Database Fundamentals

Introduction

Limitations of file-based system:

- Separation & Isolation of data
- Duplication of data
- Program data dependence
- Incompatible file formats

What is a database?

It's a collection of related data.

What is database management system (DBMS)?

A software package/system to facilitate the creation and maintenance of a computerized database.

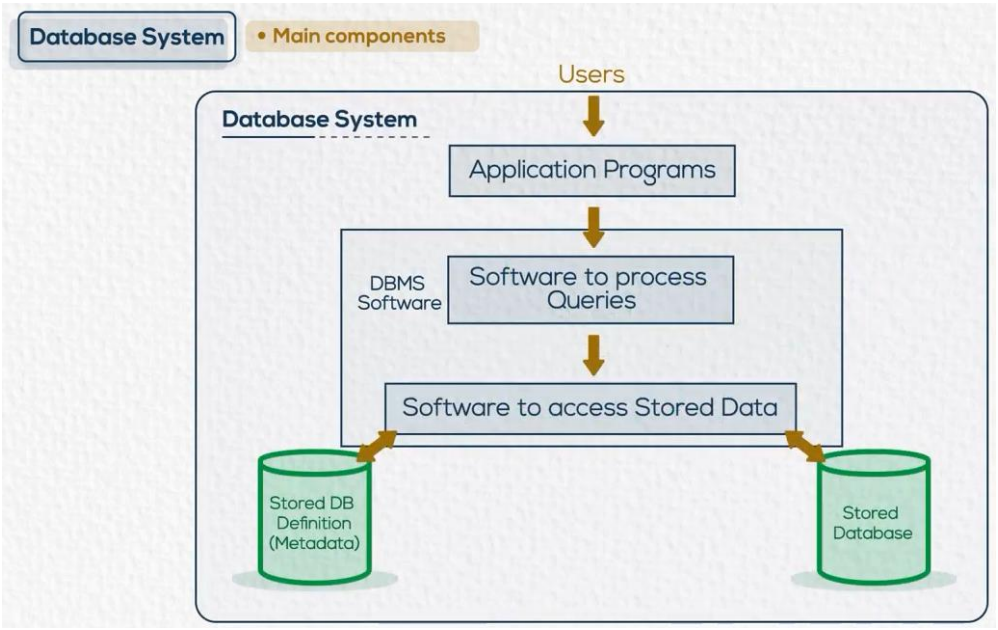
What is a database system?

It is the DBMS software together with the data itself. Sometimes, the applications are also included. (Software + Database)

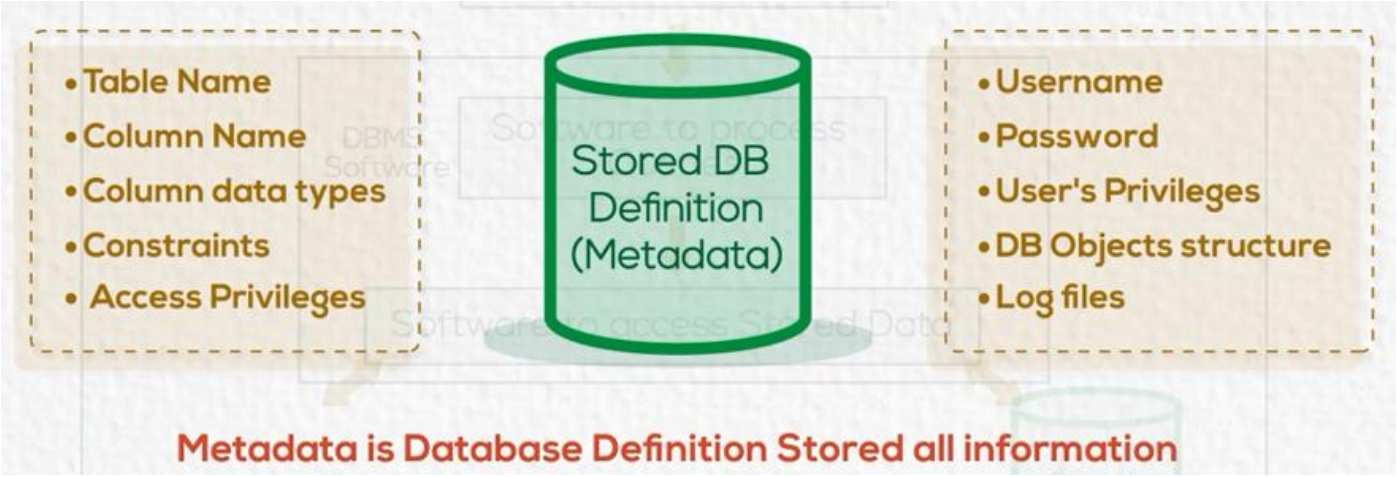
Difference between File System and DBMS:

Basis	File System	DBMS
Structure	The file system is software that manages and organizes the files in a storage medium within a computer.	DBMS is software for managing the database.
Data Redundancy	Redundant data can be present in a file system.	In DBMS there is no redundant data.
Backup and Recovery	It doesn't provide backup and recovery of data if it is lost.	It provides backup and recovery of data even if it is lost.
Query processing	There is no efficient query processing in the file system.	Efficient query processing is there in DBMS.
Consistency	There is less data consistency in the file system.	There is more data consistency because of the process of normalization.
Complexity	It is less complex compared to DBMS.	It has more complexity in handling as compared to the file system.
Security Constraints	File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file systems.
Cost	It is less expensive than DBMS.	It has a comparatively higher cost than a file system.
Data Independence	There is no data independence.	In DBMS data independence exists.
User Access	Only one user can access data at a time.	Multiple users can access data at a time.
Meaning	The user has to write procedures for managing databases	The user is not required to write procedures.
Sharing	Data is distributed in many files. So, it is not easy to share data	Due to centralized nature sharing is easy
Data Abstraction	It gives details of storage and representation of data	It hides the internal details of Database
Integrity Constraints	Integrity Constraints are difficult to implement	Integrity constraints are easy to implement
Example	Word Documents, Excel Sheets	Oracle, SQL Server

Main components of database system:



What is metadata:



Advantages of Database system:

- Controlling redundancy
- Restricting unauthorized access
- Sharing data
- Enforcing integrity constraints

ببقى متأكد ان الداتا بتخضع لـ logic rules (زى أن مفيش خانة المفروض يتحط فيها ارقام ويتحط فيها text) وكمان business rules (زى أن مفيش اتنين موظفين لهم نفس الـ id)

- Inconsistency(تضارب) can be avoided

يعنى لو حصل ابديت لداتا هتسمع في كل مكان مش هتحصل عند حد وحد لا

- Providing backup and recovery

Disadvantages of Database system:

- Needs expertise to use.
- DBMS is expensive (software and infrastructure).
- May be incompatible with any other available DBMS (can be avoided by using third party tool).

Cycle of having a database:

Step 1: Analysis and requirements gathering.

Achieved by System Analyst who is responsible for:

- Business analysis and requirements gathering.

Step 2: Database design

Achieved by Database Designer who is responsible for:

- Create database design (Conceptual Schema).

Step 3: Implementation

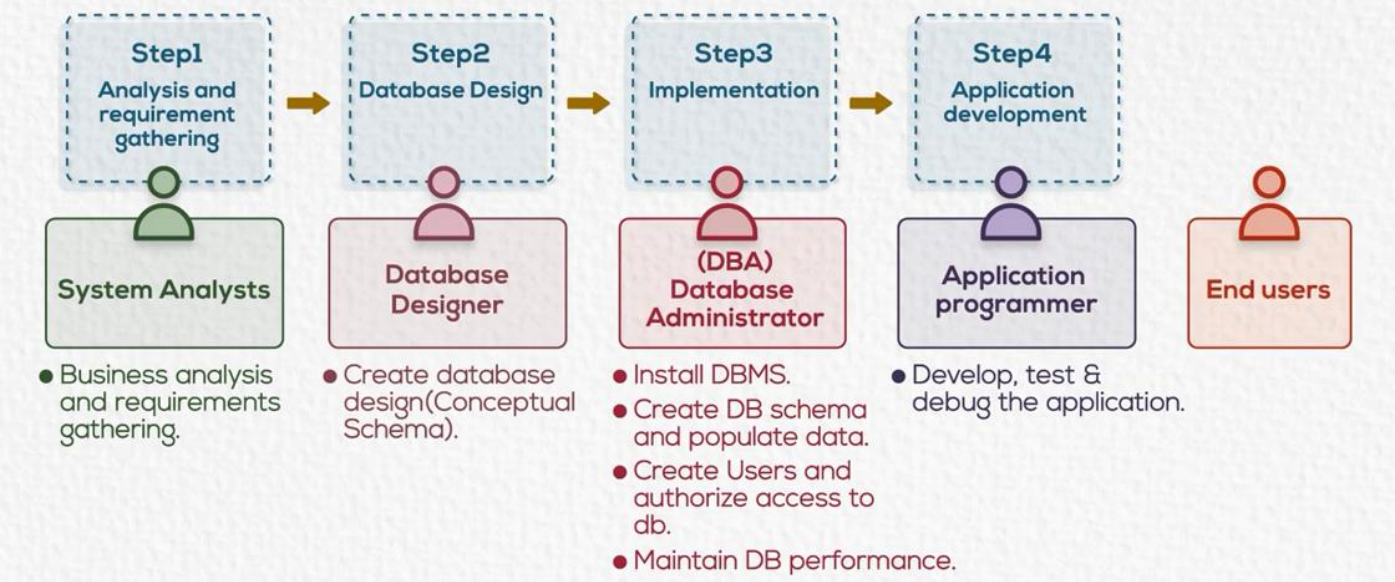
Achieved by Database Administrator (DBA) who is responsible for:

- Install DBMS.
- Create DB schema and populate data.
- Create users and authorize access to DB.
- Maintain DB performance.

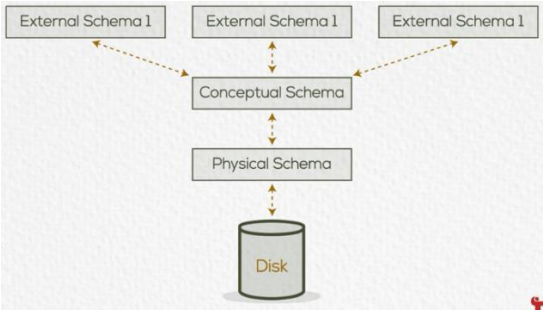
Step 4: Application development

Achieved by Application Programmer who is responsible for:

- Develop, test & debug the application.



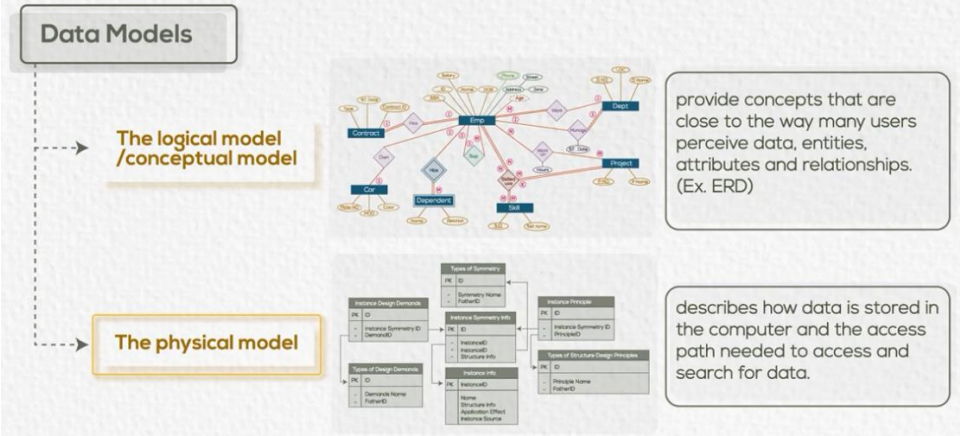
DBMS Architecture:



- **External Schema:** they are concerned with what data the user will see and how the data will be presented to the user (Could be more than one like financial schema, HR schema, etc).
- **Conceptual Schema (Logical model):** they are concerned with what is represented (define the database structures such as table sand constraints).
- **Physical Schema (Physical Model):** they are concerned with how the data are represented in the database, how the data structures are implemented.

The architecture is divided into schemas to achieve data independence (a change in any schema doesn't affect the other schemas).

Data Models:

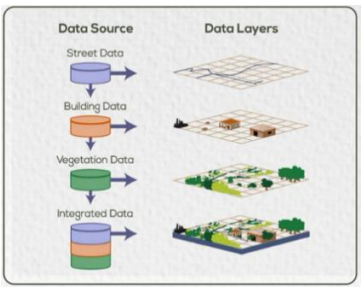


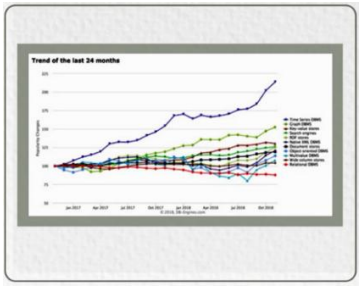
Mappings:

It's the process of transforming requests and results between levels (schemas).

DBMS other functionalities:

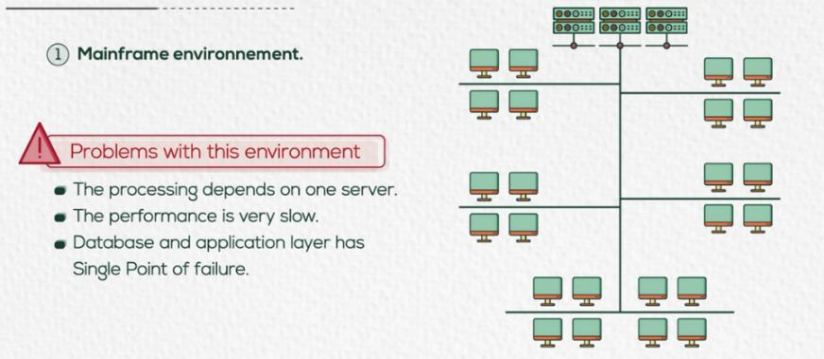
- Text/Number/Audio/Video/Image
- Spatial data
- Time series data
- Data Mining (very important)



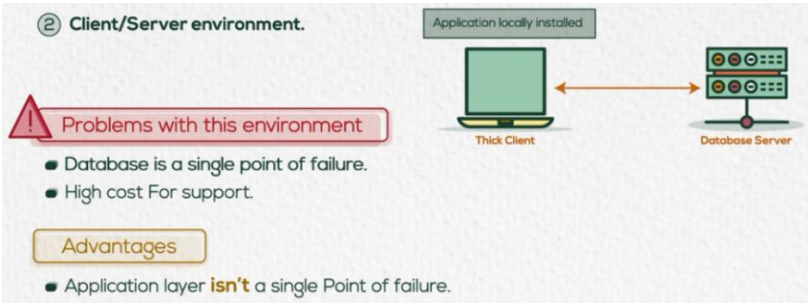


DBMS Environments:

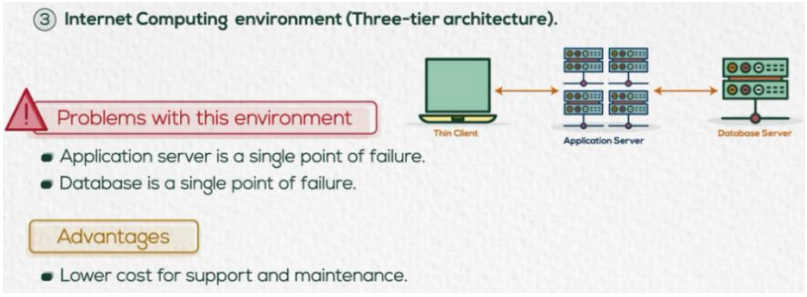
Centralized environment:



- هنا بيكون عندى mainframe واحد ومتوصل عليه مجرد monitors يعنى الـ database و الـ application على جهاز واحد وكل الأجهزة متوصله بيه عن طريق dummy terminals ببعت عليهم مجرد requests يعنى كل الـ processing على الـ mainframe.



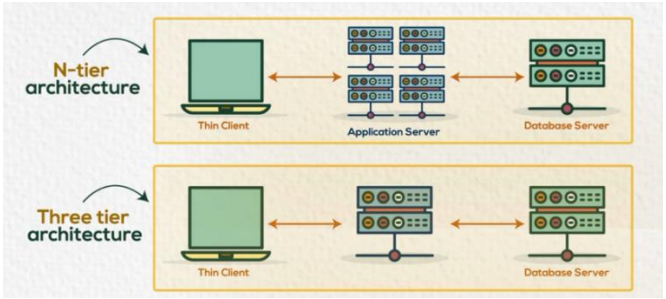
- بيكون مكون من 2 tiers وهنا الـ application بيكون موجود على الـ thick client وهنا انا خفيت الـ load بتاع الـ processing لكن بقى عندى مشكلة أن لو جيت اعمل ابديت مثلا لـ application هبقى مضطر اعمله على كل client ، وطبعا لسه عندى Single Point of Failure عند الـdatabase server.



- هنا الـ client بيكون thin عشان أنا بنزل عليه app صغير زى applet أو ممكن يدخل من الـ browser.

- كدة انا حليت شوية مشكلة الـmaintenance والـupdate لأنى ساعتها هعملها بس عند الـapplication server مش عند الـThin Clients.

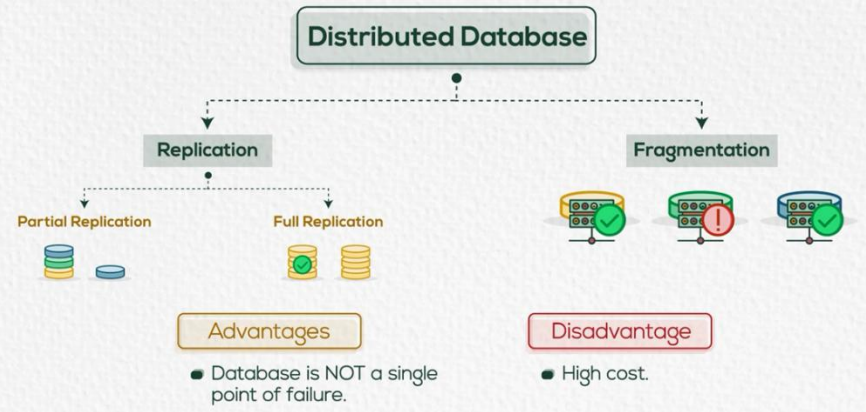
- مازال عندى هنا مشكلة point of failure على الـapplication tier وممكن أحلها عن طريق (n-tier architecture):



- وهنا بيكون عندى خيارين اما ان كل الـ applications يكونوا واحد وهنا كل الـclients متوزعين عليهم بحيث لو واحد وقع يكون في بدايل او حتى اقسم الـclients عليهم عشان اخف الـload.

- الخيار التانى انى اعمل different applications مثلا زى (HR app – financial app) ويكون كل واحد متوصل عليه الـclients بتوعه.

Distributed Environment:



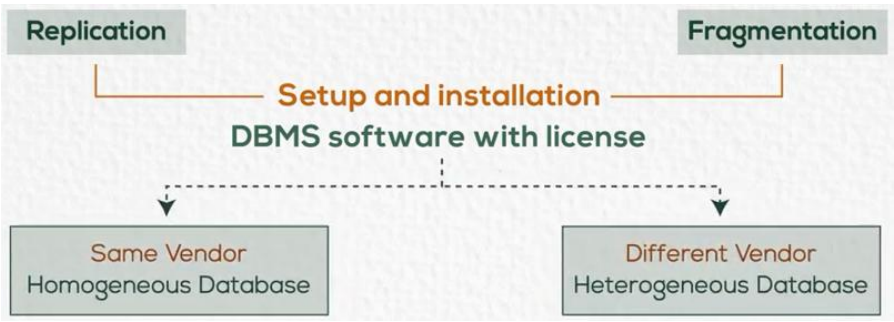
- الفكرة الرئيسية هنا انى بحل مشكلة الSingle Point of Failure عند الDatabase.

- أول ميزة هنا انها support high availability of database.

- في حالة الـ full replication بيكون عندى نسخة من الdatabase server والاتنين متوصلين ببعض بحاجة زى الheart beats كدة أول ما واحد يقع التانى بيشتغل وكل الـ requests بيحصلها rerouting على التانى.

- في حالة الـ partial replication بعمل كوبى لجزء بس من الداتا وبشتغل عليه وكل فترة باخد الchanges اللى عملتها على الreplica وبنقلها للheadquarter عشان يفضل up-to-date.

- في حالة الfragmentation باخد الfragments أما horizontal او vertical او hybrid بين الاتنين.

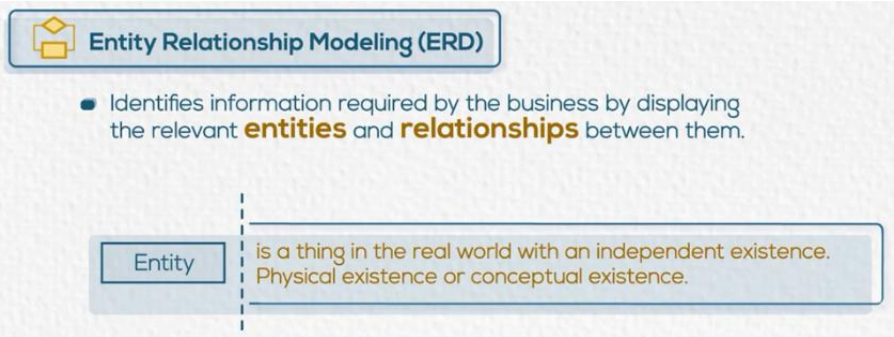


Relational Database:

- It consists of relations or tables of relations.
- Each table consists of columns and tuples (records).
- The intersection between them is domain.
- The domain can only have a single value.
- Each table has a primary key that can be one column or multiple columns together.
- The primary key (underlined): - must contain a unique value for each row of data.
  - cannot contain null values.

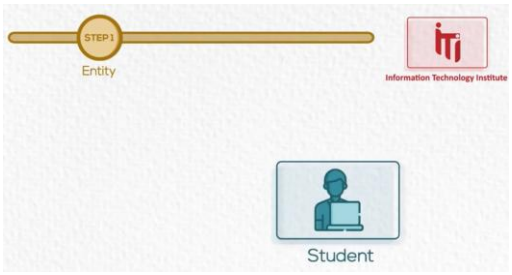
Entity Relationship Diagram (ERD):

- An Entity-Relationship Diagram (ERD) is a visual representation of the structure of a database.
- It's a way to create conceptual/logical schema/design.



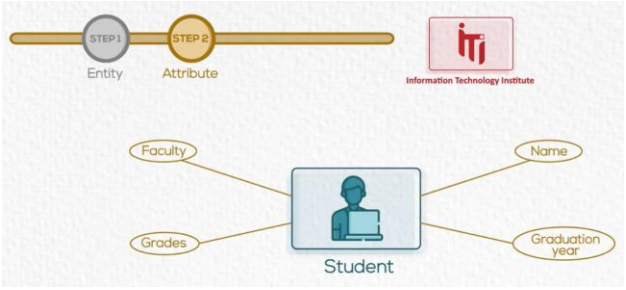
Entity

- An entity represents a real-world object or concept in the database.
- It can be something tangible (like a "Person" or "Car") or abstract (like an "Order" or "Course").
- In an ERD, entities are typically represented by rectangles.
- **Example:** In a university database, entities could be "Student," "Course," and "Professor."



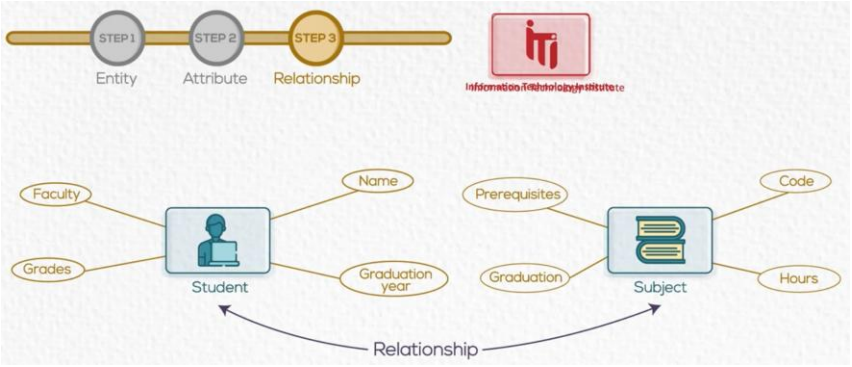
Attribute

- Attributes are the properties or characteristics of an entity.
- Each attribute holds a specific piece of data about the entity.
- In an ERD, attributes are usually represented by ovals connected to their respective entities.
- **Example:** For the "Student" entity, attributes might include "Student ID," "Name," "Date of Birth," and "Email."

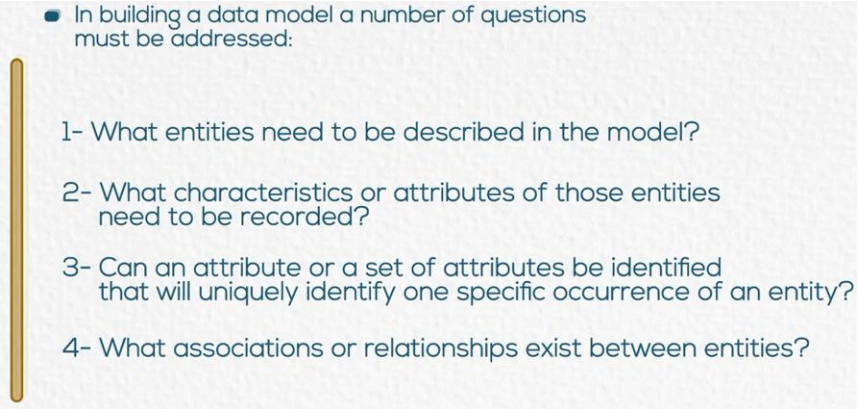


Relationship

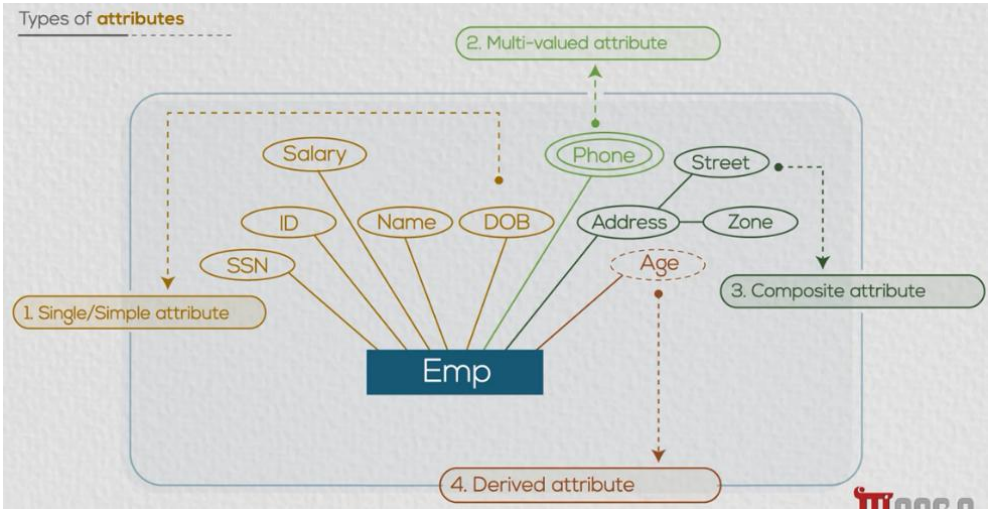
- A relationship describes how two or more entities are related to each other.
- Relationships represent the associations between entities in the database.
- In an ERD, relationships are depicted by diamonds connecting entities.
- **Example:** In a university database, a relationship could be "Enrolls," which links "Student" and "Course," indicating that students enroll in courses.



How to construct an ERD?



Types of Attributes:



- **Simple attributes**: are not divisible and have a single value for a particular entity instance.

- **Multi-valued attributes**: have a set of values for the same entity instance.

- **Composite attributes**: can be divided int subparts.

- **Derived attributes**: can be calculated from another attribute or entity.

- **Candidate (Primary) key**: an attribute that can uniquely identify one specific occurrence of an entity. (Underlined attribute)

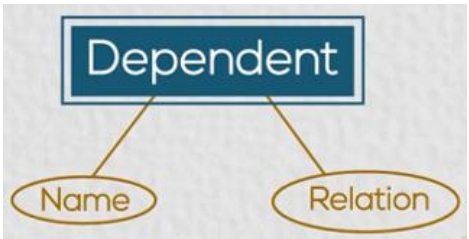
Types of Entities

- **Strong entity**:

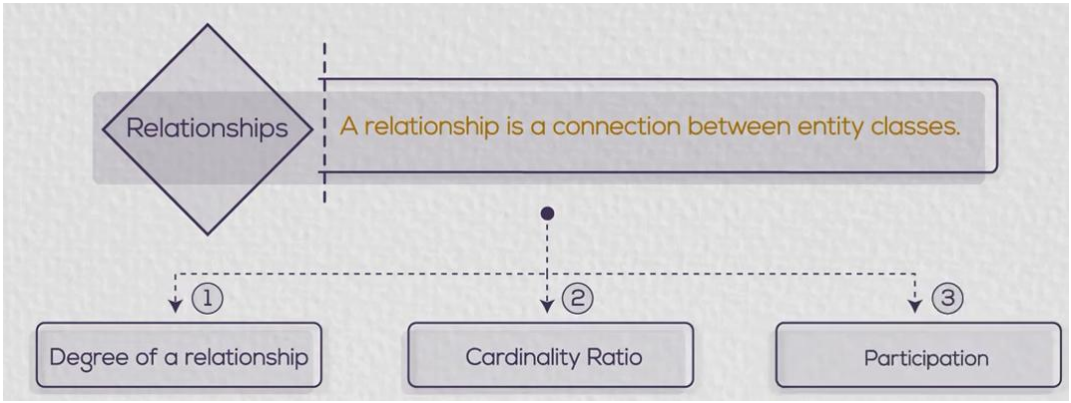
- **Definition:** A strong entity exists independently of other entities and has a primary key that uniquely identifies each instance of the entity.
- **Example:** In a university database, "Student" and "Course" are strong entities because each student and course can be uniquely identified by their "Student ID" and "Course ID," respectively.

- **Weak entity**:

- **Definition:**
    - A weak entity cannot exist without being associated with another entity, known as its "owner" or "parent" entity.
    - It does not have a primary key of its own but relies on a "foreign key" or "partial key" from the owner entity.
- **Example:** "Relative" in a company's medical insurance system might be a weak entity if it is defined only in the context of an "Employee". The "Employee ID" might be used as part of the identifying key for "Relative".



Relationships:



Relationship Degree

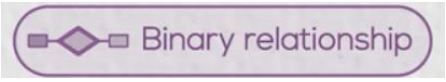
it's the number of entities in a relationship.

Unary Relationship (Degree 1)

- **Definition:** A unary relationship involves only one entity type, where the entity is related to itself.
- **Example:** An "Employee" manages another "Employee". In this case, the relationship "Manages" is unary because it involves the "Employee" entity both as the manager and the subordinate.

Binary Relationship (Degree 2)

- **Definition:** A binary relationship involves two different entity types. It is the most common type of relationship in ERDs.
- **Example:** A "Student" enrolls in a "Course". The relationship "Enrolls" is binary because it involves two entities: "Student" and "Course".



Ternary Relationship (Degree 3)

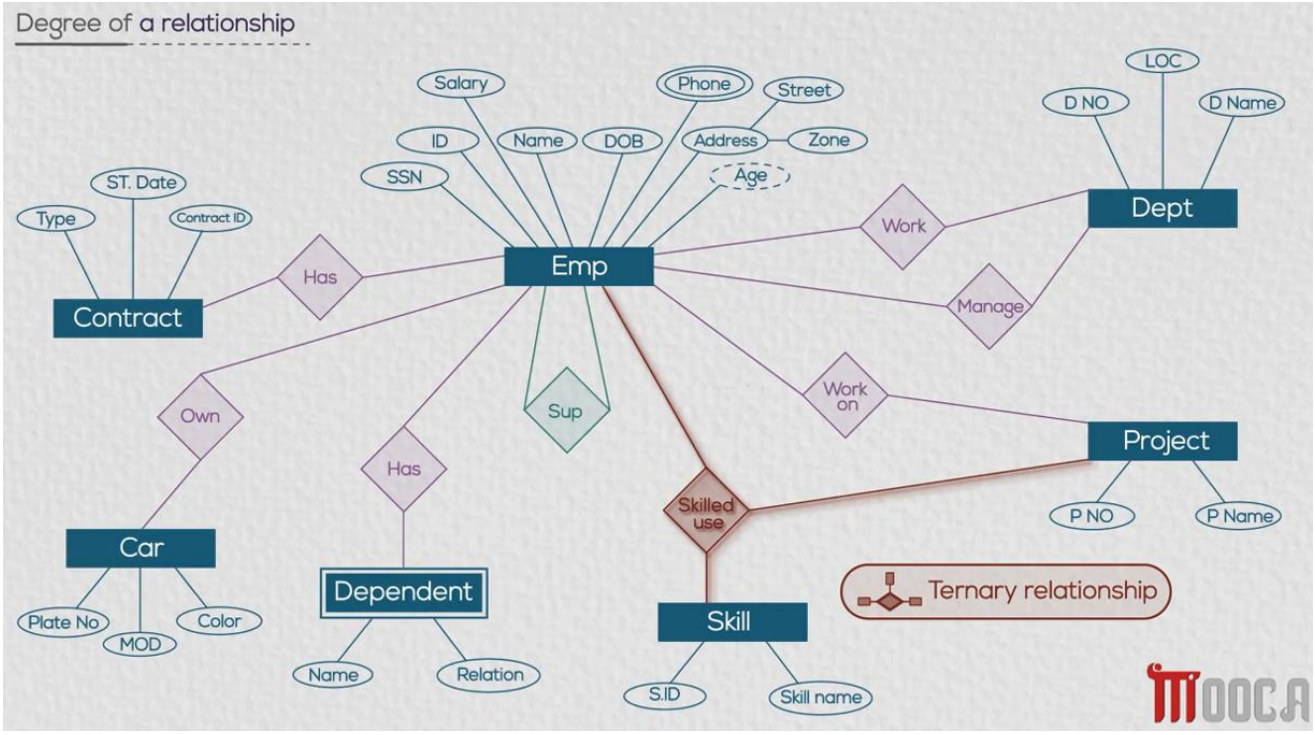
- **Definition:** A ternary relationship involves three different entity types. It represents a relationship that includes three entities simultaneously.
- **Example:** A "Doctor" prescribes a "Medication" to a "Patient". The relationship "Prescribes" is ternary because it involves the entities "Doctor," "Medication," and "Patient".

N-ary Relationship (Degree N)

- **Definition:** An N-ary relationship involves N different entity types, where N is greater than three. This is a more complex relationship that includes multiple entities.
- **Example:** In a supply chain system, a relationship might involve "Supplier," "Product," "Warehouse," and "Transporter," forming a quaternary (4-ary) relationship.

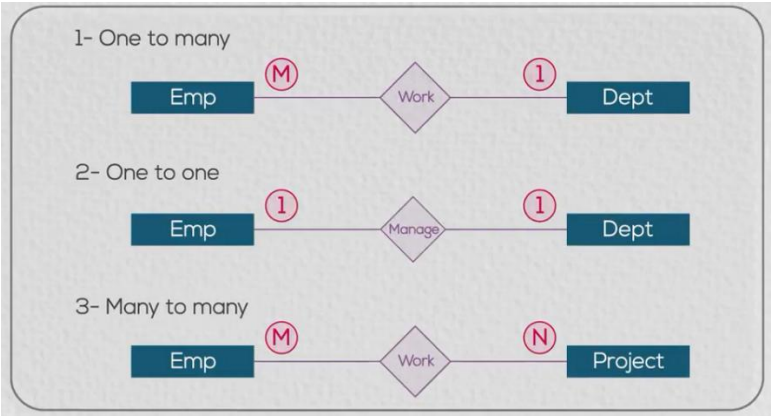
Summary of Relationship Degrees:

- *Unary:* Involves 1 entity (e.g., Employee manages Employee).
- *Binary:* Involves 2 entities (e.g., Student enrolls in Course).
- *Ternary:* Involves 3 entities (e.g., Doctor prescribes Medication to Patient).
- *N-ary:* Involves N entities (e.g., Supplier provides Product to Warehouse through Transporter).



Cardinality Ratio

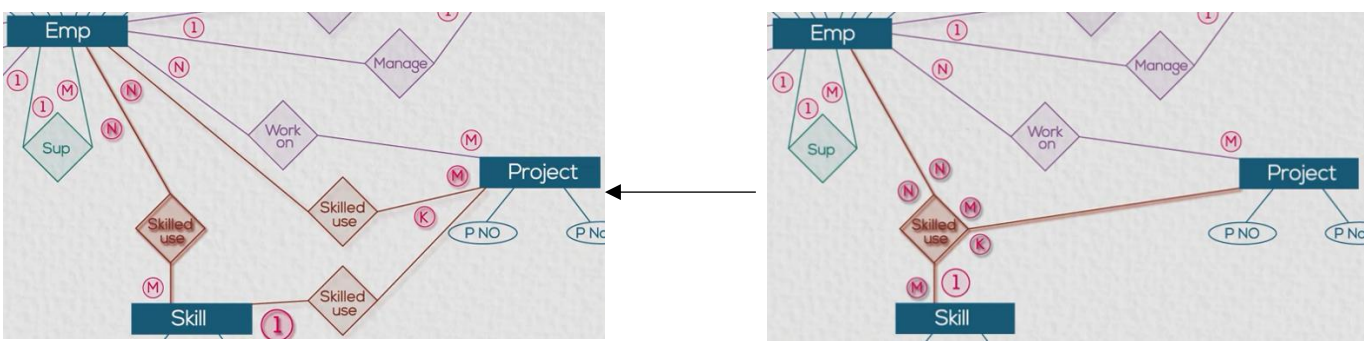
- The cardinality ratio describes the numerical relationship between entities participating in a relationship.
- It defines how many instances of one entity can or must be associated with instances of another entity.

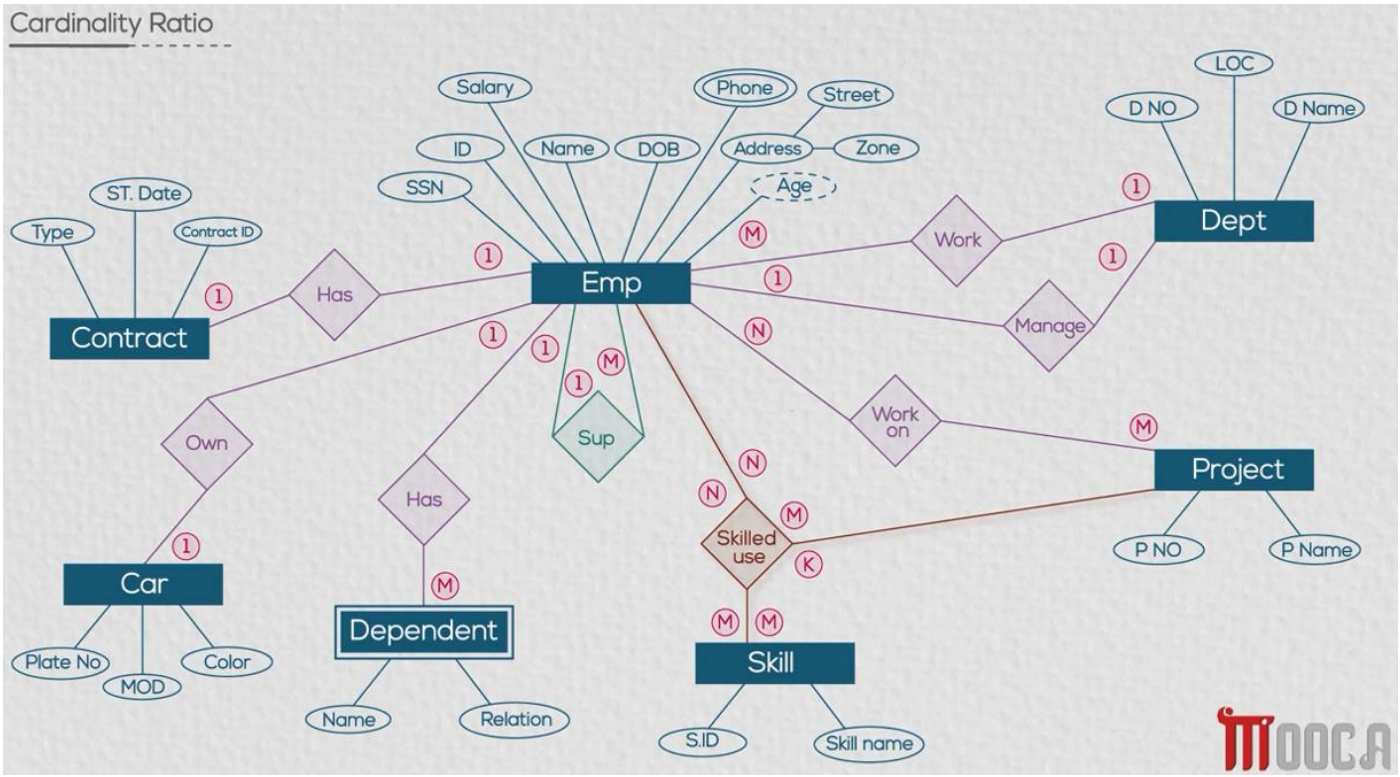


هنا أنا بمسك كل relationship وبحدد الـ max number بتاع كل طرف عن طريق انى بسأل نفسي سؤال في كل اتجاه وبكتب الرقم عند الطرف اللى بينتهى عنده السؤال.

يعنى مثلا الموظف بيشتغل في كام قسم ، قسم واحد ، يبقى هكتب واحد عند القسم ، طب القسم بيشتغل فيه كام موظف ، موظفين كتير ، فهكتب كتير عند الموظف.

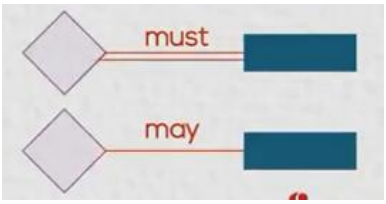
- **<u>ملحوظة مهمة:</u>** لازم الـ cardinality في حالة الـ ternary relationship تتطابق على نفس الـ side ، يعنى مثلا الاتنين يكونوا M ، اما مثلا لو واحدة 1 و واحدة M ساعتها لازم افك الـ ternary واخليها closed loop من الـ binary.





Participation

- Participation refers to the extent to which entities in a relationship must participate in that relationship.
- It indicates whether the presence of a relationship between entities is mandatory (must) or optional (may).
    - **Total Participation:** Every instance of the entity is involved in the relationship (Mandatory participation). Represented by a double line in ERDs.
    - **Partial Participation:** Some instances of the entity may not be involved in the relationship (Optional participation). Represented by a single line in ERDs.



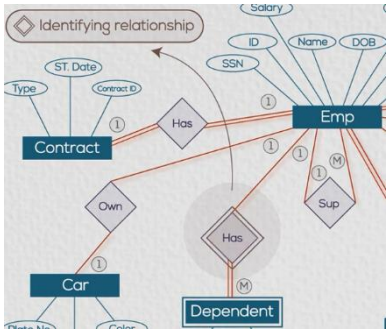
- هنا انا بعبر عن اقل رقم الـ entity ممكن تشارك بيه في الـ relationship instance سواء صفر(may) او واحد(must).

- يعنى مثلا الموظف لازم ع الأقل يشتغل تبع قسم معين يبقى هعمل double line في الـ relation من عند الموظف ، لكن القسم ممكن يكون لسه فاتح ومفيش فيه حد اتعين لسه يبقى هعمل one line من عند القسم.

- الـ participation بين الـ weak entity والـ owner entity بتاعتها لازم يكون must من ناحية الـ weak.

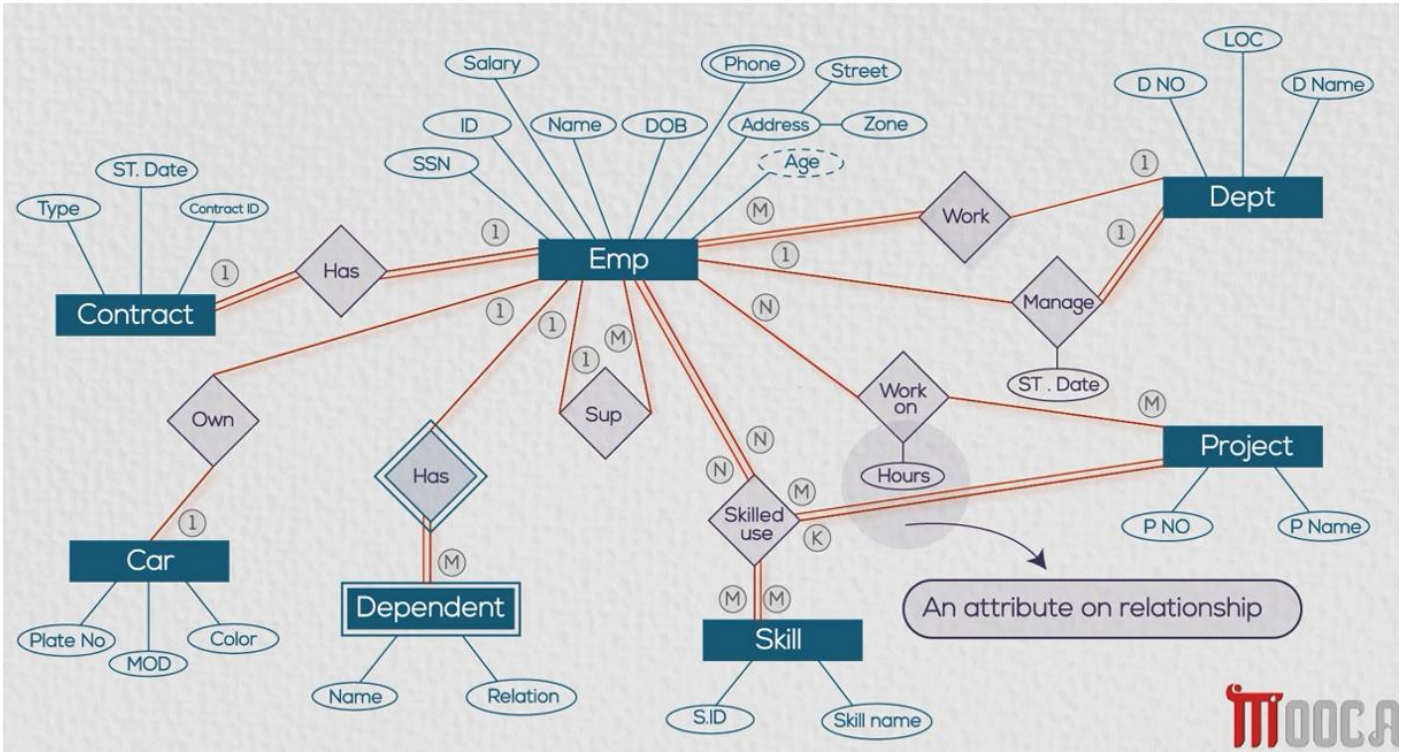
- لو عندى relationship instance واحدة بس بتكسر الـ rule اللى بسأل عنها يبقى هنا الـ participation لازم يكون may ، يعنى مثلا في حالة الـ relationship "supervise" المفروض ان كل الموظفين يكون ليهم مسئول لكن مدير الشركة دة الوحيد اللى مالوش مسئول يبقى العلاقة هنا لازم تكون may ، وطبعا بالنسبة للناحية التانية مش كل الموظفين لازم يكونوا مسئولين عن حد وبالتالي العلاقة هتكون may برضو.

- في الـ participation بتاع الـ ternary لما باجى اسأل السؤال بحاول ادخل الـ 3 entites معايا ، يعنى مثلا هل المشروع لازم يكون فيه موظفين بالمهارات دى ، هل المهارات لازم تكون عند موظفين في مشروع ، هل الموظفين لازم يكون عندهم المهارات دى في المشروع دة.



- الـ relationship اللى بين الـ weak والـ owner بتكون اسمها identifying relationship ، وبميزها انى بخلى شكل الـdiamond بتاع الـrelationship يبقى double-lined.

- ساعات بتكون في معلومة او attribute مقدرش اربطها بـ entity لكن بحطها على الـ relationship عشان تكون مظبوطة ومعبرة عن هدفها ، زى مثلا start date اللى هو هنا بيعبر عن الموظف او المدير بقى بدء يدير الإدارة دى امتى ، لو جيت حطتها كـattribute عند المدير هتبقى بتعبر عن امتى المدير بدء يشتغل عموما ، ولو حطيتها عند الإدارة يبقى معناها الإدارة بدء الشغل فيها امتى ، يبقى لازم احطه عند الـrelationship نفسها.



How to convert conceptual design to logical design:

هنا انا عاوز بقى احول الرسمة بتاعة الERD لlogical design في صورة جداول.

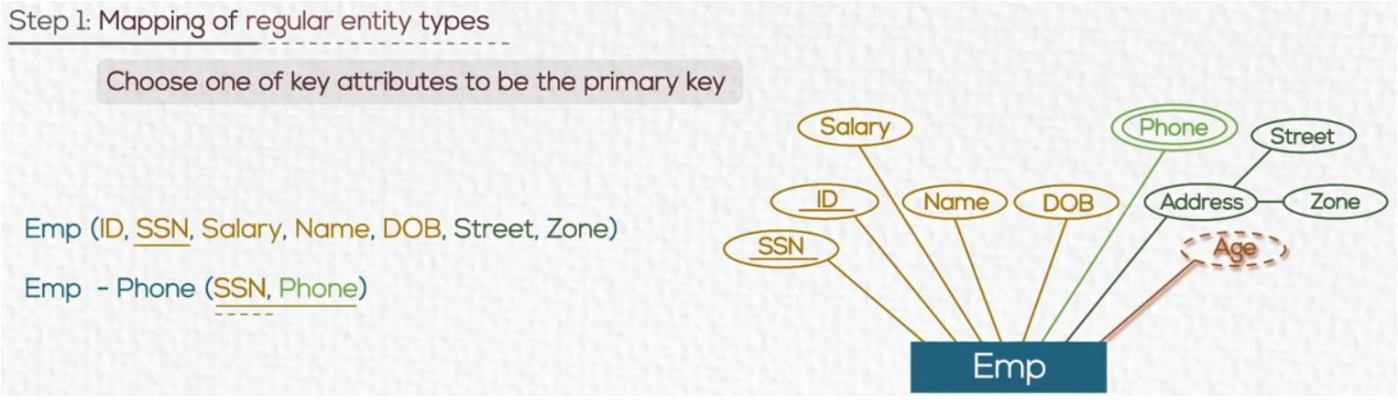
Step 1: Mapping of regular entity types

- Create a table for each type of entity.
- Choose one of key attributes to be the primary key.
- في النقطة اللى فاتت دى لو عندى اكتر من attribute ممكن يكون الprimary key بختار اللى بياخد مساحة اقل في الstorage ، يعنى لو مثلا عندى attribute نوعه text و واحد تانى نوعه number هختار اللى نوعه number.
- Every simple attribute is represented by a single column.
- The subparts of the composite attribute are represented by a single column each.
- The multivalued attribute is represented in a single table contains the multi-valued attribute and foreign key corresponding to the primary key.
  The primary key of this new table is the combination of foreign key and another column.
- في نقطة الmulti-valued attribute مش هينفع اكرره في tuple جديد لوحده لان هيبقى ادامى حاجة من اتنين ، ياما اكرر الprimary key معاه عشان ابقى رابطهم ببعض ودة ضد قواعد الprimary key ، ياما احط بس الmulti-valued attribute وساعتها هسيب باقى الخانات فاضية ودة برضو ضد قواعد الprimary key ، الحل بقى بيكون انى بعمل جدول منفصل بحط فيه الmulti-valued attribute وبحط معاه الprimary key بتاع الجدول الرئيسى as a Foreign key.

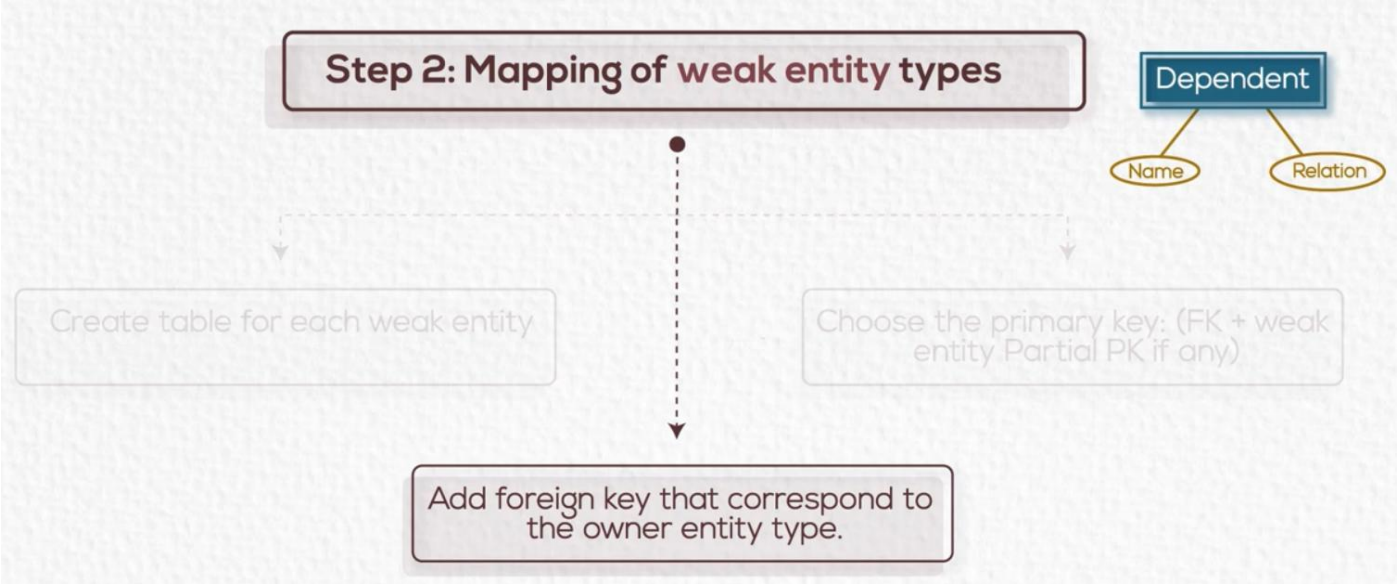
SSN	Name	Phone
123	Ahmed	0245678
		0159875

SSN	Name	Phone
123	Ahmed	0245678
123		0159875

- Foreign Key:
  - A foreign key is a column or set of columns in one table that refers to the primary key in another table.
  - The foreign key creates a link between the data in the two tables, enforcing referential integrity.
  - It ensures that a record in the child table cannot reference a non-existent record in the parent table.
- The derived attribute is not stored in the database by default except when needed.
- يعنى انا مش بخزن الderived attribute لأنى وقت ما احتاجه ممكن استنتجه من attribute تانى ، لكن بخزنه لما بيكون الطلب عليه كبير فبدل ما كل مرة احسبه واعمل headache على الcomputing بخزنه بقى.



Step 2: Mapping of weak entity



- The foreign key here is the primary key of the owner entity.
- The primary key of the weak entity table is the combination of foreign key and a partial key from the table.

Dependent (SSN, Name, Relation)

- The logical design at the end of the first two steps is:

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

Emp - Phone (SSN, Phone)

Dept (DNO, D Name, LOC)

Project (PNO, P Name)

Dependent (SSN, Name, Relation)

Car (Plate_NO, Model, Color)

Contract (Contract_ID, Type, Start_date)

Skill (Skill_id, Skill_name)

Step 3: Mapping of relationship with (Unary - Binary) degree and (1:N) cardinality

- Here, we add the primary key of 1 side as a foreign key to N side.

- طب ليه انا بعمل كدة مش العكس ، لأنى لو خدت الprimary key بتاع الN side وحطيته as foreign key في الside 1 يبقى كدة خانة الSSN في جدول الDept هتبقى multi-valued لأن في موظفين كتير شغالين في القسم الواحد ، لكن العكس هيبقى صحيح ان الموظف الواحد شغال في قسم واحد ، يعنى مثلا في حالة جدول الEmp انا هكرر عامود الSSN وعشان متلخطبش هسميه مثلا Sup-SSN ، وانا هنا مش بكرر الداتا لأن الSSN بتاع الموظف الواحد هيبقى ادامه SSN مختلف بتاع المدير بتاعه.
- في الـ unary بضيف الـ primary تانى عادى لنفس الـ entity وانا هنا مش بعمل تكرار للداتا لانى هنا بعبر عن معلومة جديدة ، وممكن اضيفه باسم متغير عشان ميحصلش لخبطة.
- لو كان الـ primary أصلا موجود في الناحية التانية زى مثلا حالة الweak entity هنا انا مش محتاج اضيفه تانى.

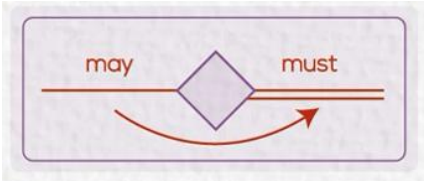
Emp (ID, SSN, Salary, Name, DOB, Street, Zone, DNO, Sup-SSN

Step 4: Mapping of relationship with (Unary - Binary) degree and (M:N) cardinality

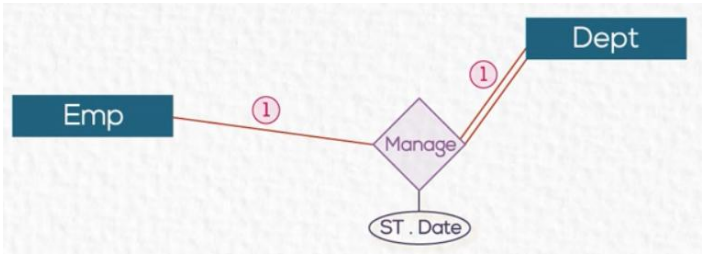
- هنا انا باخد الـ primary keys بتوع الـentities كـ foreign keys في جدول جديد باسم الـ relationship.
- الـprimary key بتاع الجدول الجديد هو الـ combination بتاع الـ foreign keys.
- لو في attribute على الـrelationship دايما بيتبع الـ foreign key ، يعنى اى mapping of relationship بيطلع منه foreign key فانا بشوف الـ foreign key دة راح فين وبحط الـ attribute دى معاه.

Step 5: Mapping of relationship with (Unary - Binary) degree and (1:1) cardinality

- Here, there are three possibilities:
  - **May-must**

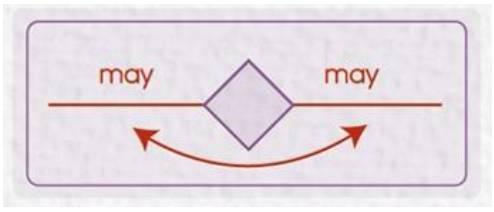


- هنا انا باخد الـprimary key بتاع الـmay وبحطه كـforeign key عند الـmust.
- يعنى مثلا في حالة العلاقة Manage بين الـEmployee والـDept هاخد الـprimary key بتاع المدير اللى هو الـSSN واحطه كـforeign key عند الجدول بتاع الـDept لأن لازم القسم يكون له مدير فدايما هيكون في قيمة للـSSN عند الـDept ، لكن مش كل الموظفين هيكونوا رؤساء اقسام فلو عملت العكس هيكون في خانات فاضية في جدول الـEmployee.
- كان عندى attribute على الـrelationship فعملت زى القاعدة وخليته يتبع الـforeign key وحطيته في جدول الـDept.



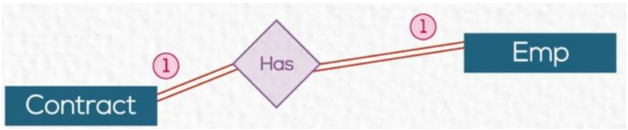
  - **May-may**

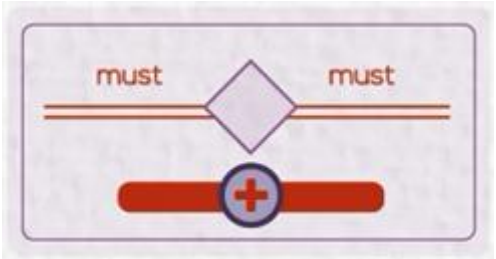
هنا انا ممكن اخد أي primary key واحطه عند التانى كـ foreign key ، او حل تالت انى اعمل جدول جديد بيهم لكن دة مش مفضل.



  - **Mus-must**

هنا انا هبقى مضطر ادمج الجدولين في جدول واحد جديد ، ويكون الـ primary key بتاع الجديد اى واحد من الاتنين القدام ، ولو في المستقبل حبيت اعمل relationship مع الـentity اللى مخترتش الـprimary key بتاعه ممكن برضو استخدم اللى انا اخترته عادى ، وكمان لو عاوز اضيف foreign key في اى جدول من الجدولين القدام ممكن اضيفه في الجديد عادى.





Step 6: Mapping of ternary relationship

هنا مفيش غير انى اعمل جدول جديد وهحط الـ 3 primary keys بتوع الجداول القديمة كـ foreign keys في الجدول الجديد.

Skills Used (SSN, PNO, Skill_id)

The final Logical design is:

Emp- Contract (ID, SSN, Salary, Name, DOB, Street, Zone, DNO, Sup-SSN, Plate_NO, Contract_ID, Type, Start_date

Emp - Phone (SSN, Phone)

Dept (DNO, D Name, LOC, MGR _SSN, ST. Date

Project (PNO, P Name)

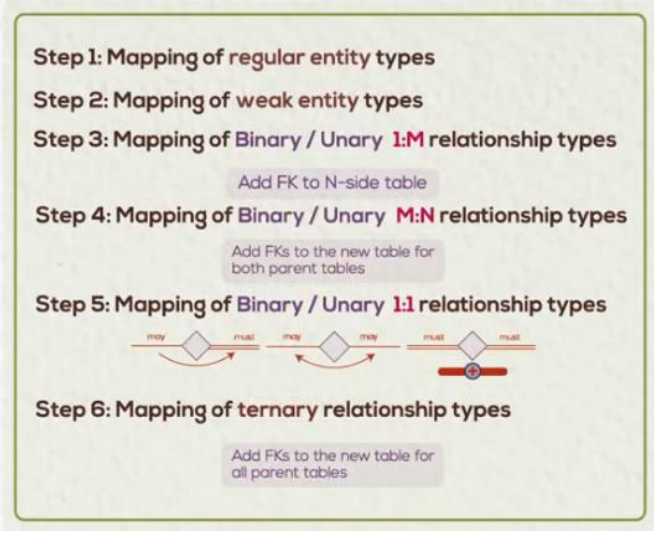
Dependent (SSN, Name, Relation)

Car (Plate_NO, Model, Color

Skill (Skill_id, Skill_name

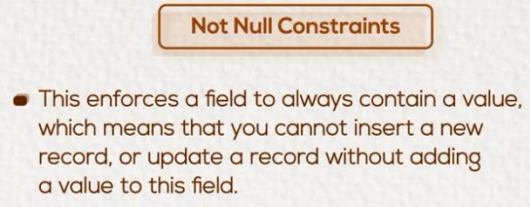
Work_On (SSN, PNO, Hours)

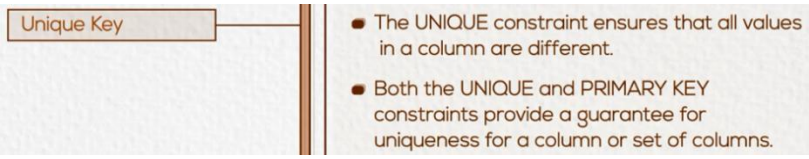
Skills Used (SSN, PNO, Skill_id)

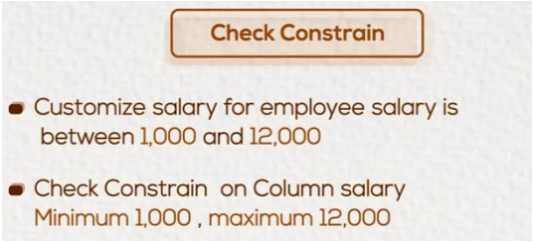


Database constraints

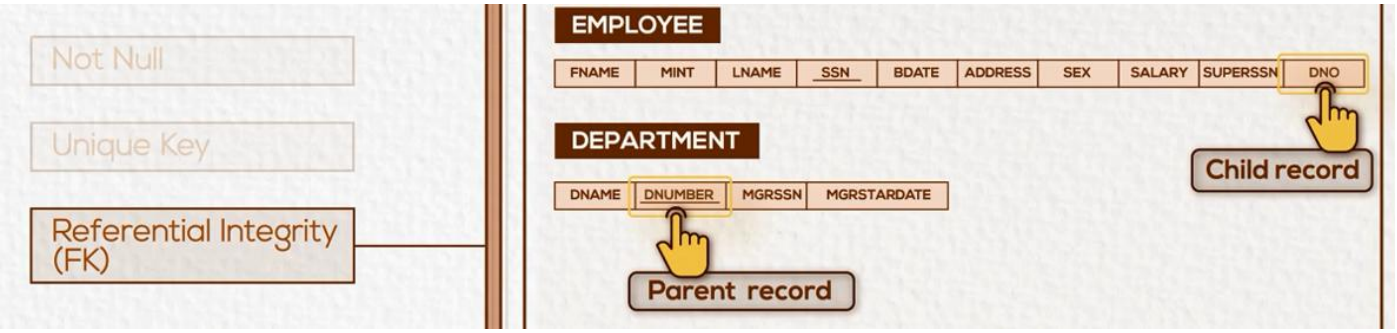
Restrictions on database table or object to help maintain integrity of data.







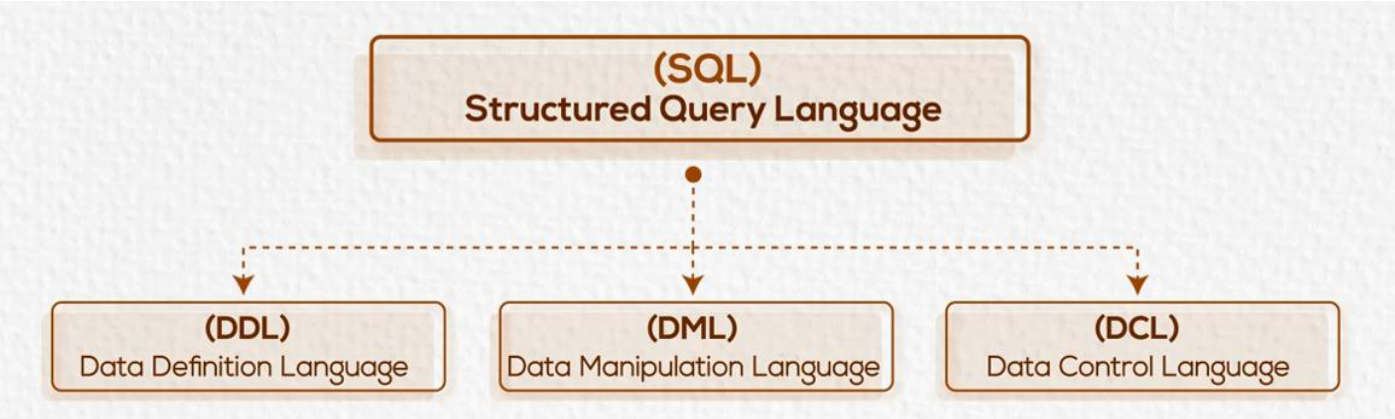
Referential integrity constraint:

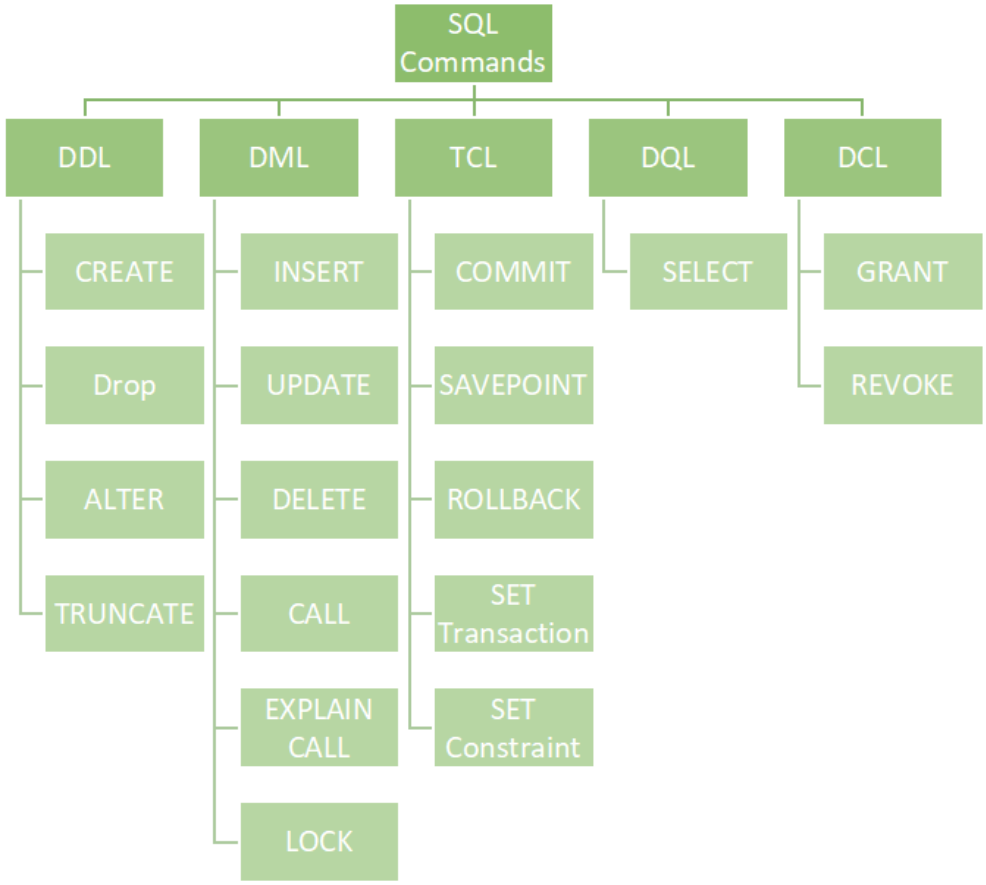


- في حالة لو انا بعمل insert لازم الأول أكون عامل parent record عشان اقدر اضيف child record ، بمعنى انه لازم الأول اضيف قسم ببياناته عشان اقدر لما اجى اضيف بيانات موظف احط رقم القسم دة.

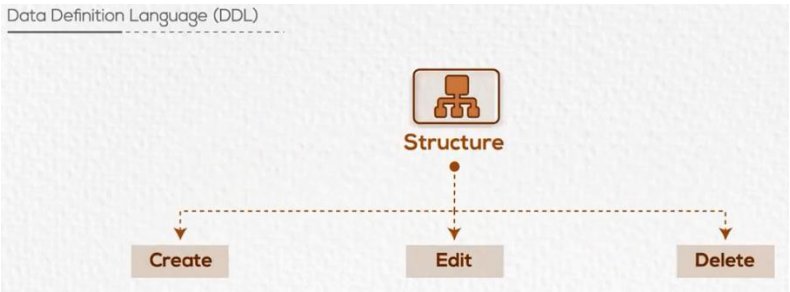
- في حالة لو انا بعمل delete لازم الأول امسح كل الـ child records عشان اقدر امسح الـ parent record ، بمعنى انه لازم اغير كل ارقام اقسام الموظفين اللى شغالين في القسم اللى همسحه قبل ما امسح القسم نفسه.

SQL





Data definition language (DDL)



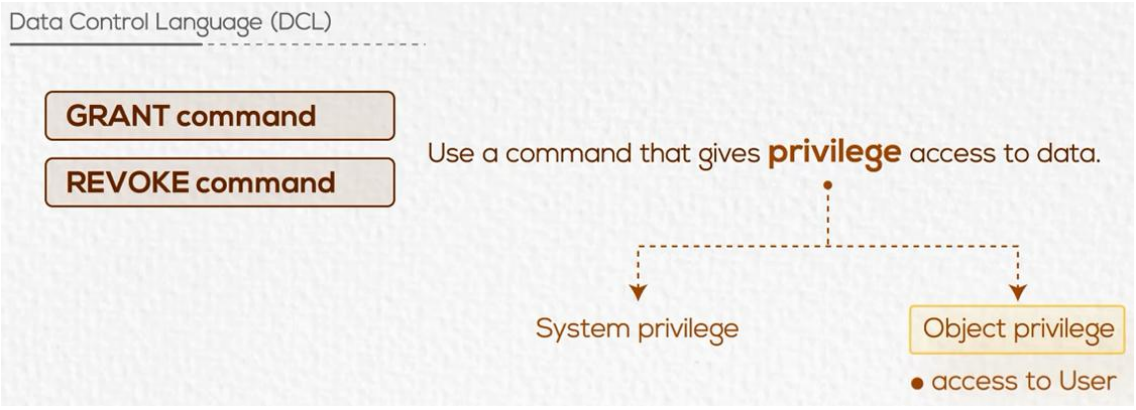
هي عبارة عن مجموعة الـ commands اللى عن طريقها بقدر اعدل في الـ structure بتاع الـ database نفسها لكن مقدرش اعدل في الداتا نفسها.

(CREATE – ALTER – DROP – TRUNCATE - RENAME)

CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers). CREATE DATABASE database_name; CREATE TABLE table_name (column_1 datatype constraint, column_2 datatype constraint, column_3 datatype constraint, ...);	**ALTER:** This command is used to add, delete or change columns in the existing table. The user needs to know the existing table name. ALTER TABLE table_name ADD (Columnname_1 datatype, … Columnname_n datatype); ALTER TABLE table_name DROP COLUMN column_name; ALTER TABLE table_name MODIFY COLUMN column_name datatype;
DROP: This command is used to remove an existing table along with its structure from the Database. DROP TABLE table_name;	**RENAME:** It is possible to change the name of table with or without data in it using simple RENAME command. RENAME TABLE Table_Name To New_Table_Name;
TRUNCATE: This command is used to remove all rows from the table, but the structure of the table still exists. TRUNCATE TABLE table_name;	

Data control language (DCL)

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.



GRANT: This command gives users access privileges to the database. GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER; GRANT SELECT ON TABLE employees TO Ahmed; GRANT ALL ON TABLE department TO Mary, Ahmed; GRANT SELECT ON TABLE employees TO Ahmed WITH GRANT OPTION; اخر واحدة معناها انى لو عملت log in بـ Ahmed ممكن هو كمان يـ share الـ privilege دى مع users تانيين	**REVOKE:** This command withdraws the user's access privileges given by using the GRANT command. REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2; REVOKE ALL on TABLE department FROM Mary;

Data manipulation language (DML)

They are the SQL commands that deal with the manipulation of data present in the database belonging to DML or Data Manipulation Language and this includes most of the SQL statements.

It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

(SELECT – INSERT – DELETE - UPDATE)

INSERT: (works on record level)

It is used to insert data into a table.

عندى 3 طرق ممكن اعمل بيهم الinsert command :

1. بكتب أسماء الـ columns كلها وبعدين بكتب الـ values بتاعتها بالترتيب

```
INSERT INTO Student (Stu_id, Stu_Name, Stu_Marks, Stu_Age) VALUES (104, 'Anmol', 89, 19);
```

2. أو بكتب الـ values علطول لو انا عارف ترتيب الـ columns

```
INSERT INTO Student VALUES (104, 'Anmol', 89, 19);
```

3. أو لو عاوز اضيف قيم معينة بس لازم اكتب الأول أسماء الـ columns بس اللى انا عاوزها

```
INSERT INTO Student (Stu_id, Stu_Age) VALUES (104, 19);
```

UPDATE: (works on column level)

It is used to update existing data within a table.

```
UPDATE Table_name SET [column_name1= value_1, ….., column_nameN = value_N] WHERE CONDITION;

UPDATE Product SET Product_Price = 80 WHERE Product_Id = 'P102' ;

UPDATE Student SET Stu_Marks = 80, Stu_Age = 21 WHERE Stu_Id = 103 AND Stu_Id = 202;
```

جملة where دى اللى بتحددلى الشرط او الـ record اللى انا عاوز اعدل بياناته ، لو مكتبتهاش كدة انا هعدل في الـ column كله مع بعض.

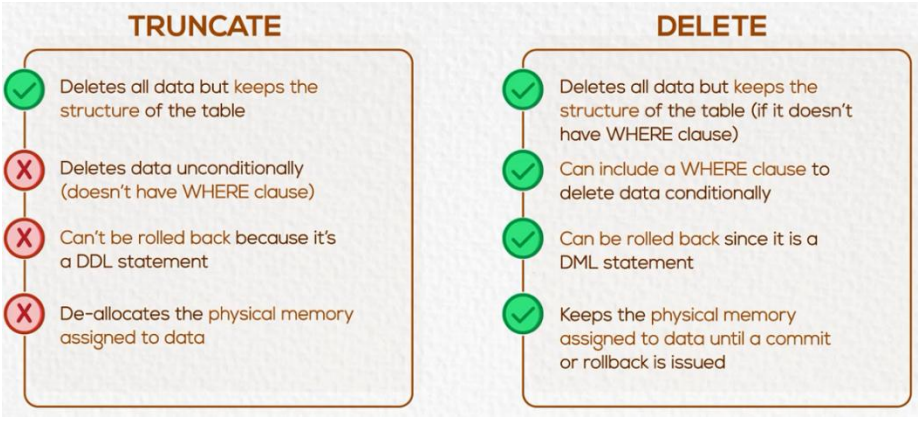
DELETE: (works on table level)

It is used to delete records from a database table.

```
DELETE FROM Table_Name WHERE condition;

DELETE FROM Student WHERE Stu_Marks > 70 ;

DELETE FROM GFG_EMPLOyees;          --------> delete all the records.
```



- طول مانا عامل access على الداتابيز وشغال بعدل في البيانات التعديل دة بيحصل على الميموري بتاعى بس مش بيحصل تعديل على الـphysical memory اللى عليها الداتابيز غير لما بعمل commit.

- لما بعمل commit دة بياخد كل التعديلات اللى عملتها وينقلها على الـphysical memory.

- أوامر الـDDL بتعمل auto-commit اول ما بنفذها.

SELECT

- It is used to retrieve data from the database.
- **Syntax:**

```
SELECT column_Name_1, column_Name_2, …..., column_Name_N FROM Name_of_table;
```

`SELECT * FROM table_name;` ---------> retrieve all columns of table.

`SELECT Emp_Id, Emp_Salary FROM Employee;` ---------> retrieve all the values of a specific column.

`SELECT * FROM Student WHERE Stu_Marks = 80;` ---------> retrieve all the records of a specific condition.

- لو أنا بكتب أسماء الcolumns وفى واحد منهم اسمه فيه مسافة ، بحط اسمه بين [column name] square brackets.

```
SELECT * FROM employees WHERE salary >=2000 AND <= 5000;
```

BETWEEN

```
SELECT FullName FROM employees WHERE salary BETWEEN 1500 AND 2500;
```

IN

```
SELECT * FROM employees WHERE Supervisor IN (16544,12046);
```
---------> equals OR operator.

- عندى هنا الـ IN دة multi-row operator يعنى بعده بقدر احط اكتر من قيمة ، عكس حاجات زى الـ <, > =, دى كلها single-row operator يعنى لازم بعدها اكتب قيمة واحدة بس .

LIKE

```
SELECT * FROM employees WHERE f_name like '?o*';
```

- هنا انا بستعمل like لما أكون عاوز match pattern بدل ما استعمل = لانى ساعتها هكون بدور ع قيمة exact زى اللى بكتبها ، بالنسبة لـ ? فهى بتعبر عن حرف واحد و الـ * بتعبر عن عدد حروف معين من صفر لحد مالانهاية.

ALIAS

```
SELECT f_name+' '+l_name as full_name FROM employees WHERE (salary*12)>10000;
```

- هنا انا معنديش column اسمه f_name+' '+l_name في الجدول فاستعلمت حاجة اسمها **Alias** ، بمعنى انى عملت column جديد باى شكل بقى عمليات حسابية او concatenation زى المثال واديتله اسم عشان الـ DBMS يقدر يعرض الـ column الجديد دة.

ORDER BY

```
SELECT * FROM employees ORDER BY f_name asc, salary desc;
```

- هنا انا بستعمل الـ clause بتاعة order by عشان اعمل ترتيب لقيم الـ column لو هي مش مترتبة مثلا ، فالاول انا بقوله رتب الأسماء تصاعدى وبعديها رتب المرتبات تنازلى ، بمعنى انه هينفذ اول شرط وبعدين لما يلاقى مثلا اتنين اسمهم احمد هيرتبهم بالمرتب بتاعهم تنازلى.

- الـ default بتاع order by انه بيرتب ascending يعنى مش لازم اكتب كلمة asc بعد اسم الـ column.

DISTINCT

```
SELECT DISNTICT dep_no FROM employees;
```

- هنا كلمة distinct بتخليه يعرض الـ column بدون تكرار القيم فيه وبترتبهم تصاعدى كمان.

```
SELECT DISNTICT dep_no, f_name FROM employees;
```

- هنا كلمة distinct بتخليه يعرض الـ column بدون تكرار القيم بتاعة الـ combination للاتنين columns سوى يعنى هي بتعمل distinction للـ result set اللى رجعت.

JOINING

- In SQL, joins are used to combine rows from two or more tables based on a related column.
- There are different types of joins, including INNER JOIN, OUTER JOIN, and EQUI JOIN.
- فكرة الـjoining عموما هو انى عاوز اجيب داتا موجودة في اكتر من جدول مش جدول واحد ، في نفس الوقت بيكون في رابط بين الجداول دى ، اللى هو غالبا بيكون الـprimary key والـforeign key.

1- EQUI JOIN

- An EQUI JOIN is a type of join that uses the equality operator (=) to match rows between tables.
- The result set includes only those rows where the specified columns have matching values.

```
SELECT fname, depName FROM employees, departments WHERE id=mngID;
```

- هنا انا بستعمل حاجة اسمها Equal join ، لانى عاوز اجيب أسماء المديرين وأسماء الأقسام بتاع كل واحد فيهم ، لكن كل column منهم موجود في جدول مختلف وفى نفس الوقت ملهمش علاقة ببعض ، فاستعملت الـ join عشان أقوله اربط كل اسم بقسمه عن طريق ان id يكون هو نفسه mngID.

- عدد الـ join conditions بيكون عدد الجداول ناقص واحد.

- الـ join condition بتكون علاقة بين الـ primary key والـ foreign key بتاعه.

```
SELECT fname, depName FROM employees, departments WHERE employees.dno=departments.dno;
```

هنا انا بعرض أسماء الموظفين وأسماء الإدارات اللى شغالين فيها ، الرابط بينهم هو column له نفس الاسم في الجدولين عشان كدة استعملت (.).

```
SELECT fname, depName FROM employees e, departments as d WHERE e.dno=d.dno;
```

هنا انا عملت alias لاسماء الجداول عشان اسهل على نفسى ، وعملتها بطريقتين ياما بسيب مسافة وبكتب الاسم الجديد او بستعمل as.

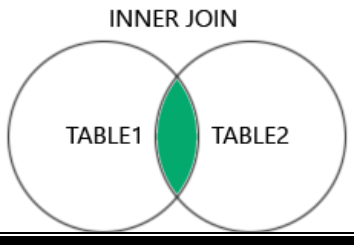
```
SELECT fname, projName, hours FROM employees, project, works_for
WHERE ssn=essn AND pno=pnumber;
```

هنا انا بعرض أسماء الموظفين وأسماء المشاريع اللى شغالين فيها وعدد الساعات اللى اشتغلوها في المشروع ، كدة انا عندى 3 جداول يعنى لازم يكون عندى 2 join conditions ، فدورت على 2 primary keys وربطهم بـ 2 foreign keys من الـ3 جداول.

2- INNER JOIN

هنا بقى انا مش هستعمل الـ = عشان اعمل join ، انا هستعمل keyword مباشرة تعمل عملية الـjoining.

```
SELECT fname, depName
FROM employees e inner join departments as d
ON e.dno=d.dno;
```

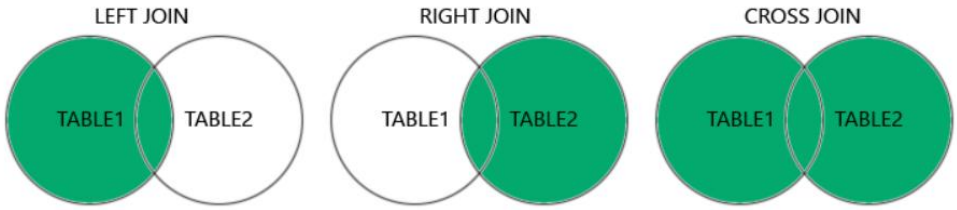


زى الـEQUI JOIN مفيش فرق

3- OUTER JOIN

- An OUTER JOIN returns all rows from one table and the matching rows from the other table.
- If there’s no match, NULL values are returned for columns of the table that lacks a match.

- هنا الouter join بتختلف في انها بتعرضلى كل الداتا حتى لو محصلش matching من الـ condition ، وفى منها 3 أنواع:



A- LEFT OUTER JOIN (or LEFT JOIN):

- Returns all rows from the left table and the matching rows from the right table.
- If there's no match, NULLs are returned for the columns of the right table.

```
SELECT fname, depName
FROM employees e LEFT OUTER JOIN departments d
ON e.dno=d.dno;
```

هنا هو بياخد الـ full data بتاعة الـ column اللى ع الشمال ، يعنى هيجيب كل الموظفين حتى لو مش شغالين في اقسام ، وساعتها هيحط ادام اسم الموظف NULL.

fname	dname
Amr	
Mohamed	
Ahmed	DP1
Kamel	DP1
Hanaa	DP1
Moheb	DP1
Asmaa	DP1
Ahmad	DP1
Noha	DP2
Mariam	DP2
Edward	DP3
Maged	DP3

B- RIGHT OUTER JOIN (or LEFT JOIN):

- Returns all rows from the right table and the matching rows from the right left.
- If there's no match, NULLs are returned for the columns of the left table.

```
SELECT fname, depName
FROM employees e RIGHT OUTER JOIN departments d
ON e.dno=d.dno;
```

هنا هو بياخد الـ full data بتاعة الـ column اللى ع اليمين ، يعنى هيجيب كل الاقسام حتى لو مفيش موظفين فيها ، وساعتها هيحط ادام اسم القسم NULL.

fname	dname
Ahmed	DP1
Kamel	DP1
Hanaa	DP1
Moheb	DP1
Asmaa	DP1
Ahmad	DP1
Noha	DP2
Mariam	DP2
Edward	DP3
Maged	DP3
	DP4

C- FULL OUTER JOIN (or FULL JOIN):

- Returns all rows when there is a match in either table.
- If there is no match, NULL values are returned for the columns of the table that lacks a match.

```
SELECT fname, depName
FROM employees e FULL OUTER JOIN departments d
ON e.dno=d.dno;
```

هنا هو بياخد الـ full data بتاعة الـ 2 columns ، يعنى هيجيب كل الاقسام وكل الموظفين وهيعمل matching للى موجود ، يعنى الأول مثلا هيكتب كل أسماء الموظفين وبعد كل موظف لو موجود له قسم هيكتب اسم القسم لو مش موجود هيرجع NULL ، ونفس الكلام مع الأقسام.

fname	dname
	DP4
Ahmad	DP1
Ahmed	DP1
Amr	
Asmaa	DP1
Edward	DP3
Hanaa	DP1
Kamel	DP1
Maged	DP3
Mariam	DP2
Mohamed	
Moheb	DP1
Noha	DP2

4- Self Join

- A self-join is a type of join in SQL where a table is joined with itself.
- This is useful when you need to:
  - Compare rows within the same table.
    - For example, finding the manager of an employee within the same Employees table.
  - Find relationships between different rows in the same table.
    - For example, comparing records in the same table, like finding pairs of employees with the same salary.
- When you perform a self-join, you treat the table as if it were two separate tables by using table aliases.
- These aliases allow you to distinguish between the two instances of the table in your query.

```
SELECT e.fname, s.fname
FROM employee e, employee s
WHERE e.supID=s.id;
```

- هنا انا بستخدم self join ودة غالبا بتكون مع جدول فيه recursive relationship ، انا عاوز اعرض اسم كل موظف واسم المدير بتاعه ، فظهرت عندى مكلة انى عاوز اعرض نفس الـ column مرتين ، فعملت alias للجدول كأنى عملت نسخة للموظفين ونسخة للمديرين ، وفى شرط الـ join جبت الـ id بتاع المدير بتاع الموظف من جدول الموظف وربطته بالـ id بتاع المدير من جدول المدير.

- هنا انا بحاول اشوف مين الـparent ومين الـchild في النسختين بتوع الجدول اللى بعملهم ، وفى الشرط بتاع الـjoin بجيب الـprimary key بتاع الـparent واساويه بالـforeign key بتاع الـchild ، فمثلا هلاقى ان الـSupervisor هو الـparent لأنه المدير الواحد له اكتر من موظف تحته لكن الموظف له مدير واحد.

- في الغالب دايما بيكون الجدول اللى موجود عندى في الداتا بيز أصلا هو الـchild والجدول النسخة منه اللى بيتعمل جديد بيكون هو الـparent.

Sub-queries

```
SELECT * FROM employees
WHERE salary > ( SELECT salary FROM employees WHERE fname='Ahmed' AND lname='Ali');
```

- هنا انا عاوز اعرض كل الموظفين اللى مرتبهم اكبر من مرتب احمد على ، وعشان المرتب دة معلومة مش عندى جاهزة انا لسه هجيبها من جوة الجدول فاستخدمت sub-queries او nested-queries.

Multi-row Operators: (IN – ALL – ANY)

- In SQL, multi-row operators like ALL and ANY are used in conjunction with subqueries to compare a value to a set of values returned by the subquery.
- These operators allow you to perform comparisons between a single value and multiple values.

1. ALL Operator

- The ALL operator compares a value to every value in a list or subquery.
- The condition must be true for all values in the list for the entire condition to be true.
- **Syntax:**

```
expression operator ALL (subquery)
```

- **Where:**
  - `expression:` The value or column you're comparing.
  - `operator:` A comparison operator, such as =, >, <, >=, <=, <>.
  - `subquery:` A subquery that returns a set of values.
- **Example:** Find employees who have a salary greater than all salaries in the Sales department.

```
SELECT *
FROM Employees
WHERE Salary > ALL (SELECT Salary FROM Employees WHERE Department = 'Sales');
```

2. ANY Operator

- The ANY operator compares a value to each value in a list or subquery.
- The condition must be true for at least one value in the list for the condition to be true.
- **Syntax:**

```
expression operator ANY (subquery)
```

- **Where:**
  - `expression:` The value or column you're comparing.
  - `operator:` A comparison operator, such as =, >, <, >=, <=, <>.
  - `subquery:` A subquery that returns a set of values.
- **Example:** Find employees who have a salary greater than any salary in the Sales department.

```
SELECT *
FROM Employees
WHERE Salary > ANY (SELECT Salary FROM Employees WHERE Department = 'Sales');
```

Example to Illustrate Both:

Consider a scenario where you want to find employees whose salary is higher than the highest salary in the Sales department (ALL) and whose salary is higher than at least one salary in the HR department (ANY):

```
SELECT *
FROM Employees
WHERE Salary > ALL (SELECT Salary FROM Employees WHERE Department = 'Sales') AND
      Salary > ANY (SELECT Salary FROM Employees WHERE Department = 'HR');
```

SQL Aggregate Functions: (MAX() – MIN() – AVG() – COUNT() - SUM())

- SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.

COUNT()

- COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.

- COUNT function uses the COUNT(*) that returns the count of all the rows in a specified table. COUNT(*) considers duplicate and Null.

- Count(salary): Return number of Non-null values over the column salary.
- Count(Distinct Salary): Return number of distinct Non-Null values over the column.

SUM()

- Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

- sum(salary): Sum all Non-Null values of Column salary.
- sum(Distinct salary): Sum of all distinct Non-Null values.

AVG()

- The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-null values.

MAX()

- MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

- Max(salary): Maximum value in the salary column except NULL.

MIN()

- MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

- Min(salary): Minimum value in the salary column except NULL.

```
-- Find the maximum order amount
SELECT MAX(TotalAmount) AS MaxOrder
FROM Orders;

-- Find the minimum order amount
SELECT MIN(TotalAmount) AS MinOrder
FROM Orders;

-- Find the average order amount
SELECT AVG(TotalAmount) AS AvgOrder
FROM Orders;
```

```
-- Count the total number of orders
SELECT COUNT(*) AS NumberOfOrders
FROM Orders;

-- Find the total sum of all orders
SELECT SUM(TotalAmount) AS TotalSales
FROM Orders;
```

- بما ان الaggregate functions بترجع قيمة واحدة بس يبقى انا مينفعش اعرض جنبها في الSelect اى column تانى لأن مش هينفع احط قيمة واحدة جنب كذا قيمة ، بالتالى انا مجبر انى اعمل Group by الcolumn التانى اللى عاوز اعرضه دة ، زى الquery اللى جاية دى :

```
SELECT Min(Salary), Dept_ID
FROM Employee
GROUP BY Dept_ID
```

هنا انا جمعت كل مجموعة موظفين عن طريق القسم بتاعهم وبجيب اقل مرتب جوة كل قسم.

GROUP BY & HAVING

- The GROUP BY and HAVING clauses in SQL are used together to group rows that share certain properties and filter the results based on aggregate functions. They are often used to organize and summarize data.
- The GROUP BY clause is used to group rows that have the same values in specified columns into summary rows.
- The HAVING clause is used to filter the results after the GROUP BY clause has been applied.
- It's similar to the WHERE clause, but WHERE filters rows before grouping, while HAVING filters groups after aggregation.
- The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.
- In the query, the GROUP BY clause is placed after the WHERE clause.
- In the query, the GROUP BY clause is placed before the ORDER BY clause if used.
- In the query, the Group BY clause is placed before the Having clause.
- Place condition in the having clause.

- مينفعش استخدم WHERE مع Aggregate Functions لأن الـ WHERE بتشتغل على الColumn كله يعنى بتمشى على كل صف وتطبق الCondition لكن الAggregate Function بترجع one value فعشان كدة بستخدم معاها HAVING.
- بمعنى انى مينفعش اكتب حاجة زى كدة WHERE SUM(Salary) < 1000 لأن هنا الWHERE بتروح تنفذ الCondition على كل row فمش منطقى انى اعمل Aggregate function على row واحد المفروض انها بتتعمل على Column كامل.
- بس دة برضو مش معناه انى مينفعش استعملها خالص مع الaggregate functions في اى query ، يعنى مثلا في المثال اللى جاى دة:

```
SELECT COUNT(Eid), Address
FROM Emplyee
WHERE Did IN (10,30)
GROUP BY Address
```

Eid	Ename	Salary	Address	did
1	ahmed	3000	cairo	10
2	ali	5000	cairo	10
3	eman	2000	cairo	10
4	khalid	1000	alex	10
5	yousef	4000	alex	10
6	sameh	5000	alex	10
7	mohamed	6000	alex	20
8	alaa	7000	alex	20
9	ola	4000	cairo	20
10	reem	2000	cairo	20
11	nada	9000	cairo	20
12	sayed	8000	mansoura	30
13	reham	1500	mansoura	30
14	sally	2000	mansoura	30
15	~~omar~~	3000	mansoura	30
NULL				

هنا الWHERE بتشتغل على الcolumn قبل ما يتبعت للGROUP BY ، يعنى كل الناس اللى في قسم رقم 20 هيتشالوا من الresult set وبعد كدة الGROUP BY هتجمع القسمين 10 و 30 وبعدين الCOUNT تبدء تعد جوة كل GROUP.

```
SELECT AVG(salary) FROM employees GROUP BY depNum HAVING MAX(salary)>10000;
```

- هنا هو بيعرض متوسط المرتبات بتاع الموظفين بتوع كل قسم لوحده بشرط ان يكون اكبر مرتب في القسم اكبر من 10000.

Example:

Imagine you have an Orders table with columns OrderID, CustomerID, OrderDate, and TotalAmount. You want to find customers who have made more than 5 orders and spent more than $10,000 in total.

```
SELECT CustomerID, COUNT(OrderID) AS NumberOfOrders, SUM(TotalAmount) AS TotalSpent
FROM Orders
GROUP BY CustomerID
HAVING COUNT(OrderID) > 5 AND SUM(TotalAmount) > 10000;
```

Explanation:

- `GROUP BY CustomerID:` Groups the rows by each CustomerID.
- `COUNT(OrderID):` Counts the number of orders for each customer.
- `SUM(TotalAmount):` Calculates the total amount spent by each customer.
- `HAVING:` Filters out customers who haven't made more than 5 orders or spent more than $10,000.

REGEXP:

- In MySQL, REGEXP is a pattern-matching operator that allows you to search for strings that match a regular expression pattern.
- It's similar to the `LIKE` operator but offers more flexibility with complex patterns.
- Regular expressions (regex) are a powerful way to define search patterns for text.
- **Syntax:**

```
SELECT column_name
FROM table_name
WHERE column_name REGEXP 'pattern';
```

Dot (.)

- Matches any single character.

`SELECT 'abc' REGEXP 'a.c';` -- Returns 1 (true), matches "a" followed by any character, then "c"

Caret (^)

- Matches the start of a string.

`SELECT 'abc' REGEXP '^a';` -- Returns 1 (true), matches strings starting with "a"

Dollar ($)

- Matches the end of a string.

`SELECT 'abc' REGEXP 'c$';` -- Returns 1 (true), matches strings ending with "c"

Asterisk (*)

- Matches zero or more occurrences of the previous character or group.

`SELECT 'aaab' REGEXP 'a*b';` -- Returns 1 (true), matches zero or more "a"s followed by "b"

Plus (+)

- Matches one or more occurrences of the previous character or group.

`SELECT 'aaab' REGEXP 'a+b';` -- Returns 1 (true), matches one or more "a"s followed by "b"

Question Mark (?)

- Matches zero or one occurrence of the previous character or group.

`SELECT 'ab' REGEXP 'a?b';` -- Returns 1 (true), matches zero or one "a" followed by "b"

Character Class ([])

- Matches any single character within the brackets.

`SELECT 'abc' REGEXP '[ab]';` -- Returns 1 (true), matches either "a" or "b"

Negated Character Class ([^])

- Matches any single character not within the brackets.

`SELECT 'abc' REGEXP '[^ab]';` -- Returns 1 (true), matches any character except "a" or "b"

Execution sequence of SELECT:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Order	Clause	Function	Notes
1	FROM and JOIN	Tables are joined to get the base data.	The **FROM** clause, and subsequent **JOINs** are first executed to determine the total working set of data that is being queried. This includes subqueries in this clause and can cause temporary tables to be created under the hood containing all the columns and rows of the tables being joined.
2	WHERE	The base data is filtered.	Once we have the total working set of data, the first-pass **WHERE** constraints are applied to the individual rows, and rows that do not satisfy the constraint are discarded. Each of the constraints can only access columns directly from the tables requested in the **FROM** clause. Aliases in the **SELECT** part of the query are not accessible in most databases since they may include expressions dependent on parts of the query that have not yet executed.
3	GROUP BY	The filtered base data is grouped.	The remaining rows after the **WHERE** constraints are applied are then grouped based on common values in the column specified in the **GROUP BY** clause. As a result of the grouping, there will only be as many rows as there are unique values in that column. Implicitly, this means that you should only need to use this when you have aggregate functions in your query.
4	HAVING	The grouped base data is filtered.	If the query has a **GROUP BY** clause, then the constraints in the **HAVING** clause are then applied to the grouped rows, discard the grouped rows that don't satisfy the constraint. Like the **WHERE** clause, aliases are also not accessible from this step in most databases.
5	SELECT	The final data is returned.	Any expressions in the **SELECT** part of the query are finally computed.
6	OREDER BY	The final data is sorted.	If an order is specified by the **ORDER BY** clause, the rows are then sorted by the specified data in either ascending or descending order. Since all the expressions in the **SELECT** part of the query have been computed, you can reference aliases in this clause.
7	LIMIT / OFFSET	The returned data is limited to row count.	Finally, the rows that fall outside the range specified by the **LIMIT** and **OFFSET** are discarded, leaving the final set of rows to be returned from the query.

- عشان اقدر استعمل اى حاجة جوة الـ SELECT أو الـ ORDER BY لازم تكون معمول ليها mention الأول جوة GROUP BY.

Example (1):

Display the department name and the maximum salary for each department given that its average salary is greater than 1200, then sort the result by department name.

```
SELECT dname, MAX(salary) as max
FROM employee e, departments d
WHERE e.dno=d.dno
GROUP BY dname
HAVING AVG(salary) > 1200
ORDER BY dname
```

ترتيب تنفيذ الquery:

1- اول حاجة بتتنفذ هي الfrom ودة كأني بروح اجيب الجداول اللى قولت عليها واسجلهم في الميموري.

Fname	Lname	SSN	Bdate	Address	Sex	Salary	Superssn	Dno
Asmaa	Ali	102661	18-Oct-85		F	1200		10
Moheb	Rafaat	102674	06-May-84	6 Makram Ebid St	M	1700	112233	10
Ahmad	Sameh	102680	12-Dec-86	75 sabaa st	M	1900	102674	10
Ahmed	Ali	112233	01-Jan-65	15 Ali fahmy St.Giza	M	1300	223344	10
Hanaa	Sobhy	123456	18-Mar-73	38 Abdel Khalik Tharwat St. Downtown.Cairo	F	800	223344	10
Kamel	Mohamed	223344	15-Oct-70	38 Mohy el dien abo el Ezz St.Cairo	M	1800	321654	10
Mohamed	Hamed	223366	25-Aug-80	8 Lebanon st	M			
Amr	Omran	321654	14-Sep-63	44 Heliopolis.Cairo	M	2500		
Edward	Hanna	512463	19-Aug-72	18 Abaas El 3akaad St. Nasr City.Cairo	M	1500	321654	30
Maged	Raoof	521634	04-Jun-80	18 Kholosi st.Shobra.Cairo	M	1000	968574	30
Mariam	Adel	669955	06-Dec-82	269 El-Haram st. Giza	F	750	512463	20
Noha	Mohamed	968574	02-Jan-75	55 Orabi St. El Mohandiseen .Cairo	F	1600	321654	20

Dname	Dno	MGRSSN	MGRStart Date
DP1	10	223344	01-Jan-05
DP2	20	968574	03-Jan-06
DP3	30	512463	06-Jan-06
DP4	40		

2- تاني حاجة بتتنفذ هي الwhere ، يعني بروح اشوف ايه الconditions اللى كاتبها ادامها وبنفذها ، هنا مثلا انا بعمل EQUI JOIN للجدولين فكأنهم اتحطوا جوة جدول واحد ، وبما اني عملت join وكان في موظفين في جدول الemployee مفيش ادامهم ارقام إدارة ف اتشالوا من الجدول الجديد.

Fname	Lname	SSN	Bdate	Address	Sex	Salary	Superssn	Dno	Dno	Dname	MGRSSN	MGRStart Date
Asmaa	Ali	102661	18-Oct-85		F	1200		10	10	DP1	223344	01-Jan-05
Moheb	Rafaat	102674	06-May-84	6 Makram Ebid St	M	1700	112233	10	10	DP1	223344	01-Jan-05
Ahmad	Sameh	102680	12-Dec-86	75 sabaa st	M	1900	102674	10	10	DP1	223344	01-Jan-05
Ahmed	Ali	112233	01-Jan-65	15 Ali fahmy St.Giza	M	1300	223344	10	10	DP1	223344	01-Jan-05
Hanaa	Sobhy	123456	18-Mar-73	38 Abdel Khalik Tharwat St. Downtown.Cairo	F	800	223344	10	10	DP1	223344	01-Jan-05
Kamel	Mohamed	223344	15-Oct-70	38 Mohy el dien abo el Ezz St.Cairo	M	1800	321654	10	10	DP1	223344	01-Jan-05
Mohamed	Hamed	223366	25-Aug-80	8 Lebanon st	M							
Amr	Omran	321654	14-Sep-63	44 Heliopolis.Cairo	M	2500						
Edward	Hanna	512463	19-Aug-72	18 Abaas El 3akaad St. Nasr City.Cairo	M	1500	321654	30	30	DP3	512463	06-Jan-06
Maged	Raoof	521634	04-Jun-80	18 Kholosi st.Shobra.Cairo	M	1000	968574	30	30	DP3	512463	06-Jan-06
Mariam	Adel	669955	06-Dec-82	269 El-Haram st. Giza	F	750	512463	20	20	DP2	968574	03-Jan-06
Noha	Mohamed	968574	02-Jan-75	55 Orabi St. El Mohandiseen .Cairo	F	1600	321654	20	20	DP2	968574	03-Jan-06

Fname	Lname	SSN	Bdate	Address	Sex	Salary	Superssn	Dno	Dno	Dname	MGRSSN	MGRStart Date
Asmaa	Ali	102661	18-Oct-85		F	1200		10	10	DP1	223344	01-Jan-05
Moheb	Rafaat	102674	06-May-84	6 Makram Ebid St	M	1700	112233	10	10	DP1	223344	01-Jan-05
Ahmad	Sameh	102680	12-Dec-86	75 sabaa st	M	1900	102674	10	10	DP1	223344	01-Jan-05
Ahmed	Ali	112233	01-Jan-65	15 Ali fahmy St.Giza	M	1300	223344	10	10	DP1	223344	01-Jan-05
Hanaa	Sobhy	123456	18-Mar-73	38 Abdel Khalik Tharwat St. Downtown.Cairo	F	800	223344	10	10	DP1	223344	01-Jan-05
Kamel	Mohamed	223344	15-Oct-70	38 Mohy el dien abo el Ezz St.Cairo	M	1800	321654	10	10	DP1	223344	01-Jan-05
Edward	Hanna	512463	19-Aug-72	18 Abaas El 3akaad St. Nasr City.Cairo	M	1500	321654	30	30	DP3	512463	06-Jan-06
Maged	Raoof	521634	04-Jun-80	18 Kholosi st.Shobra.Cairo	M	1000	968574	30	30	DP3	512463	06-Jan-06
Mariam	Adel	669955	06-Dec-82	269 El-Haram st. Giza	F	750	512463	20	20	DP2	968574	03-Jan-06
Noha	Mohamed	968574	02-Jan-75	55 Orabi St. El Mohandiseen .Cairo	F	1600	321654	20	20	DP2	968574	03-Jan-06

3- بعد كدة بينفذ الgroup by ، يعني بيروح على كل مجموعة records شبه بعض في الشرط اللى انا كاتبه (اللى هو في المثال dname) ويحطهم في جروب لوحدهم وكأنه بيحط عليهم label بالشرط اللى انا عملت ليهم جروب بيه.

Fname	Lname	SSN	Bdate	Address	Sex	Salary	Superssn	Dno	Dno	Dname	MGRSSN	MGRStart Date	
Asmaa	Ali	102661	18-Oct-85		F	1200		10	10	DP1	223344	01-Jan-05	
Moheb	Rafaat	102674	06-May-84	6 Makram Ebid St	M	1700	112233	10	10	DP1	223344	01-Jan-05	
Ahmad	Sameh	102680	12-Dec-86	75 sabaa st	M	1900	102674	10	10	DP1	223344	01-Jan-05	
Ahmed	Ali	112233	01-Jan-65	15 Ali fahmy St.Giz	M	1300	223344	10	10	DP1	223344	01-Jan-05	
Hanaa	Sobhy	123456	18-Mar-73	38 Abdel Khalik Th	F	800	223344	10	10	DP1	223344	01-Jan-05	
Kamel	Mohamed	223344	15-Oct-70	38 Mohy el dien ab	M	1800	321654	10	10	DP1	223344	01-Jan-05	DP1
Edward	Hanna	512463	19-Aug-72	18 Abaas El 3akaac	M	1500	321654	30	30	DP3	512463	06-Jan-06	
Maged	Raoof	521634	04-Jun-80	18 Kholosi st.Shobr	M	1000	968574	30	30	DP3	512463	06-Jan-06	DP3
Mariam	Adel	669955	06-Dec-82	269 El-Haram st. G	F	750	512463	20	20	DP2	968574	03-Jan-06	
Noha	Mohamed	968574	02-Jan-75	55 Orabi St. El Mol	F	1600	321654	20	20	DP2	968574	03-Jan-06	DP2

- ومن اللحظة دى بقى الDBMS معدش شايف داتا جوة الجدول غير اللى انا كنت ذاكرها جوة الGROUP BY بس ، يعني معدش شايف غير الdname.

4- بعد كدة بيروح ينفذ أي aggregate function موجودة عندى ، يعنى هينفذ اللى جوة HAVING واللى جوة SELECT ويروح يحطهم في الlabel بتاع كل جروب.

Fname	Lname	SSN	Bdate	Address	Sex	Salary	Superssn	Dno	Dno	Dname	MGRSSN	MGRStart Date	
Asmaa	Ali	102661	18-Oct-85		F	1200		10	10	DP1	223344	01-Jan-05	
Moheb	Rafaat	102674	06-May-84	6 Makram Ebid St	M	1700	112233	10	10	DP1	223344	01-Jan-05	
Ahmad	Sameh	102680	12-Dec-86	75 sabaa st	M	1900	102674	10	10	DP1	223344	01-Jan-05	DP1, 1900, 1450
Ahmed	Ali	112233	01-Jan-65	15 Ali fahmy St.Giz	M	1300	223344	10	10	DP1	223344	01-Jan-05	
Hanaa	Sobhy	123456	18-Mar-73	38 Abdel Khalik Th	F	800	223344	10	10	DP1	223344	01-Jan-05	
Kamel	Mohamed	223344	15-Oct-70	38 Mohy el dien ab	M	1800	321654	10	10	DP1	223344	01-Jan-05	
Edward	Hanna	512463	19-Aug-72	18 Abaas El 3akaad	M	1500	321654	30	30	DP3	512463	06-Jan-06	DP3, 1500, 1250
Maged	Raoof	521634	04-Jun-80	18 Kholosi st.Shobr	M	1000	968574	30	30	DP3	512463	06-Jan-06	
Mariam	Adel	669955	06-Dec-82	269 El-Haram st. G	F	750	512463	20	20	DP2	968574	03-Jan-06	DP2, 1600, 1175
Noha	Mohamed	968574	02-Jan-75	55 Orabi St. El Mol	F	1600	321654	20	20	DP2	968574	03-Jan-06	

5- بعد كدة هيروح ينفذالشرط بتاع الHAVING نفسها ، فهلاقى ان DP2 مش محقق الشرط فكدة هيتشال من الresult set بتاعتى.

Fname	Lname	SSN	Bdate	Address	Sex	Salary	Superssn	Dno	Dno	Dname	MGRSSN	MGRStart Date	
Asmaa	Ali	102661	18-Oct-85		F	1200		10	10	DP1	223344	01-Jan-05	
Moheb	Rafaat	102674	06-May-84	6 Makram Ebid St	M	1700	112233	10	10	DP1	223344	01-Jan-05	
Ahmad	Sameh	102680	12-Dec-86	75 sabaa st	M	1900	102674	10	10	DP1	223344	01-Jan-05	DP1, 1900, 1450
Ahmed	Ali	112233	01-Jan-65	15 Ali fahmy St.Giz	M	1300	223344	10	10	DP1	223344	01-Jan-05	
Hanaa	Sobhy	123456	18-Mar-73	38 Abdel Khalik Th	F	800	223344	10	10	DP1	223344	01-Jan-05	
Kamel	Mohamed	223344	15-Oct-70	38 Mohy el dien ab	M	1800	321654	10	10	DP1	223344	01-Jan-05	
Edward	Hanna	512463	19-Aug-72	18 Abaas El 3akaad	M	1500	321654	30	30	DP3	512463	06-Jan-06	DP3, 1500, 1250
Maged	Raoof	521634	04-Jun-80	18 Kholosi st.Shobr	M	1000	968574	30	30	DP3	512463	06-Jan-06	
Mariam	Adel	669955	06-Dec-82	269 El-Haram st. G	F	750	512463	20	20	DP2	968574	03-Jan-06	~~DP2, 1600, 1175~~
Noha	Mohamed	968574	02-Jan-75	55 Orabi St. El Mol	F	1600	321654	20	20	DP2	968574	03-Jan-06	

6- بعد كدة هيروح ينفذ الSELECT ، فهيشوف انا طالب منه يعرض ايه ويعمل بيه result set جديدة ، ولازم اللى انا طالبه يعرضه دة يكون موجود جوة الlabel بتاع كل جروب ، يعنى من الاخر لازم أكون كاتبه جوة الGROUP BY او حاسبه بaggregate function ، يعنى لو مثلا عاوز اعرض الdno لازم أكون عامله mention جوة الGROUP BY.

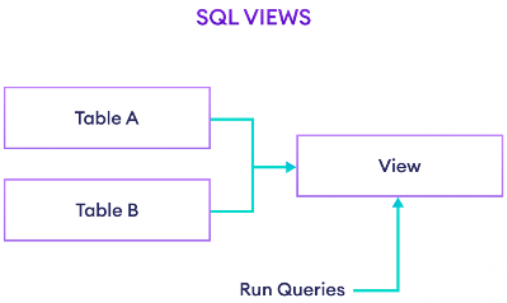
Dname	Max
DP1	1900
DP3	1500

7- اخر حاجة بقى هيروح ينفذ الORDER BY وبرضو لازم يعمل اوردر بحاجة تكون موجودة جوة الlabel حتى لو انا مش عارضها جوة الSELECT ، فهيكون شكل الquery النهائية كدة:

Dname	Max
DP1	1900
DP3	1500

Views

- In SQL, a view is a virtual table (logical table) based on the result-set of an SQL statement (SELECT).
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- A View can either have all the rows of a table or specific rows based on certain conditions.
- You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.
- A view contains no data of its own, but it's like a window through which data from tables can be viewed or changed.
- The tables on which a view is based are called the base tables.
- The views is stored as a SELECT statement in the data dictionary.
- A view always shows up-to-date data! The database engine recreates the view, every time a user queries it.



Views Types

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

- في حالة الComplex Views مش دايما هقدر اعمل DML Operations لأن الView بتاعى بيكون معمول فيه functions و aggregations فدة معناه ان الداتا اللى ظاهرة دى أصلا مش حقيقية او مش زى الداتا اللى في الtables الاصلية.

- هنا انا عملت view with check option ، دة معناه انى لما اجى اعمل اى DML على الview دة الـ DBMS بيتحقق كل مرة من الشرط بتاع الwhere قبل ما يعمله ، وطبعا بما ان الـ view هو نفسه معندوش data of its own فلما اجى اعمل اى DML الـ DBMS بيقوم واخدها ورايح يعملها على الbase tables.

- هنا هو بيشوف الأول لو مفيش view له نفس الاسم بيعمل واحد جديد ، ولو في واحد أصلا بيعدل فيه ، لان مينفعش اتنين views يكون ليهم نفس الاسم.

Advantages of Views:

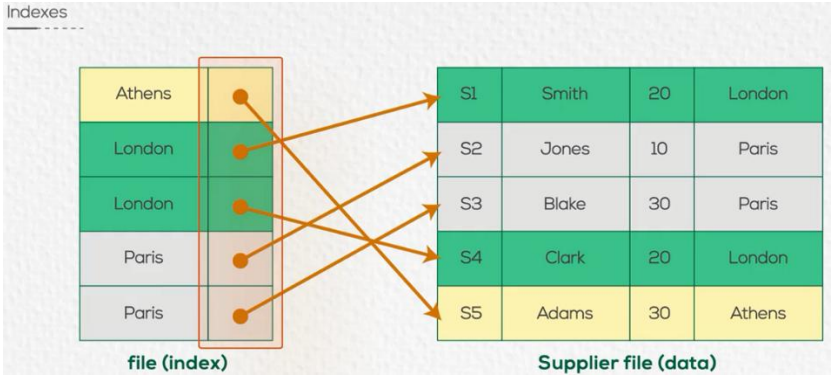
1- **Restricting data access** – By creating views, you can restrict access to specific rows or columns of a table, ensuring that users only see the data they are authorized to access.

ودة لأن الـ DCL بتعمل restrictions على الـ table كله ، لكن انا هنا ممكن اقسم كمان الـ restrictions دى على أجزاء من الـtable.

2- **Simplify commands for the user** – Views can present a simplified, business-focused schema to end-users or applications, hiding the complexity of the underlying database structure.
3- **Store complex queries** – Views allow you to encapsulate complex SQL queries in a simplified and reusable way. Once a view is created, users can query the view as if it were a simple table, without needing to understand or repeatedly write the underlying complex SQL logic.
4- **Reusability Across Applications** – A view can be reused in multiple queries, reports, or applications, ensuring consistency in data retrieval and logic.
   This reduces code duplication and maintenance efforts that If the business logic changes, you only need to update the view definition rather than changing multiple queries scattered across the application.
5- **Consistency Across Changes** – If the underlying data changes, the view dynamically reflects the updated data, ensuring that users always see the most current information.
6- **Multiple view facility** – Different views can be created on the same table for different users.
7- **Data Independence** – Views allow you to present a logical view of the data that is independent of how the data is actually stored in the underlying tables.
   This means that changes to the physical schema (e.g., adding or removing columns, splitting or merging tables) can be made without requiring changes to the queries that users or applications use, as long as the view definitions are appropriately adjusted.

Index

- An index is a schema object.
- It is used by the server to speed up the retrieval of rows by using a pointer.
- It can reduce disk I/O(input/output) by using a rapid path access method to locate data quickly.
- An index helps to speed up select queries and where clauses, but it slows down data input, with the update and the insert statements (DMLs).
- Indexes can be created or dropped with no effect on the data.
- Indexes is used to solve stored data problems:
    - **Not sorted**
    - **Scattered:** data are not stored in consecutive spaces on physical memory.



- الـ index هنا بيسرع الأداء ، فمثلا لو انا عاوز ادور على الـ records بتاعة London الـ DBMS بيمشى على الـ index من الأول لحد ما يلاقى كلمة London وبعدين يفضل ماشى لحد ما كلمة London تتغير كدة هو عرف ان دى كل الـrecords بتاعة London فهو مش محتاج يكمل خلاص ، عكس الطريقة العادية لأنى ساعتها بعمل full-table scan لحد ما الاقى كل الـ records اللى عاوزها.

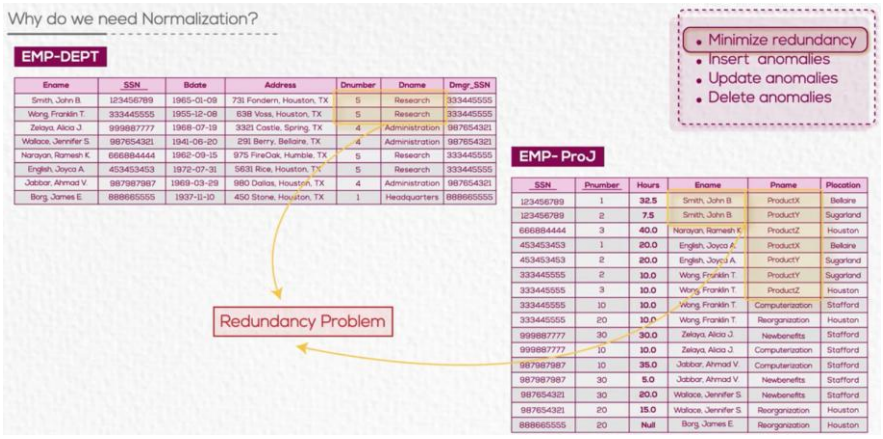
- لكن في عيب برضو عند الـindex ، وهو انه بيبطئ أوامر الـDML شوية لانه ساعتها بيشتغل على حاجتين ، الـrecords اللى موجودة في الجدول الاصلى والـindex نفسه ، يعنى لو انا مثلا بعمل update لـrecord فانا هعدله في الـtable الأول وبعدين في الـindex وبعدين اروح اعمل sorting للـindex عشان يبقى ماشى مع المعلومات الجديدة ، فدة كله بيعمل slow down للـperformance.

Normalization of data:

- Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization rules divide larger tables into smaller tables and link them using relationships.
- The purpose of Normalization is to eliminate redundant (repetitive) data and ensure data is stored logically.

Data Redundancy

- Redundant data means storing the same piece of data in multiple places.
- This not only wastes storage space but also creates potential issues with data consistency.
- If data is duplicated and one instance is updated while others are not, it leads to inconsistency.
- Normalization helps reduce redundancy by breaking down data into separate tables, ensuring that each piece of information is stored only once.



- مشكلة مثلا زى اللى في الصورة ، لو جيت بصيت في جدول الEMP-DEPT هلاقى ان لو مثلا عندى 100 موظف شغالين في قسم رقم 5 ، كدة انا هبقى محتاج انى اكتب اسم القسم ورقم مدير القسم 100 مرة كمان.

Data Modification Anomalies

- Data modification anomalies refer to potential problems that can occur when inserting, updating, or deleting data in a database that is not properly normalized.

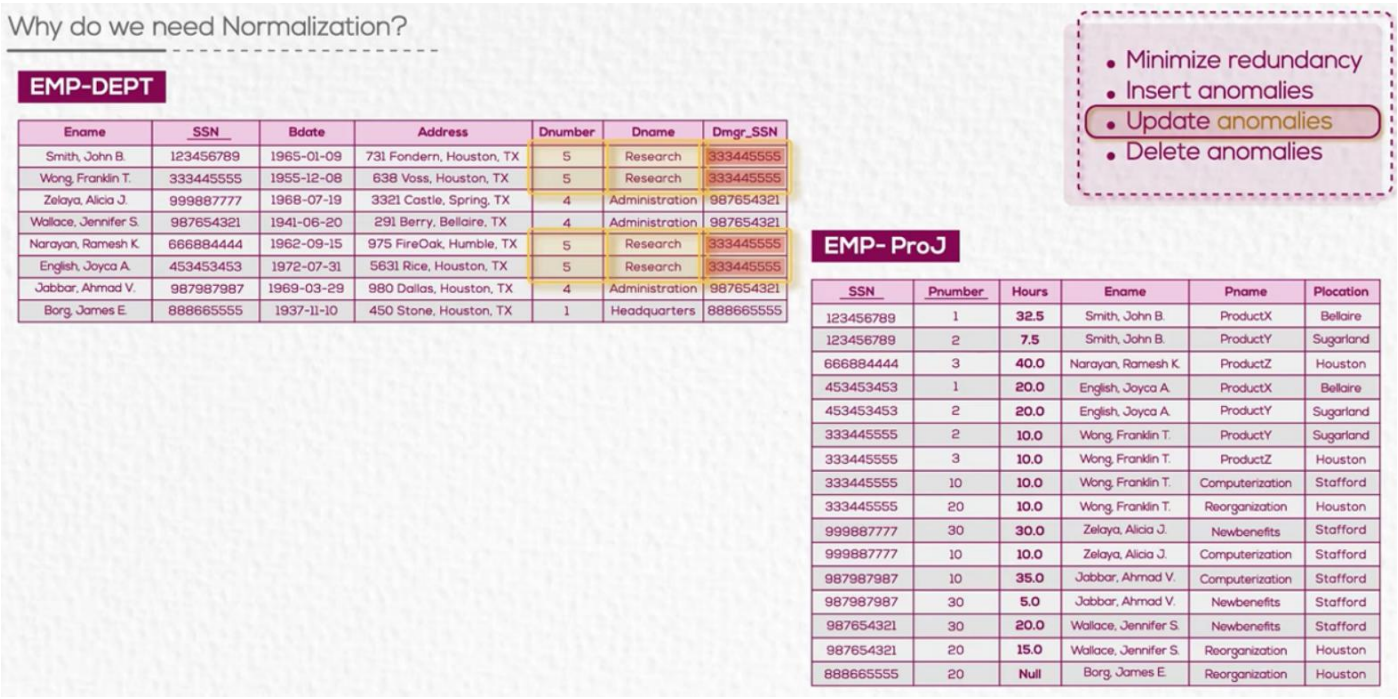
Insertion Anomaly

- An insert anomaly occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

- يعنى مثلا لو بصيت على جدول الEMP-DEPT هلاقى ان الSSN دة الprimary key بتاع الجدول ، يعنى مش هقدر اضيف اى بيانات عن قسم معين غير لما يكون في موظف متعين فيه.

Update Anomaly

- An update anomaly occurs when data is duplicated in multiple rows, and a change in one instance requires the same change to be made in all instances.
- Failure to update all copies leads to inconsistencies.
- يعنى لو مثلا عاوز اعدل رقم رئيس قسم معين (Dmgr_SSN) ، لو القسم دة فيه 100 موظف فانا كدة عاوز اعدل المعلومة دى في الـ 100 row بتوع موظفين القسم دة.

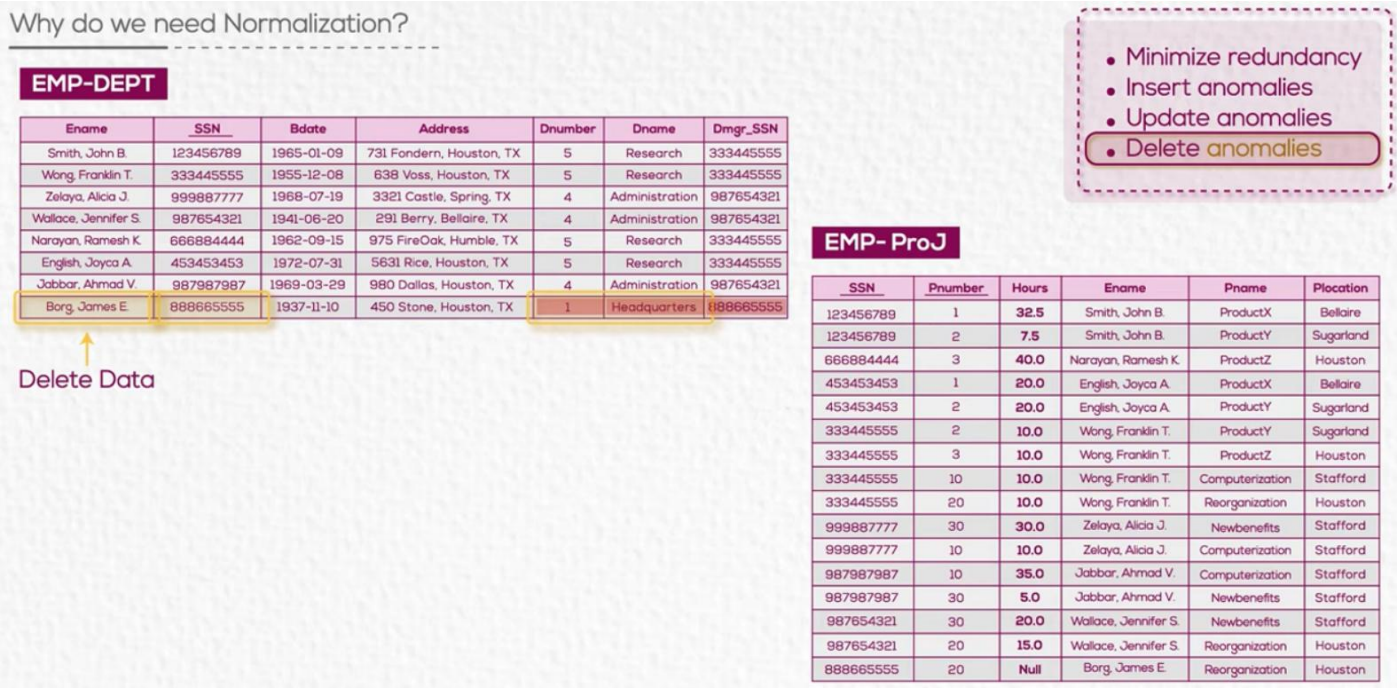


Ename	SSN	Bdate	Address	Dnumber	Dname	Dmgr_SSN
Smith, John B.	123456789	1965-01-09	731 Fondern, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

SSN	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

Deletion Anomaly

- A delete anomaly occurs when the deletion of data representing one fact results in the unintended loss of other data.
- يعنى مثلا عندى القسم رقم 1 دة شغال فيه موظف واحد بس ، لو جيت مسحت الrecord بتاع الموظف دة كدة انا مسحت بيانات القسم كمان ، يعنى يعتبر القسم قفل ومعدش موجود عندى ، فكدة وجود القسم معتمد على وجود الموظف وهيعملى مشكلة في الdeletion بتاع الموظف.

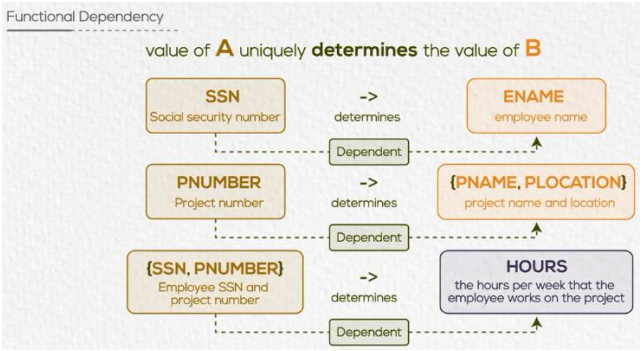


Ename	SSN	Bdate	Address	Dnumber	Dname	Dmgr_SSN
Smith, John B.	123456789	1965-01-09	731 Fondern, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

SSN	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

Functional Dependency

- Functional Dependency is a constraint between two columns or two sets of columns.
- Functional Dependency occurs when the value of one attribute (or a set of attributes) uniquely determines the value of another attribute (or set of attributes).
- If an attribute A functionally determines B, then it is written as A ⟶ B.
- **For Example:**
  - employee_id ⟶ name, employee_id functionally determines the name of the employee.
  - {student_id, time} ⟶ {lecture_room}, student ID and time determine the lecture room where the student should be.



Full Functional Dependency

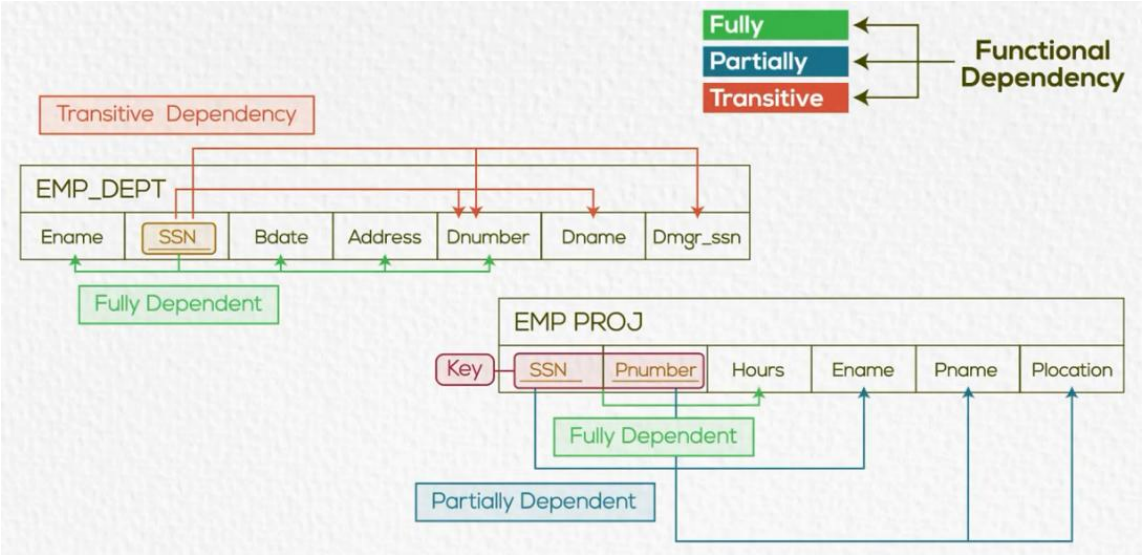
- A functional dependency X ⟶Y is full if Y is functionally dependent on X and not on any proper subset of X.
- او بمعنى تانى Non-key attribute is fully dependent on the key ، زى مثلا وجود الBdate معتمد اعتماد كلى على وجود الSSN ، او وجود الHours معتمد اعتماد كلى على {SSN,Pnumber}.

Partial Functional Dependency

- A functional dependency X ⟶Y is partial if Y is functionally dependent on a part of the composite key X.
- او بمعنى تانى عندى Non-key attribute معتمد على جزء من الComposite key ، زى مثلا الPlocation معتمد على Pnumber بس اللى هو جزء من الComposite Key {SSN,Pnumber}.

Transitive Functional Dependency

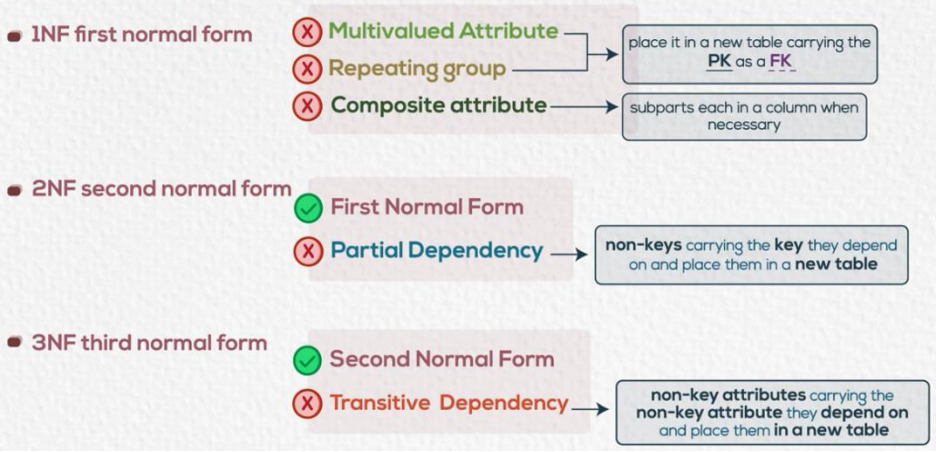
- A functional dependency is transitive if there is an indirect relationship between attributes.
- This occurs when X ⟶Y and Y ⟶Z, then X ⟶Z is a transitive dependency.
- او بمعنى تانى عندى Non-key attribute معتمد على Non-key attribute معتمد على الKey ، زى مثلا الDname معتمد على الDnumber اللى معتمد على الSSN.



Normal Forms

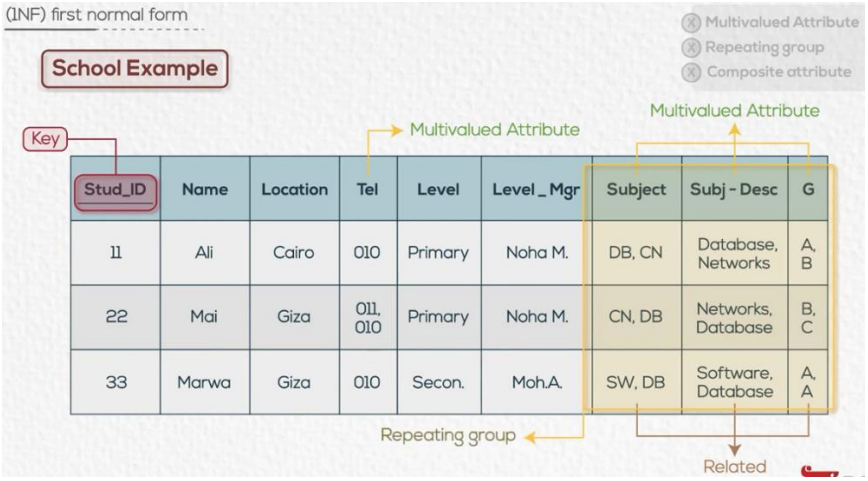
- Normalization is achieved through a series of steps called normal forms.
- Normal forms are guidelines used to structure a database in such a way that it reduces redundancy and ensures data integrity.
- Each normal form has specific rules that must be followed.
- Each normal form builds upon the previous one (Sequential).

- However, higher levels of normalization can lead to more complex database designs and queries. It is important to strike a balance between normalization and practicality when designing a database.



First Normal Form (1NF)

- A table will be 1NF if it contains only atomic (indivisible) values.
- Ensures that the table structure is free of repeating groups, multi-valued attributes and composite attributes.



Stud_ID	Name	Location	Tel	Level	Level_Mgr	Subject	Subj - Desc	G
11	Ali	Cairo	010	Primary	Noha M.	DB, CN	Database, Networks	A, B
22	Mai	Giza	011, 010	Primary	Noha M.	CN, DB	Networks, Database	B, C
33	Marwa	Giza	010	Secon.	Moh.A.	SW, DB	Software, Database	A, A

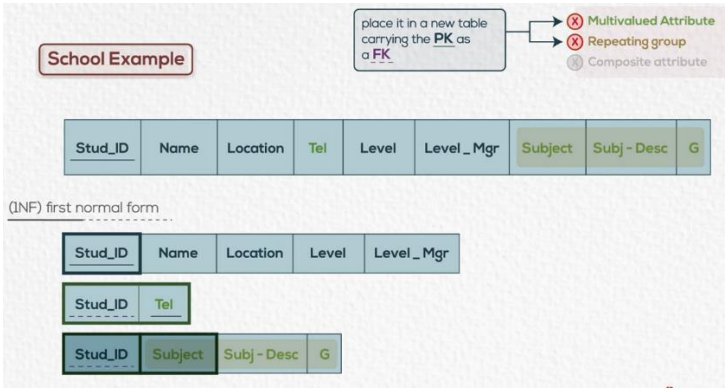
- عشان أقول أن الـ relation بتاعتى في الـ 1NF لازم ميكونش فيها (Multivalued attribute – Repeating group – Composite Attribute).

- في المثال بتاعى أنا عندى الـ Tel دة multi-valued attribute ، والـ (Subj , Subj-Desc , G) دول كل واحدة فيهم لوحدها Multi-valued attribute لكن الـ3 سوى فيه علاقة بينهم اذا هما repeating group.

- يبقى علشان اخليهم على الـ 1NF لازم اكسر الـrelation بتاعتى لـ separate relations.

Repeating Group:

- A repeating group occurs when a database table has a field (or set of fields) that stores multiple related values for a single entity or record.
- This is often seen when a table includes fields that hold lists or arrays of values, rather than representing each value as a separate record in a related table.

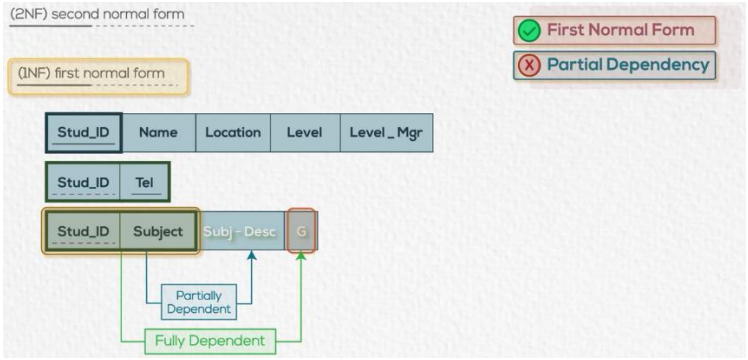


- الـ Stud_ID و Tel دول مع بعض composite primary key ، والـ Stud_ID و Subject دول مع بعض composite primary key.

- لو كان عندى composite attribute بقسمه لـseparate attributes within the same table .

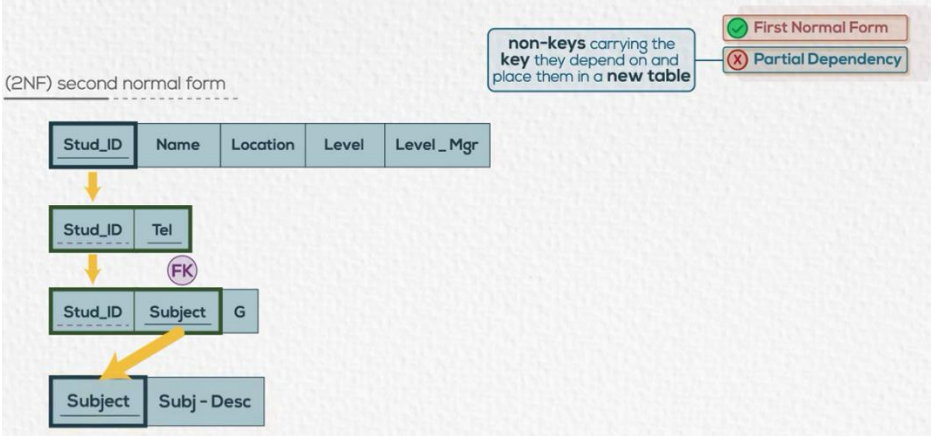
Second Normal Form (2NF)

- A table will be 2NF if:
  - The table is 1NF.
  - All non-key attributes are fully functional dependent on the primary key (No Partial Dependency).



- عشان أقول أن الـ relation بتاعتى في الـ 2NF لازم ميكونش فيها Partial Dependency وطبعا تكون على الـ 1NF.

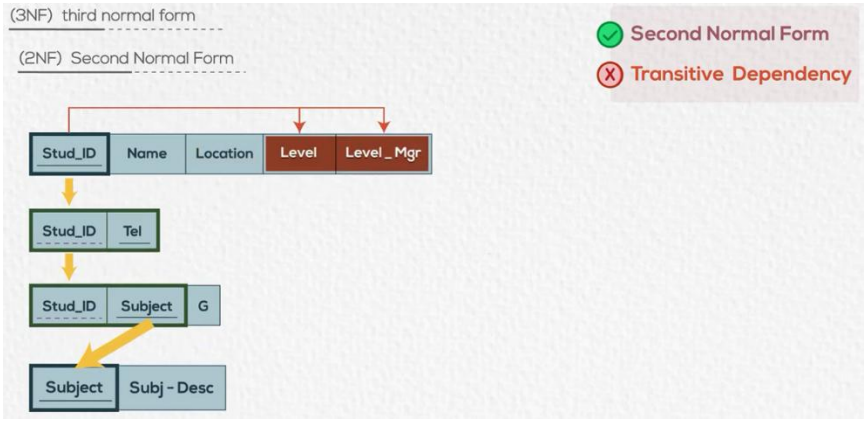
- اول جدول الـ key بتاعه single فمفيش كدة كدة partial dependency ، الجدول التانى الـkey بتاعه composite لكن معنديش أصلا non-key attribute يبقى برضو مفيش مشكلة ، الجدول التالت بقى عندى اتنين لكن بما أن الـG معتمد على الـcomposite key على بعضه اذا هو fully dependent ، يبقى معنديش partial dependency غير الـ Subj-Desc.



- الحل هنا بيكون انى باخد الattribute دة مع جزء الcomposite key اللى بيعتمد عليه واعملهم في جدول لوحدهم ، ساعتها هيكون جزء الcomposite key هو الـ primary key وبيكون برضو foreign key في الجدول اللى فات عشان كل الجداول تبقى مربوطة ببعض ولأن الـ G كمان معتمدة على الcomposite key على بعضه فمينفعش افصله.

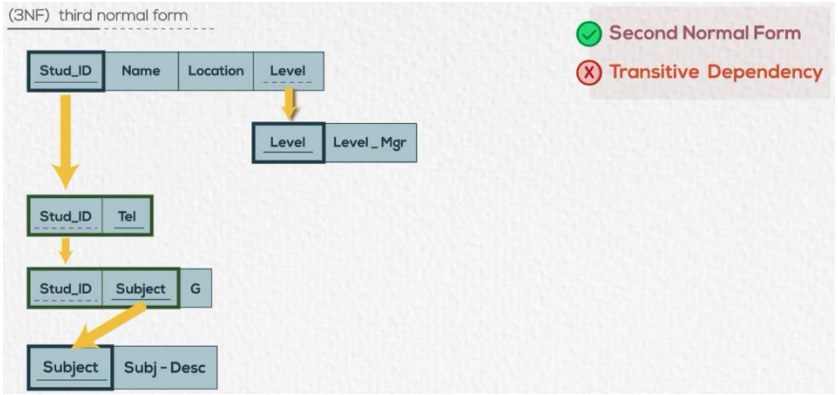
Third Normal Form (3NF)

- A table will be 3NF if:
  - The table is 2NF.
  - There are no transitive dependencies, meaning non-key attributes do not depend on other non-key attributes.



- عشان أقول أن الـ relation بتاعتى في الـ 3NF لازم ميكونش فيها Transitive Dependency وطبعا تكون على الـ 2NF.

- هنا انا عندى الـ Level_Mgr بيعتمد على الـ level نفسه ، والـ level بيعتمد على الprimary key اللى هو Stud_ID اذا انا عندى Transitive dependency.



- الحل هنا بيكون انى باخد الattribute دة مع الattribute اللى بيعتمد عليه واعملهم في جدول لوحدهم ، ساعتها هيكون الattribute هو الـ primary key وبيكون برضو foreign key في الجدول اللى فات عشان كل الجداول تبقى مربوطة ببعض ولأن الـ level معلومة مهمة عن الطالب فلازم يفضل في جدول الطالب.

- في 4NF و 5NF وهكذا وكل ما بزود كل ما بكسر الrelations وازود عدد الجداول لكن دة مش recommended اوى لان انا كدة بأثر على الperfomance بس برضو دة كله بيتحدد على طبيعة استخدامي للداتا وعلى حسب الbusiness case بتاعتى.

Example:

Given the student sheet example provided, design a relational database schema that captures all the information in the sheet. The database should be normalized to at least the Third Normal Form (3NF) to eliminate redundancy and ensure data integrity.

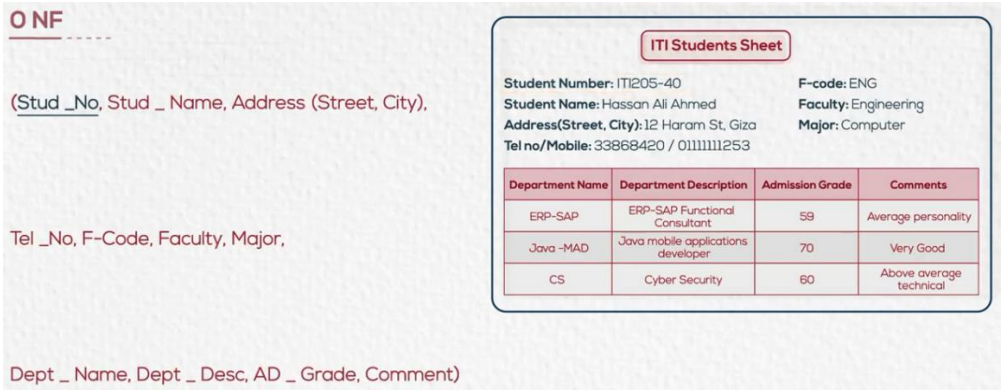
ITI Students Sheet

Student Number: ITI205-40
Student Name: Hassan Ali Ahmed
Address(Street, City): 12 Haram St, Giza
Tel no/Mobile: 33868420 / 01111111253
F-code: ENG
Faculty: Engineering
Major: Computer

Department Name	Department Description	Admission Grade	Comments
ERP-SAP	ERP-SAP Functional Consultant	59	Average personality
Java -MAD	Java mobile applications developer	70	Very Good
CS	Cyber Security	60	Above average technical

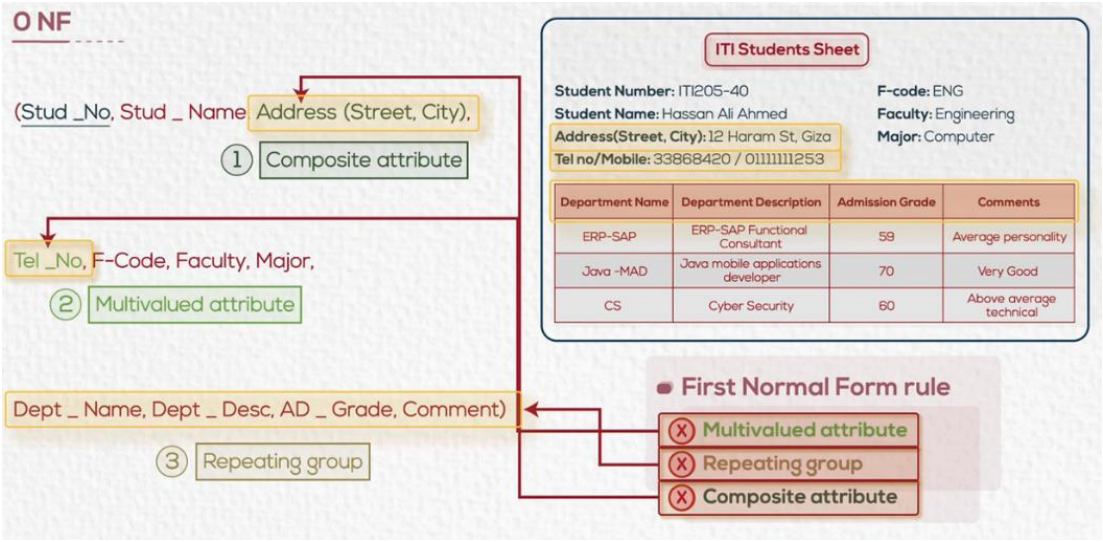
Solution Steps:

1- Turn the report (customer sheet) into 0NF (All attributes in one table):



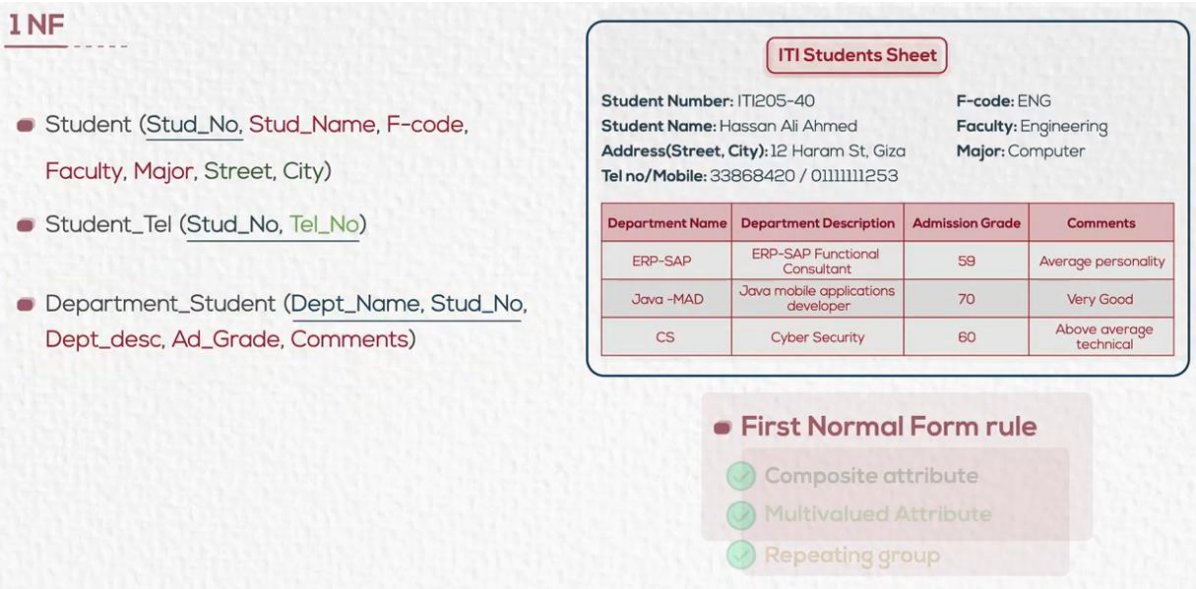
2- Transform 0NF to 1NF

Problem:



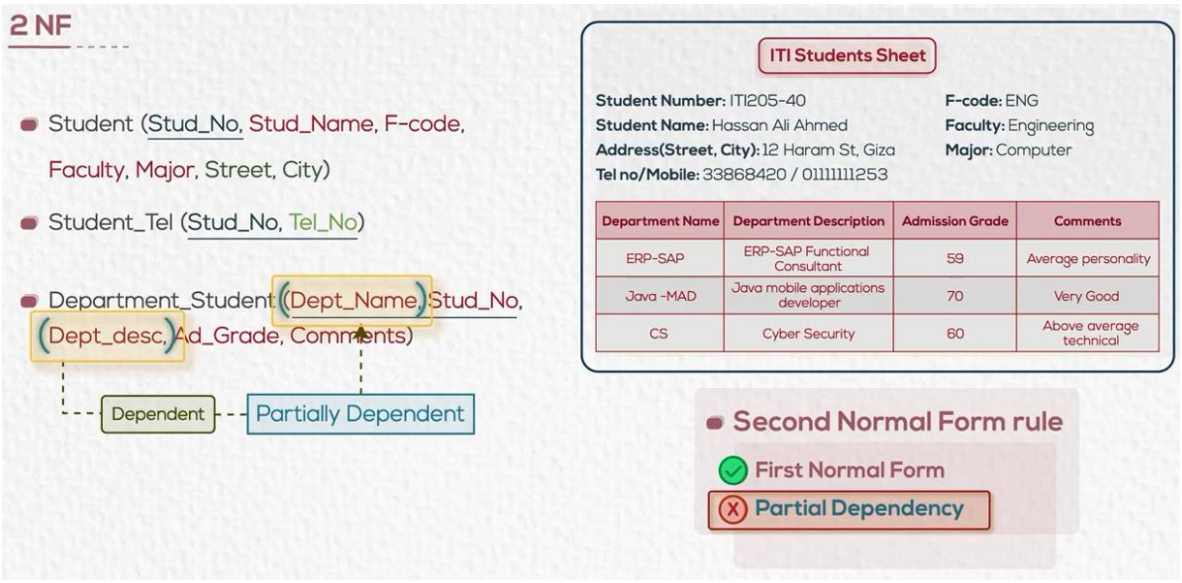
- عندى هنا الأربعة attributes اللى في الاخر دول multi-valued attributes يعنى بيتكرر قيمهم كتير لكل طالب وفى نفس الوقت الأربعة مرتبطين ببعض فكدة يعتبروا Repeating Group.

Solution:



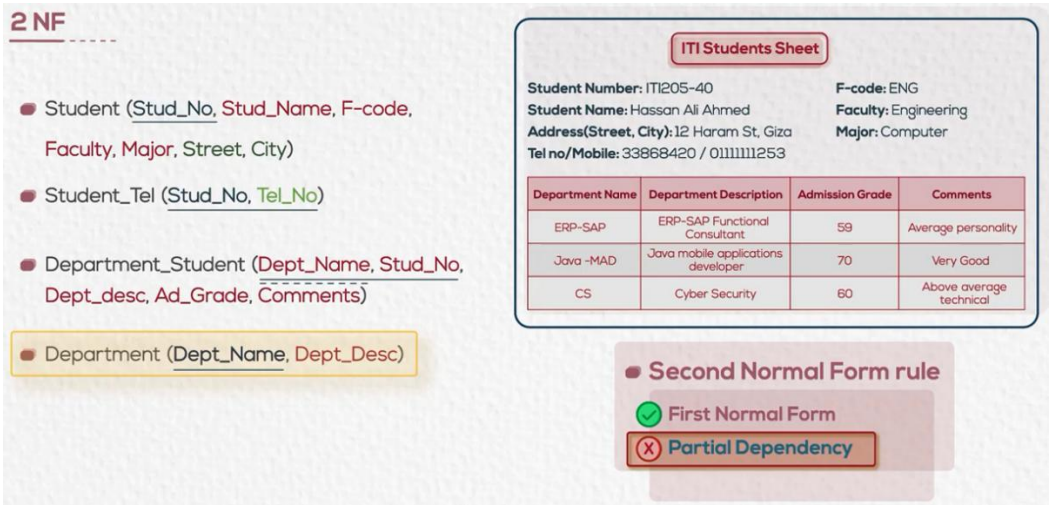
3- Transform 1NF to 2NF

Problem:



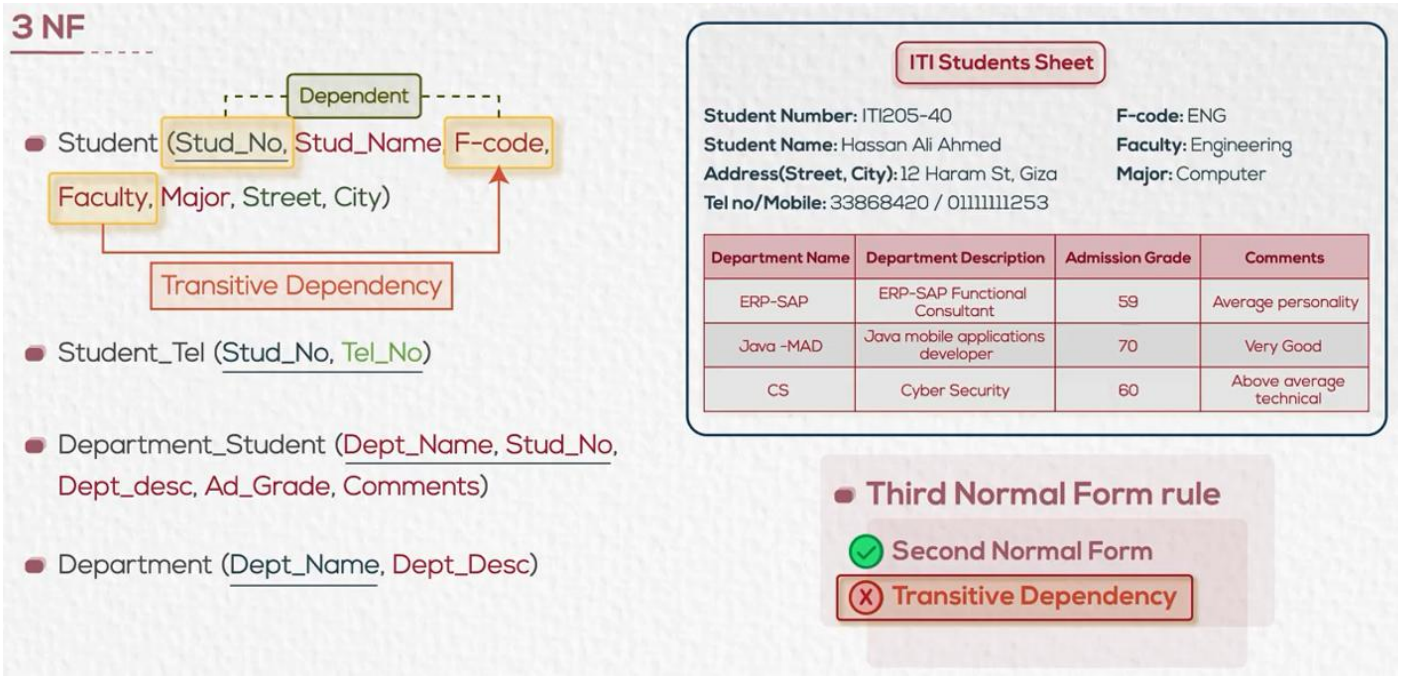
- هنا الDept_desc هي بس اللى partially dependent لأنها معتمدة بس على القسم نفسه ملهاش دعوة برقم الطالب ، لكن الAd_Grade والComments الاتنين محتاجين معلومة القسم ومعلومة الطالب.

Solution:

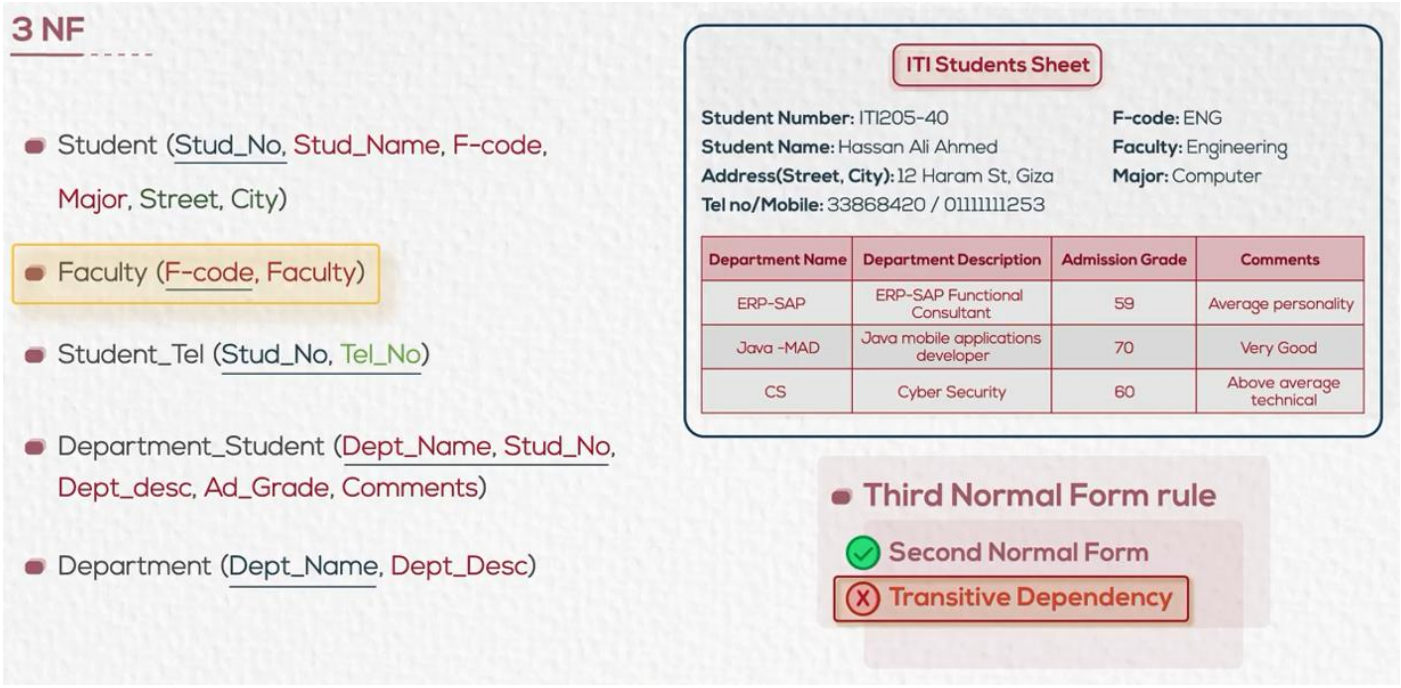


4- Transform 2NF to 3NF

Problem:

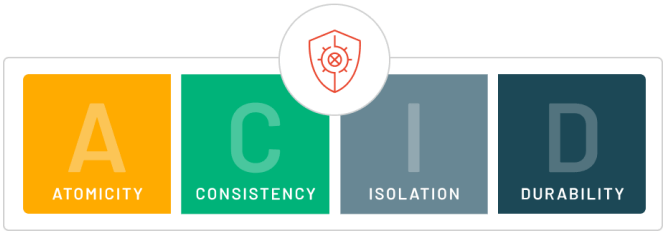


Solution:



ACID Transactions

- A transaction is any operation that is treated as a single unit of work, which either **completes fully** or **does not complete at all** and leaves the storage system in a consistent state.
- The classic example of a transaction is what occurs when you withdraw money from your bank account.
    - Either the money has left your bank account, or it has not — there cannot be an in-between state.
- ACID transactions are a set of properties that ensure reliable processing in database management systems.



- They stand for Atomicity, Consistency, Isolation, and Durability:
    - **Atomicity:**
        - This guarantees that all parts of a transaction are completed successfully.
        - If any part of the transaction fails, the entire transaction is rolled back to its previous state.
        - This ensures a transaction is "all or nothing."
    - **Consistency:**
        - The database must go from one valid state to another, maintaining all predefined rules and constraints.
        - After a transaction, all data must meet the integrity constraints, ensuring the database remains consistent.
    - **Isolation:**
        - Transactions are often executed concurrently.
        - Isolation ensures that the intermediate state of a transaction is invisible to other transactions until it's completed.
        - This prevents interference and maintains transaction integrity, as if each transaction were executed sequentially.
    - **Durability:**
        - Once a transaction is committed, it is permanently saved, even in the event of a system failure.
        - This means any changes made by the transaction are durable and remain in the database.

Example:

```
-- Start a transaction
START TRANSACTION;

-- Example: Transfer $100 from Account A to Account B
UPDATE accounts SET balance = balance - 100 WHERE account_id = 1;  -- Deduct from A
UPDATE accounts SET balance = balance + 100 WHERE account_id = 2;  -- Add to B

-- Commit changes if everything is good
COMMIT;

-- Rollback in case of an error (undo all changes)
ROLLBACK;
```

About the Author

Connect with me on Linkedin: mohaned-ahmad

Explore more at: GitHub Repository