROM Students
WHERE First_Name LIKE 'Ahm?d'
```

3. **ALIAS** we use it when we do a calculation on a specific column and we want to display the result in a new column (**just for displaying**)

```
SELECT First_Name + ' '+Last_Name AS [Full Name]
FROM Students
WHERE GPA >2.3
```

4. **ORDER BY** arrange the data (**By default it arranges the values ascendingly**)

```
SELECT *
FROM Students
ORDER BY First_Name ASC, Last_Name DESC
```

Output

ID	First_Name	Last_Name	Address	City	Country	Birth_Date
216	Ahmed	Mohamed	20 el-hamra st	Tanta	Egypt	20/12/200
213	Ahmed	Abed	19 el-ahram st	Cairo	Egypt	10/01/200

5. **DISTINCT** remove duplicate value

```
SELECT DISTINCT First_Name , Last_Name
FROM Students
```

Output

First_Name	Last_Name
Ahmed	Abed
Saeed	Abdallah

6. **JOIN** Condition

    a. **Equi Join**

A condition showing how DBMS can link more than one table together and it's always a relationship between the Primary key and Foreign key

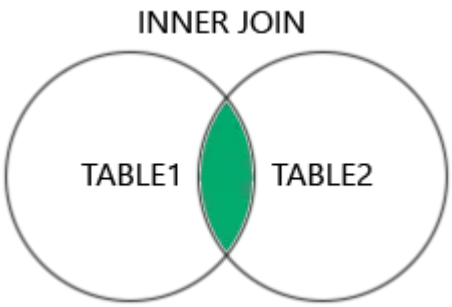
NOTE: The number of JOIN conditions is always equal to the number of tables -1 (t-1)

```
SELECT First_Name , SNAME
FROM Students as Stud , Subjects as Sub
WHERE Stud.SNO = Sub.SNO;
```

Output

First_Name	SNAME
Mohammed	MATH
Saeed	LANGUAGE
Ahmed	SCIENCE

```
2. **INNER JOIN**
```



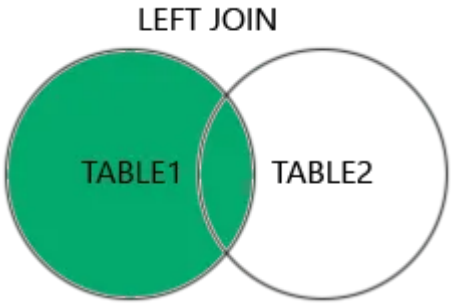

```
SELECT First_Name , SNAME
FROM Students as Stud INNER JOIN Subjects as Sub
ON Stud.SNO = Sub.SNO;
```

Output

First_Name	SNAME
Mohammed	MATH
Saeed	LANGUAGE
Ahmed	SCIENCE

3. **OUTER, FULL JOIN**
    1. **LEFT JOIN**

It return the left table and the intersection data with the right table.



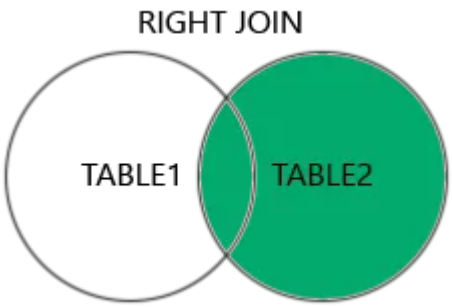

```
SELECT First_Name , SNAME
FROM Students as Stud LEFT JOIN Subjects as Sub
ON Stud.SNO = Sub.SNO;
```

Output

First_Name	SNAME
Mohammed	MATH
Saeed	LANGUAGE
Ahmed	SCIENCE

```
2. **RIGHT JOIN**
```

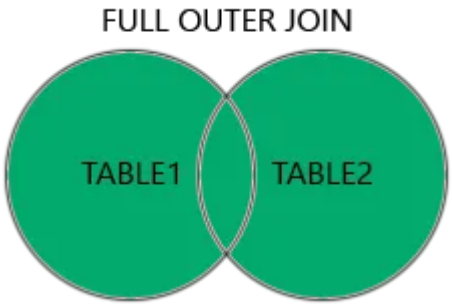
It returns the full right table and the intersection with the left table.



```
SELECT First_Name , SNAME
FROM Students as Stud RIGHT JOIN Subjects as Sub
ON Stud.SNO = Sub.SNO;
```

3. **FULL JOIN**

```
SELECT First_Name , SNAME
FROM Students as Stud FULL JOIN Subjects as Sub
ON Stud.SNO = Sub.SNO;
```



7. **Sub-query**

it's a query inside the main query.

we use it when we want to get an info that isn't available directly and need to access a specific table to get such info

```
SELECT *
FROM Students
where GPA >(SELECT GPA FROM Students WHERE First_Name='Saeed'and
Last_Name='Abdallah')
```

Output

ID	First_Name	Last_Name	Address	City	Country	Birth_Date	GPA	SNO
213	Mohammed	Abed	19 el-ahram st	Cairo	Egypt	10/01/2003	2.1	1101
216	Ahmed	Mohamed	20 el-hamra st	Tanta	Egypt	20/12/2005	3	1104

8. **Aggregate Functions**

Functions that take multiple inputs and return only one result, aggregate functions ignore the NULL values

Such as (MAX, MIN, SUM, COUNT , AVERAGE)

```
SELECT MAX(GPA) AS MAX, MIN(GPA) AS MIN
FROM Students;
```

Output

MAX	MIN
3	1.4

9. **GROUP BY & HAVING**

Group by : we use it to divide data of a table based on the attribute you input in "GROUP BY"

```
SELECT  SEX,AVG(GPA) AS [GPA AVERAGE]
FROM Students
GROUP BY SEX
```

Output | Available Tables

Sex	GPA AVERAGE
F	3.4
M	2.1666666666666665

if we want to put a condition based on aggregate function so we use "HAVING" clause

FOR EXAMPLE :

```
SELECT  SEX,AVG(GPA) AS [GPA AVERAGE], SNO
FROM Students
GROUP BY SEX
HAVING MAX(SNO) > 1104
```

Output | Available Tables

Sex	GPA AVERAGE	SNO
F	3.4	1105

Chapter 06_SQL- Other DB objects

VIEWS(Create, Modify, Remove, Types)

1. View

- A database object
- It's a logical table based on a table or another view
- A view contains no data of its own, but is like a **window through** which **data from tables** can be viewed or changed
- The tables on which a view is based are **called base tables**
- The **view** is stored as a **SELECT statement** in the data dictionary (metadata)

Creating Views

EXAMLPE:-

```
CREATE VIEW vw_work_hrs
AS
SELECT Fname,Lname, Pname, Hours
FROM Employee, Project, Works_on
WHERE SSN=ESSN AND PNO=PNUMBER
```

Creating Views with Check Option

Check Option whenever you create any DML on this view to **edit data or store data in it** for example, you have to confirm that the Where condition is satisfied.

```
CREATE VIEW Suppliers
AS
SELECT*
FROM suppliers
WHERE status>15
WITH CHECK OPTION; //it must validate the where condition every
time when creating any DML to insert data or to update data
```

Modifying View

CREATE OR REPLACE VIEW view_name

```
CREATE OR REPLACE VIEW vw_work_hrs
AS
SELECT Fname, Lname, Pname, Hours
FROM
Employee, Project, Works_on
WHERE SSN=ESSN AND PNO=PNUMBER AND Dno=5
```

Advantages of using Views:

1. Restrict data access
2. Make complex queries easy
3. Provide data independence
4. Present different view of the same data

امتي نستخدم الفيوز او فوائدها ،، بنستخدم الفيوز لو عاوزين نتيح لشخص معين انه يشوف جزء من جدول ما فساعتها بناخد الجزء اللي احنا عاوزينه يشوفه ويتحكم فيه ونعمل في فيو وديله الاحقية انه يستخدمه لأن الدي سي إل بتدينا السماحية في اننا ندي لحد احقية الاستخدام او التعديل علي كل الجدول مش بعض منه

فالفيو بقي بتحل المشكلة دي انها بتخلينا نقدر نجزء الداتا اللي عندنا علي حسب اللي محتاجينه

Types of Views:

Feature	Simple Views	Complex Views
Number of tables	One	One or more
Contain functions	No	Yes
Contain groups of data	No	Yes
DML operations through a view	Yes	Not always

2. Indexes

We use indexes to solve the following problems

1. the data is not sorted by default
2. the data is scattered in the physical memory

When we insert data the DBMS inserts it in the physical memory. once I add a new record later, it searches for a space in the memory whether it's beside the data I added before or far from it, and puts the data in it. So the data is scattered in the memory not ordered together. There are some empty spaces or allocated spaces of memory between them.

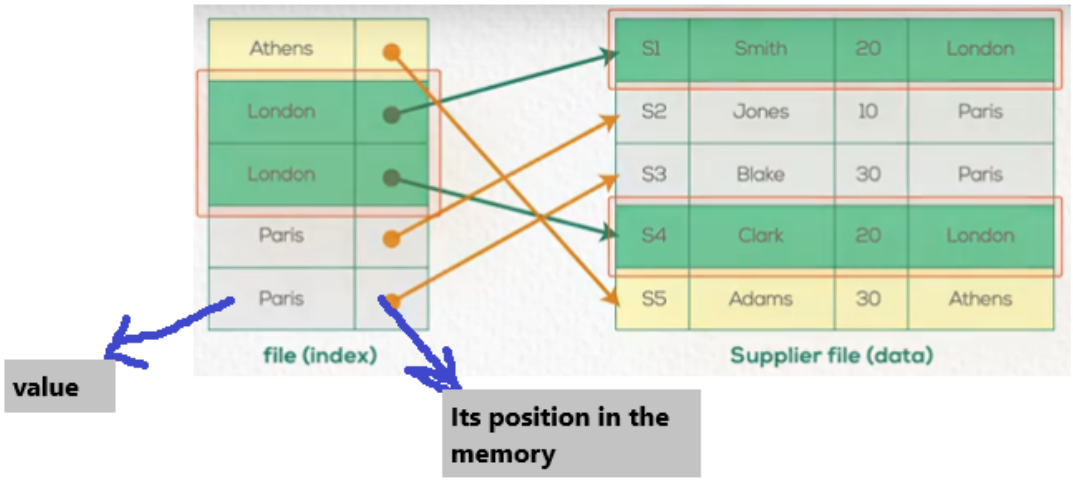
The idea of indexes

It's like the phone note we used it in the past

the idea is we order the phone note alphabetically and save the number with its corresponding name according to the first letter of its name and when we want to search for a specific name we just reach the page with a letter of its first name and we find the name. That's exactly what the indexes do.

Why indexes?

- They are used to speed up the retrieval of records in responsible to certain search conditions.
- May be defined on multiple columns
- Can be created by the useror by the DBMS
- Are used and maintained by the DBMS



مثال علي الانديكس ،،،دلوقتي لو عندنا جدول فيه داتا كتير (الصبلاير فايل) واحنا كل مرة بندور علي الصبلاير علي حسب البلاد فأحنا اولا هنستهلك وقت كبير علشان نعمل سكان للفايل كله ثانيا مش متأكدين ان مثلا لو دورنا علي لندن ان اول نتيجة هيطلعها للندن دي مفيش نتيجة تانية للندن زيها في الداتا فساعتها بناخد الداتا بتاعتنا دي ونحطها في انديكس بالترتيب وكل قيمة قدامها مكانها في الميموري او في الداتا فلما نيجي ندور هيبقي الموضوع اسهل واسرع

The disadvantages iof indexes

- Indexes cause overhead on DML (inser, update ,delete)

  لأننا لما بنيجي نعمل اي عملية من العمليات دي فأحنا بنبقي مضطرين اننا نعملها مرتين مرة في الجدول الأصلي ومرة في الاندكس ذات نفسه

Create an index when:

- Retrieving data heavily from table
- Columns are used in search conditions and joins
- Column contain large number of nulls

Don't create indexes when:

- Table is updated frequently

Creating of indexes

CREATE INDEX index_name ON table_name(column_name)

```
CREATE INDEX emp_indx ON Employee(Salary)
```

Chapter 07-Normalization

What's Normalization?

A process that takes a table through a series of tests (Normal Forms)

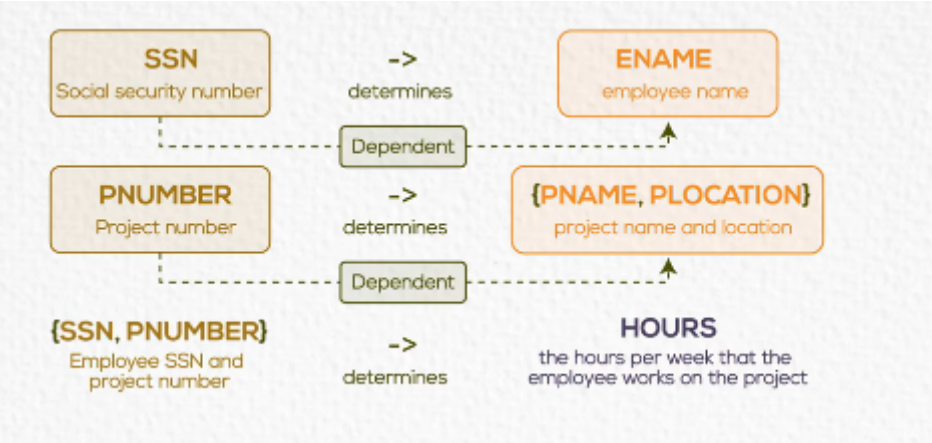
to

- Certifiy the goodness of a design and thus to **minimize redundancy and insert, update, delete anomalies, frequent null values**
- Use Normalization to another method for create a database design

Functional Dependency

A constraint between two attributes (columns) or two sets of columns

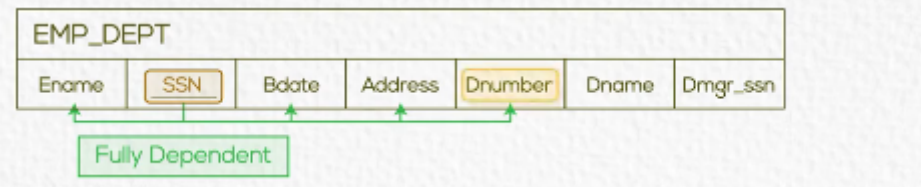
for every valid instance of A uniquely determines the value of B



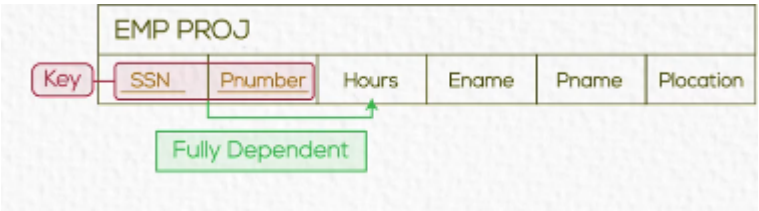
Types of functional dependency

1. **Fully dependency**

It means there's a non key attribute which is fully dependent on key, it depends completely on key



Ename fully,Bdate, Address, Dnumber are fully dependent on SSN

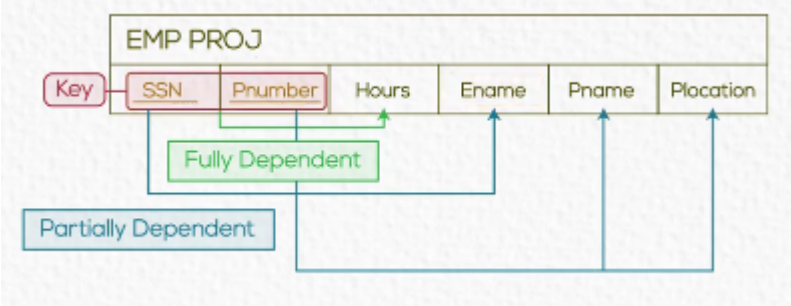


Hours is fully dependent on SSN,Pnumber (**composite primary key**)

بمعني انك ممكن تحدد عدد ساعات العمل بتاعت الموظف لما تعرف الرقم بتاعه ورقم المشروع لأن علي حسب نوع المشروع بيتحدد ساعات العمل

2. **Partially Dependency**

Non key attribute depends on part of the key

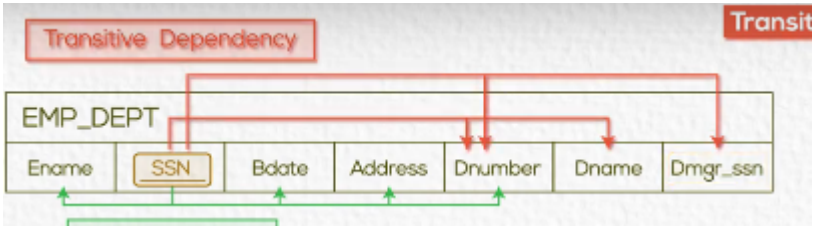


Ename is partially dependent on SSN

Pname,Plocation are partially dependent on Pnumber

3. **Transitive Dependency**

Non key attribute depends on non key attribute depends on key attribute



Dname depends on Dnumber

Dnumber depends on SSN

Dname is transitive dependent on SSN

First Normal Form

Normalization the process of decomposing unsatisfactory "bad" relations (tables contain some issues) by breaking up their ttributes into smaller relations(tables).

Normal Form Condition using keys and FDs of a relation to certifiy whether a relation schema is in a particular **normal form**(1NF first normal form, 2NF second normal form, 3NF third normal form)

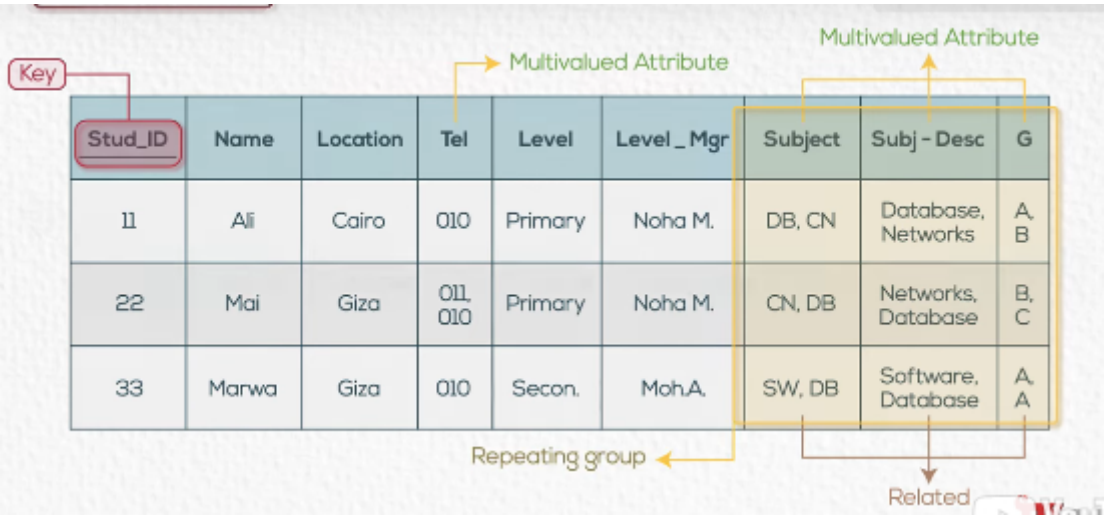
1. **1NF** When it doesn't contain:-

A) Multivalued Attribute

B) Repeating Group

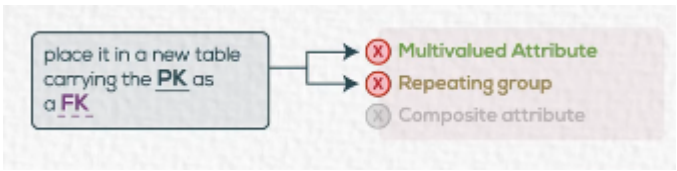
C) Composite Attribute

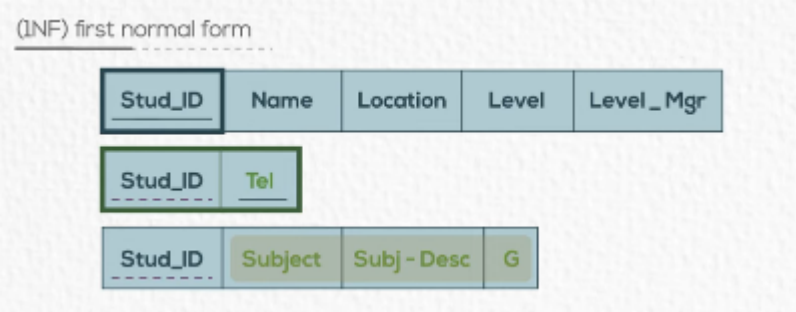
EXAMPLE:



So in this case we have two problems (Multivalued Attribute, Repeating Group)

We solve this problem by place it in a new table carrying the **PK** as a **FK**

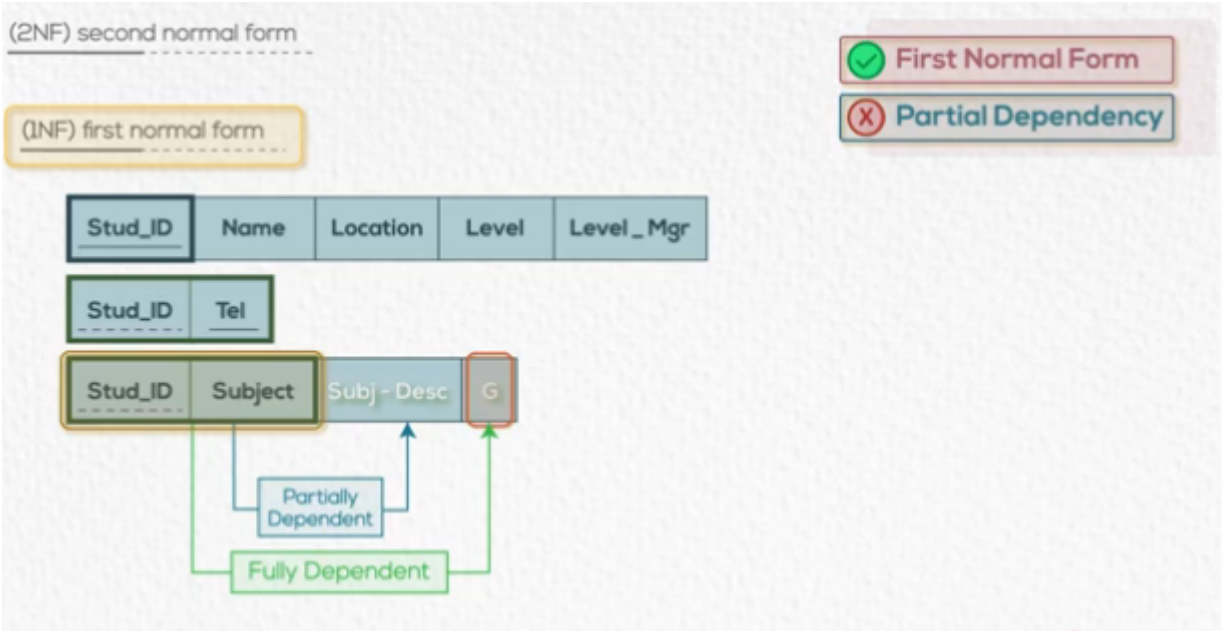




2. **2NF** When:-

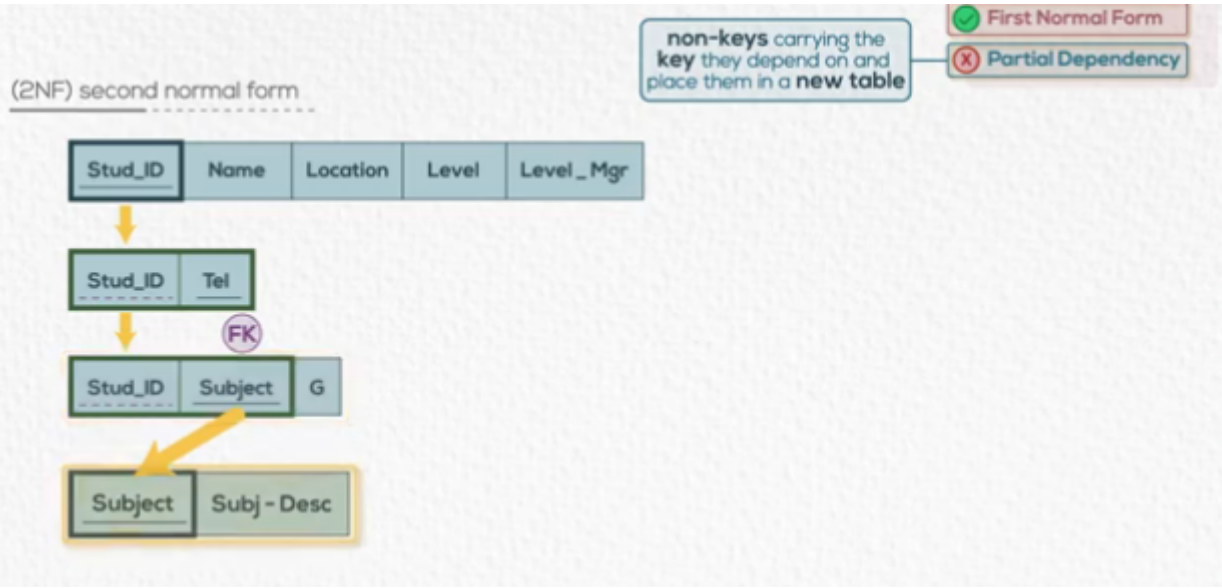
A) It applies 1NF

B) Doesn't contain partial dependency



we solve this by taking the

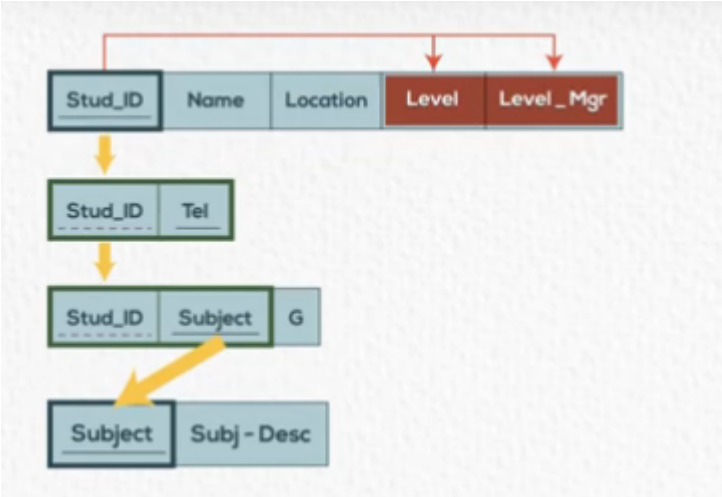
non-keys carrying the **key** they depend on and place them in a **new table**



3. **3NF** When:-

A) It applies 2NF

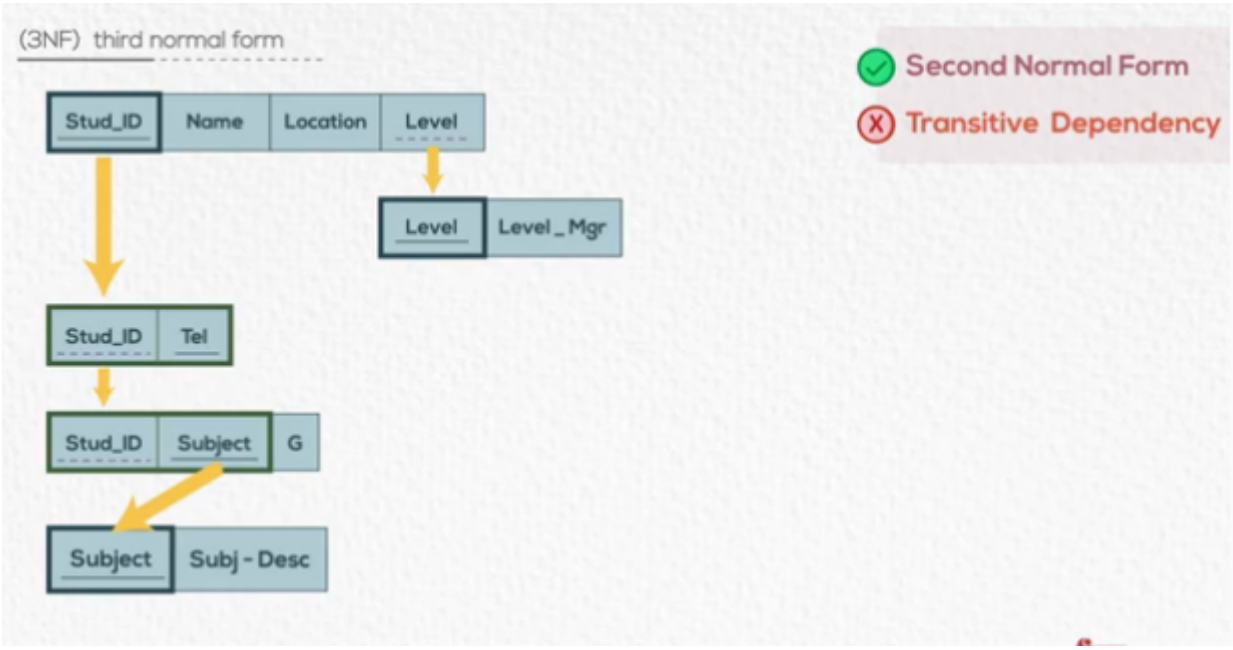
B) Doesn't contain Transitive Dependency



We solve this by

taking the case of transitive dependency and separate it into a new table

and make the dependent attribute a **PK** in a new table and **FK** in the old table



Normal Forms Summary