Name : Abdullah Ali Elkholy

LinkedIn

Explain Database (ITI Course) :

❖ Agenda :

Day 0l:

DB Life cycle

DB Design

ERD

File Based System

DB Systems

Day 02:

DB Mapping

DB Schema

SQL

Create DB

Day 03:

joins

Normalization

Day 04:

Aggregate Function

grouping

Union

Subqueries

EERD

Execution Cycle

Day 05:

DB Engine

Services

Ranking Function

Transact-SQL

Day 06:

DB Constraints

Create DB (SQLServer)

Rules —Create DB

Day 07:

Variables

if /while

functions

Day 08:

View

index

Merger -Pivot tables

Day 09:

Stored Procedures

Triggers

XML

Day l0:

backup & restores

Mirroring

Cursor

jobs

SQLCLR

Day 01 :

1. Db Life Cycle

I- Analysis --> System Analyst

Scope (Requirement Document)day

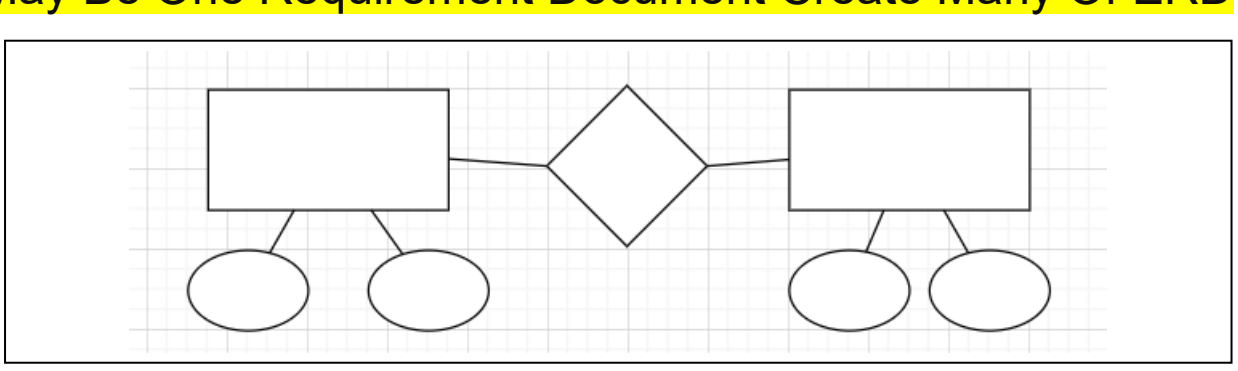
HR System

2- Db Design Db Designer

ERD ➔ Entity Relationship Diagram

Entity Is An Object Which I Want To Store Info About This.

Note : May Be One Requirement Document Create Many Of ERD



3- Db Mapping --> Db Designer

- Set Of Rules
- Actual Schema
- Tables & Relationships

4- Db Implementation (Shared Db)

- Physical Db
- Tool :

RDBMS : Relational Db Management System

(Ex : SQL Server - Oracle - MySQL – Access - SQL (Structure Query Lang)

5-Application

Gui – Interface :

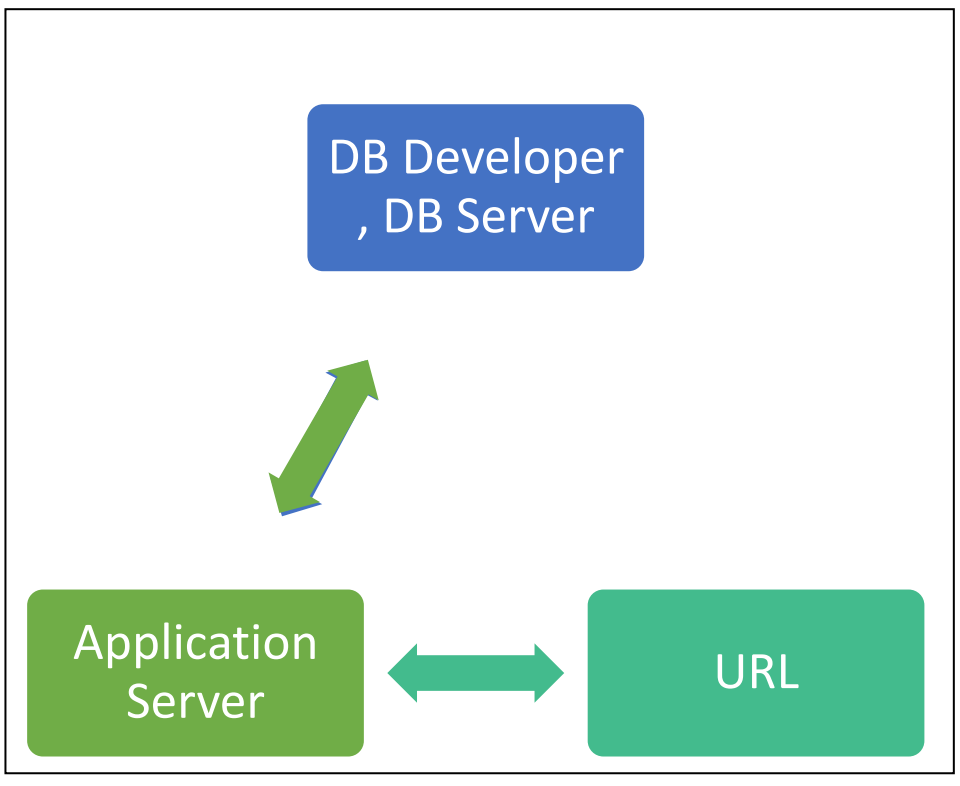
Ex : Web Site Mobile - Desktop

Application Programmer (Db User)

Mvc C# Java React

Application Server

6-Client --> End User --> Browser -> URL



2. File Based System :

Delimited File

1,ahmed,22
3,eman,25
3,eman,25
3,eman,25

Student.Txt

Fixed Width File (Store Using Bytes)

1,ahmed,22
3,eman,25
3,eman,25
3,eman,25

Dept.Txt

❖ **Problems :**

1-Difficult Search
2-Low Performance
3-Separated Copies
4-No Relationship
5-No Db Integrity
6-Db Duplication
7-Long Development Time
8-Security & Permissions
9-Constraints & Rules
10-No Data Quality
11-Manual Backup & Restore
12-No Standardization
13-Diff Integration

3. Db System :

Tables And Relationships .

Table Student :

Sid	Sname	Deptid
1	Ahmed	10
2	Abdo	20
3	Mona	10

Foreign Key

Table Department :

DeptID	Name
10	IT
20	Security
30	SW

Benefits :

1- One Standard
2-Metadata + Data
3- Column --> Datatype
4-Centeralized Db
5- Primary Key (Unique -- Not Null)

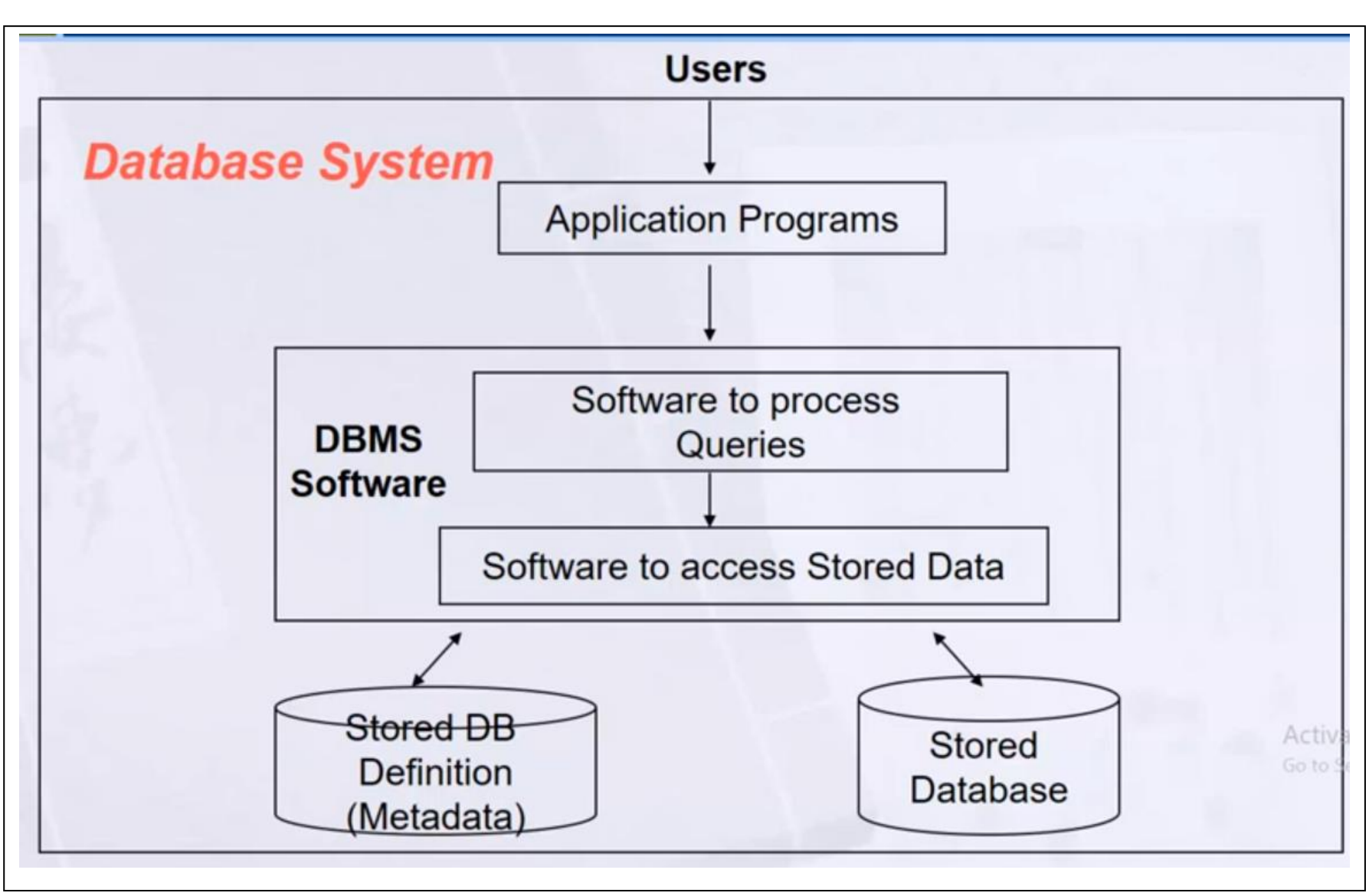
4. Basic Definitions

1. Database:

- A Collection Of Related Data.

2. Database Management System (DBMS):
   - A Software Package/ System To Facilitate The Creation And Maintenance Of Computerized Database.
   - (Model Introduced In 1970 IBM But RDBMS Appears In1980)
3. Database System:
   - The DBMS Software Together With The Data Itself
   - Sometimes, The Applications Are Also Included.
     (Software + Database)

5. Database System :



6. DBMS Advantages :

1. Standardization And Better Data Accessibility And Response (SQL)
2. Sharing Data.: Different Users Get Different Views Of The Data
3. Enforcing Integrity Constraints
4. Improved Data Quality : Constraints, Data Validation Rules
5. Inconsistency Can Be Avoided Because Of Data Sharing.
6. Restricting Unauthorized Access.
7. Providing Backup And Recovery : Disaster Recovery Is Easier
8. Minimal Data Redundancy : Leads To Increased Data Integrity/Consistency
9. Program-Data Independence

- Metadata Stored In DBMS, So Applications Don't Worry About Data Formats
- Data Queries/Updates Managed By DBMS.

7. DBMS Disadvantages

1. It Needs Expertise To Use
2. DBMS Itself Is Expensive.
3. The DBMS May Be Incompatible With Other Available DBMS

8. Database Users

1. Database Administrator (DBA).
2. System Analysts.
3. Database Designer.
4. Database Developer.
5. Application Programmers
6. Bi & Bigdata Specialist (Data Scientist).
7. End Users.

9. Entity Relationship Diagram Concepts :

Entity-Relationship Diagram (ERD) :

- Identifies Information Required By The Business By Displaying The Relevant Entities And The Relationships Between Them.

10. The ER Model

Basic Constructs Of The E-R Model:

1. **Entities :**
   - Person, Place, Object, Event, Concept (Often Corresponds To A Real Time Object That Is Distinguishable(يمكن تمييزه) From Any Other Object)
2. **Attributes :**
   - Property Or Characteristic Of An Entity Type (Often Corresponds (يتوافق) To A Field In A Table)
3. **Relationships :**
   - Link Between Entities (Corresponds To Primary Key-Foreign Key Equivalencies In Related Tables)

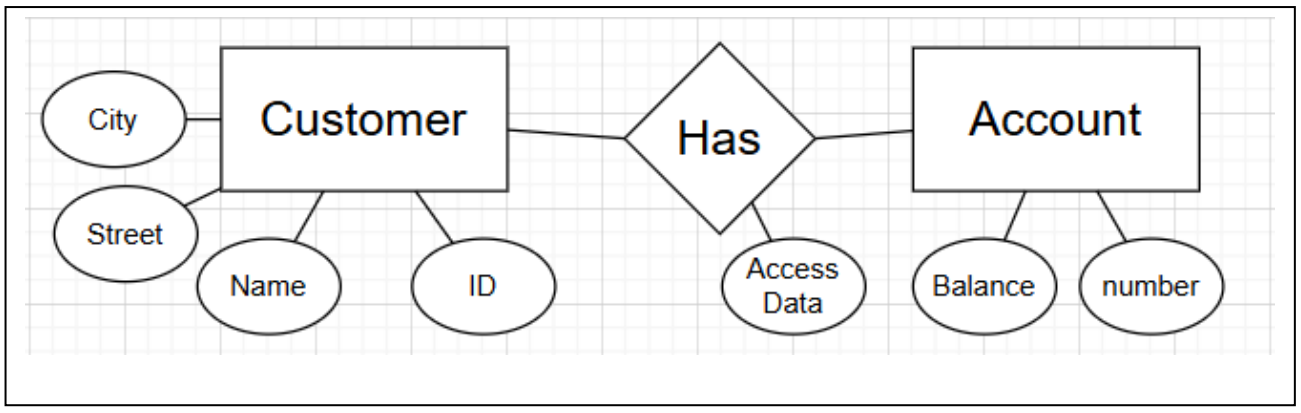
11. ER Diagram: Starting Example :

> Rectangles: Entity Sets

> Diamonds: Relationship Sets

> Ellipses: Attributes

> Access Data Attribute : Is Shared Between Customer And Account



Note :

1. When U Are Reading Requirement Document :

   -> **Verbs** Indicate To Relationship Between Entities

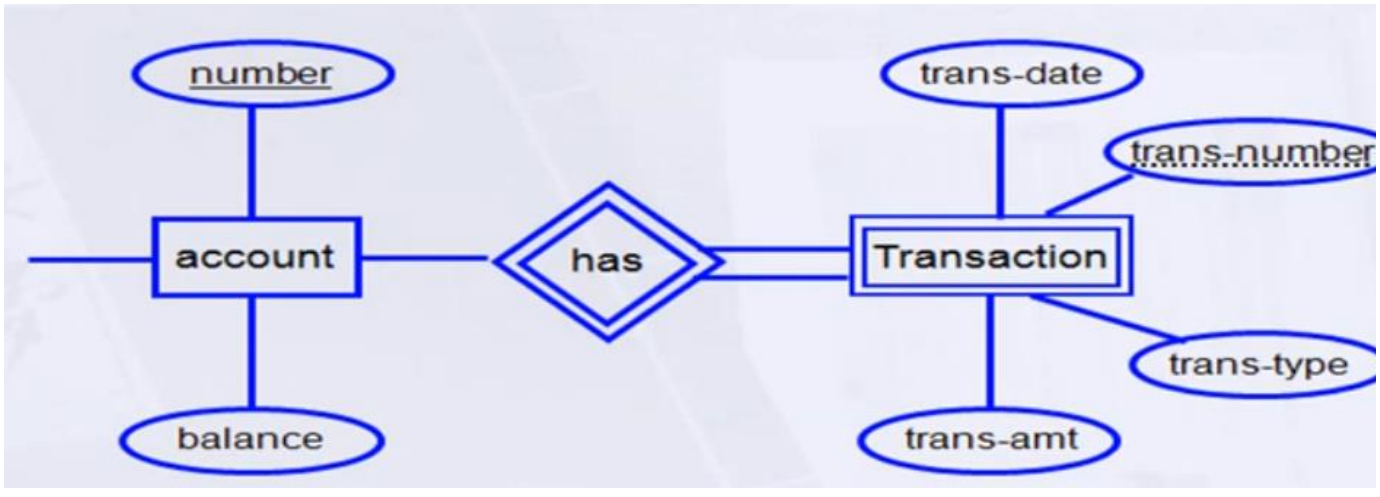
   -> **Nouns** Often Indicates To Names Of Entity Or Attributes

2. May Be Multiple Relationship Between Two Entities , Must Indicates To Different Meaning.

12. Strong Entity Vs Weak Entity

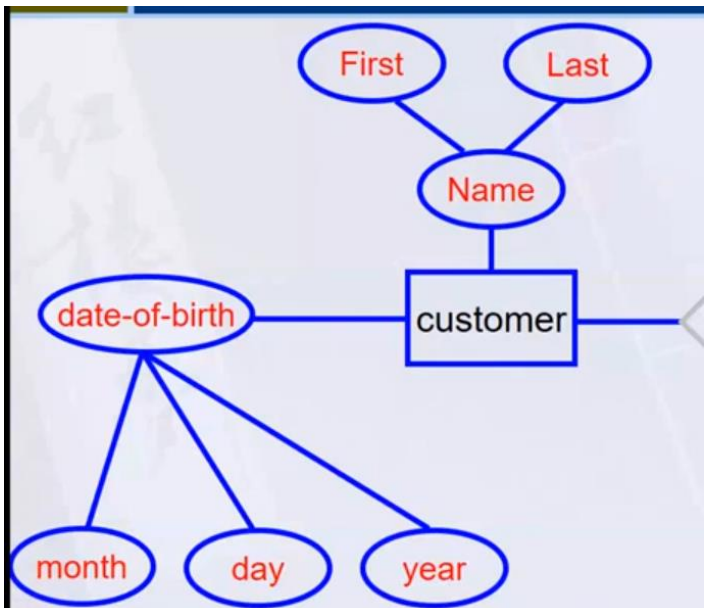
1. A Strong Entity :
   - ❖ An Entity Set That Has A Primary Key.
2. A Weak Entity :
   - ❖ An Entity Set That Do Not Have Sufficient(كافٍ) Attributes To Form A Primary Key.
3. Partial Key:
   - ❖ A Set Of Attributes That Can Be Associated With P.K Of An Owner Entity Set To Distinguish (يميز) A Weak Entity.

Ex :



13. Next: Types Of Attributes

1. Composite Attribute

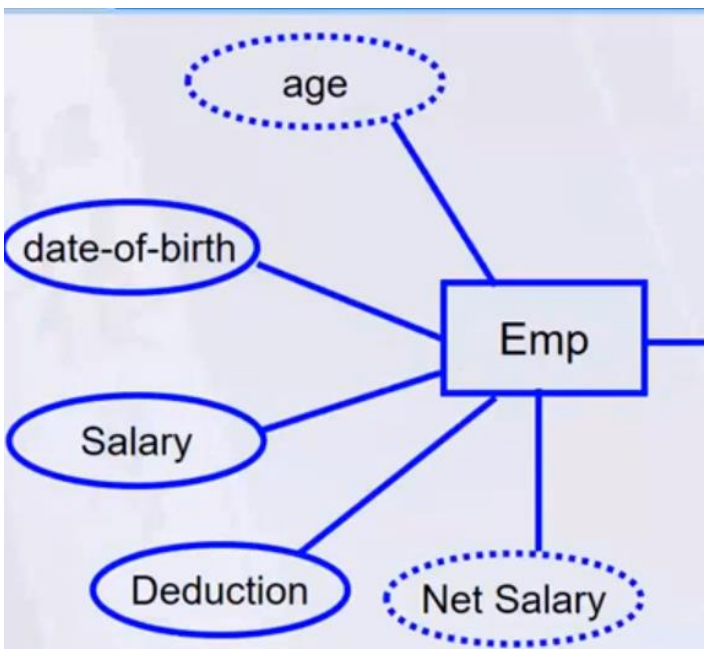


2. Multi-Valued Attribute :

- Can Multiply To The Person.
- Double Ellipse.

3.Derived Attribute :

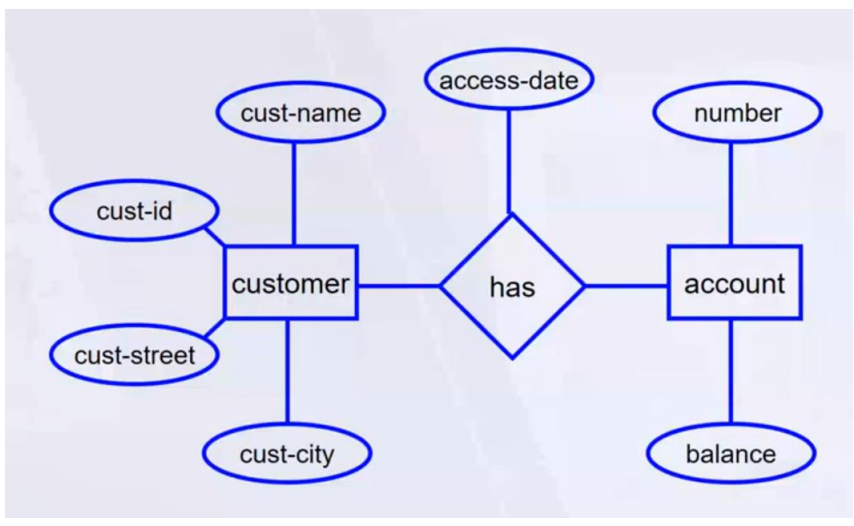
- Calculate At Runtime.
- Dashed Ellipse.



4. Complex Attribute

5. Simple Attribute :

- It Cannot Be Divided.
- It Cannot Calculate At Runtime.
- No Repeated To The Same Person.



14. Relationship:

- ❖ A Relationship Is An Association Among Several Entities.
- ❖ A Relationship May Also Have Attributes

For Example :

Consider The Entity Sets Customer And Loan And The Relationship Set Borrower. We Could Associate The Attribute Date-Issued To That Relationship To Specify The Date When The Loan Was Issued.

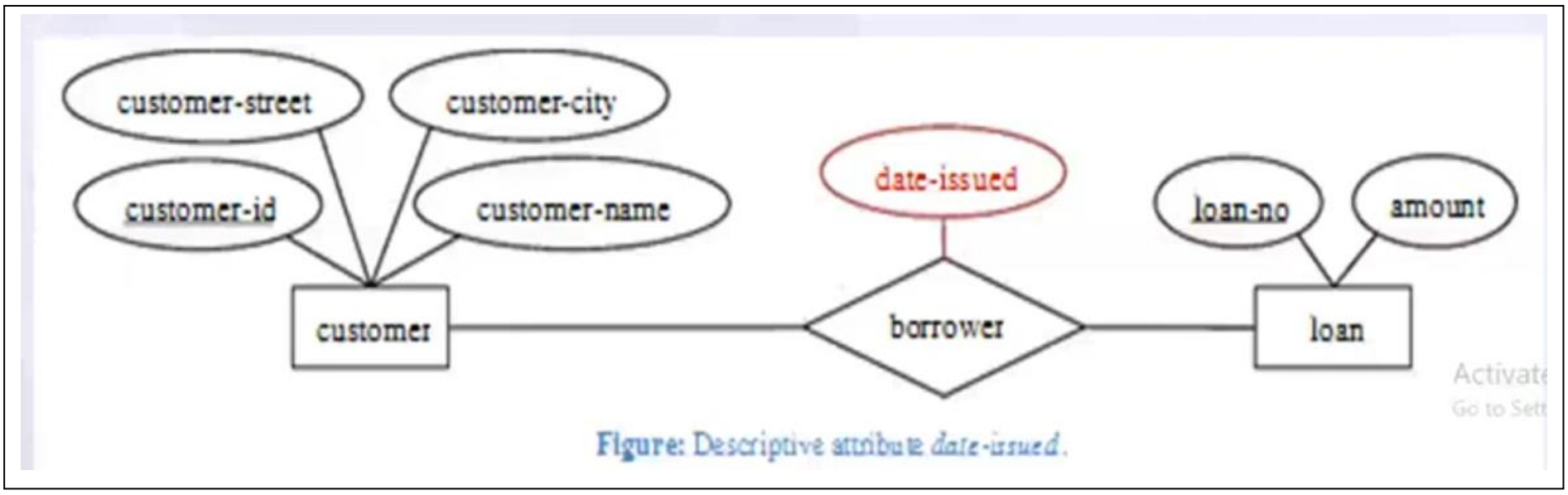
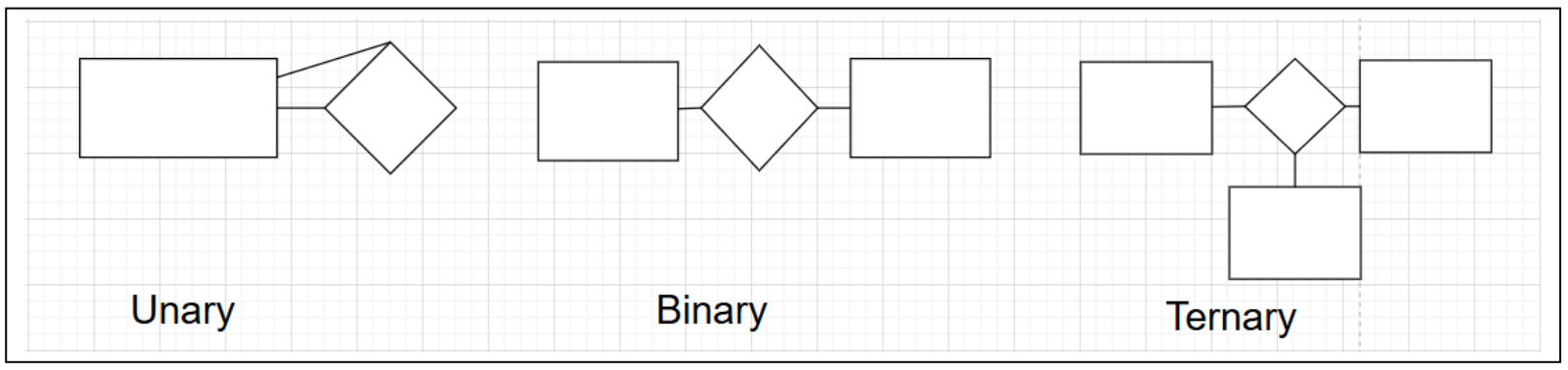


Figure: Descriptive attribute *date-issued*.

15. Relation Types :

Relation Has Three Properties:

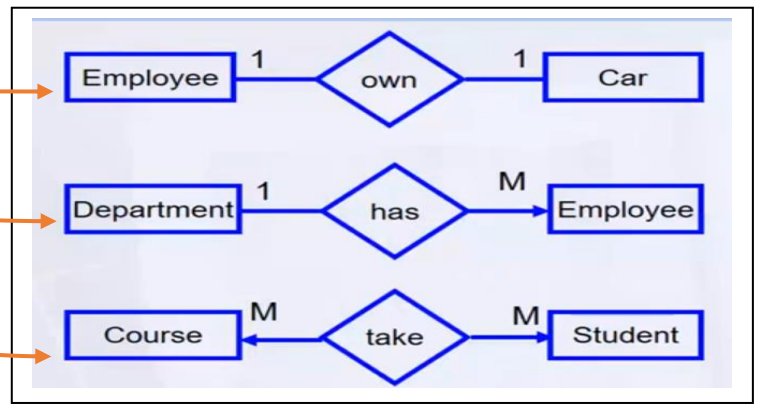
- ❖ Degree Of Relationships :
  - **Degree:**
    Number Of Entity Types That Participate In A Relationship
  - **Three Cases :**
    - ✓ **Unary**:
      - Between Two Instances Of One Entity Type
      - A Relations In Which The Same Entity Participates More Than Once
    - ✓ **Binary:** Between The Instances Of Two Entity Types
    - ✓ **Ternary**: Among The Instances Of Three Entity Types



❖ Cardinality Constraint

> How Many Instances Of One Entity Will Or Must Be Connected To A Single Instance From The Other Entities.

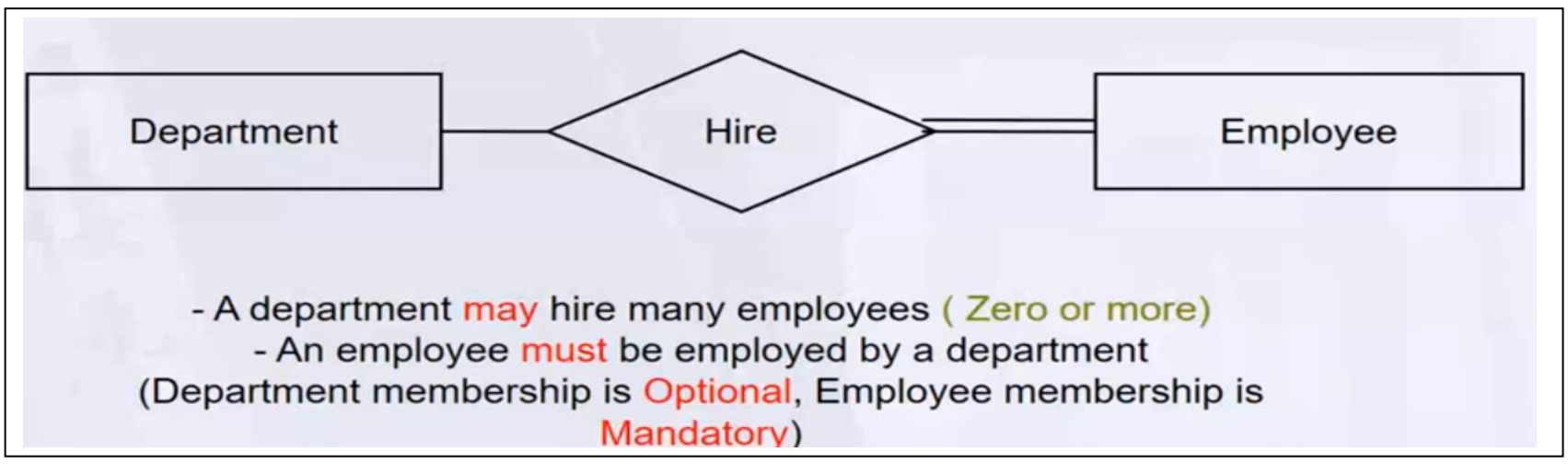
1. One-One Relationship
2. One-Many Relationship
3. Many- Many Relationship



❖ Participation Constraint :

- An Employee Must Work For A Department
- An Employee Entity Can Exist Only If It Participates In A Works_For Relationship Instance So This Participation Is **Total**
- Only Some Employees Manage Departments The Participation Is **Partial**

Example :



Note :

=> Partial : If Exist : (Zero Or More – May - Optional)

=> Total : If Exist : (Must – One Or More – Mandatory(إلزامي))

16. Keys

> Different Types Of Keys:

1. Candidate Key: Search For Attribute Validate As A Primary Key
2. Primary Key : Unique And Not Null , Underline
3. Foreign Key
4. Composite Key
5. Partial Key
6. Alternate Key
7. Super Key

17. Summery:

- **Db Life Cycle**

  I-Analysis -> System Analyst
    - Requirement Document

  2-Db Design -> Db Designer
    - ERD

  3-Db Mapping --> Db Designer
    - Db Schema (Tables& Relationships)

  4-Db Implementation ---> Db Developer
    - SQL (Physical Db)

  5-Application -> Application Programmer
    - GUI -- Interface
    - Web Desktop Mobile

  6-Client End User

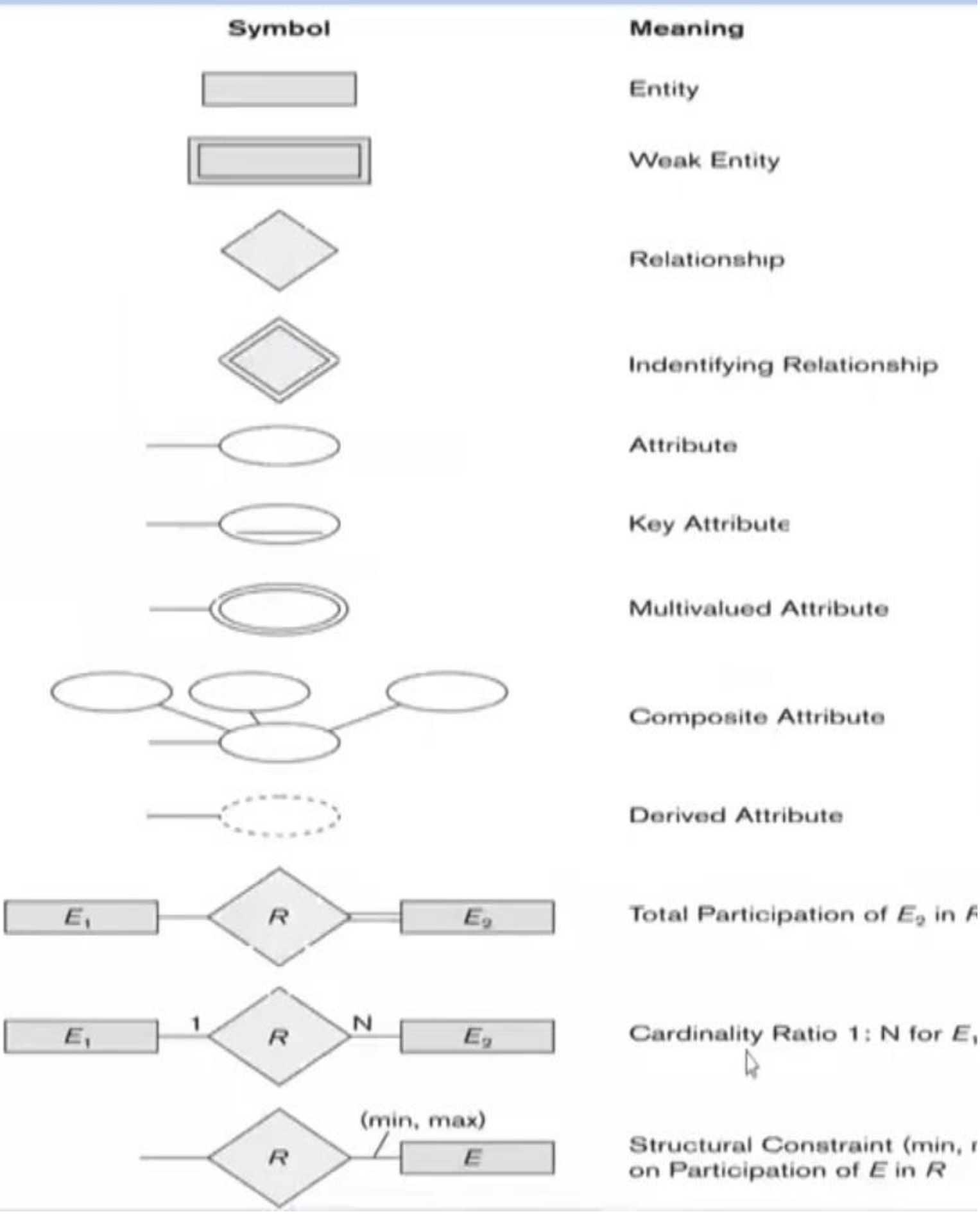
- **ERD :**

1. Entity
   - Strong Entity PK
   - Weak Entity Partial Key
2. Attributes
   - Simple
   - Composite
   - Multivalued
   - Computed
   - Complex

- **Relationship**
  - Degree
    - Unary
    - Binary
    - Ternary
  - Cardinality
    - 1 1
    - 1 M
    - M M
  - Participation
    - total
    - Partial

18. Notation Of ERD :

Figure 3.14
Summary of the notation for ER diagrams.



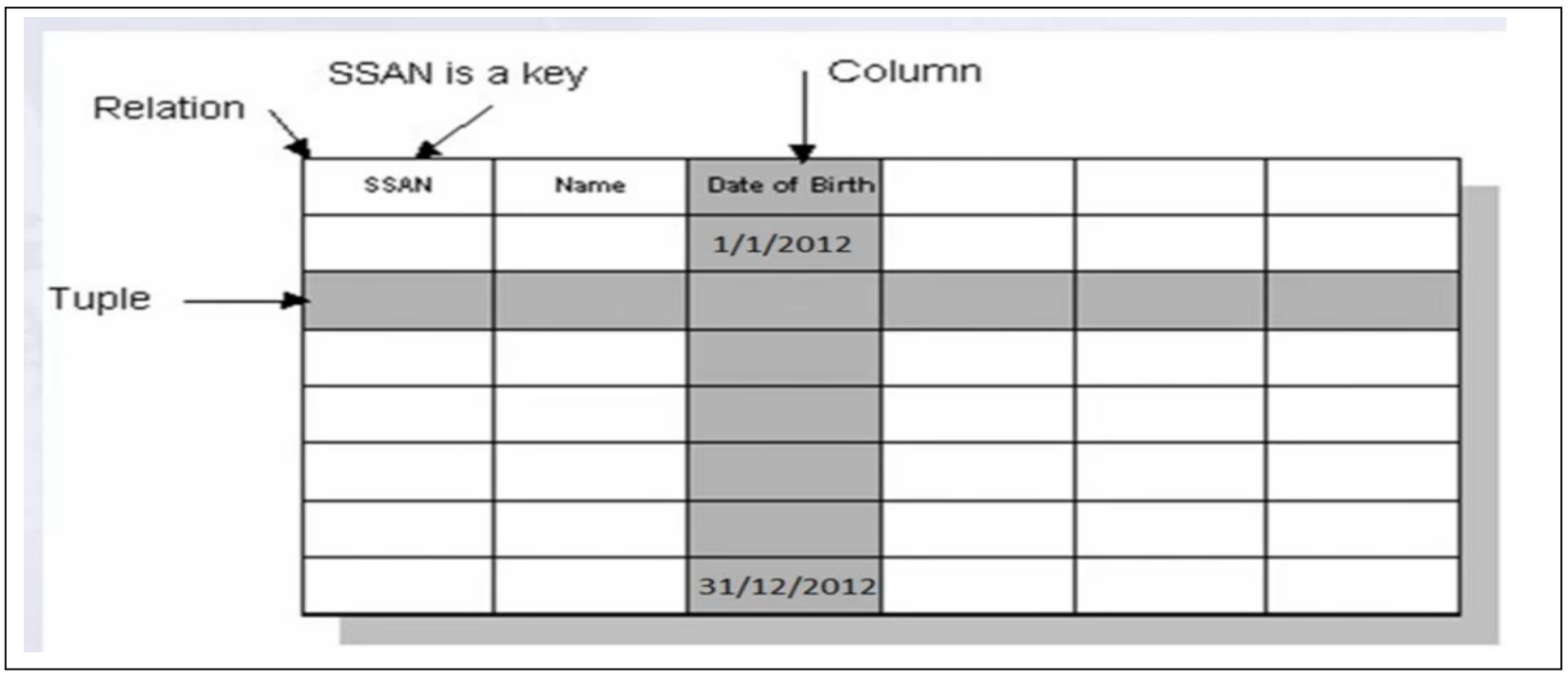
Day 02 :

Note :

- I Can’t Delete Parent Has A Child
- Constraint File :
  - Write In It Any Constraint Or Role To Use .

1. Relational Database Definitions :

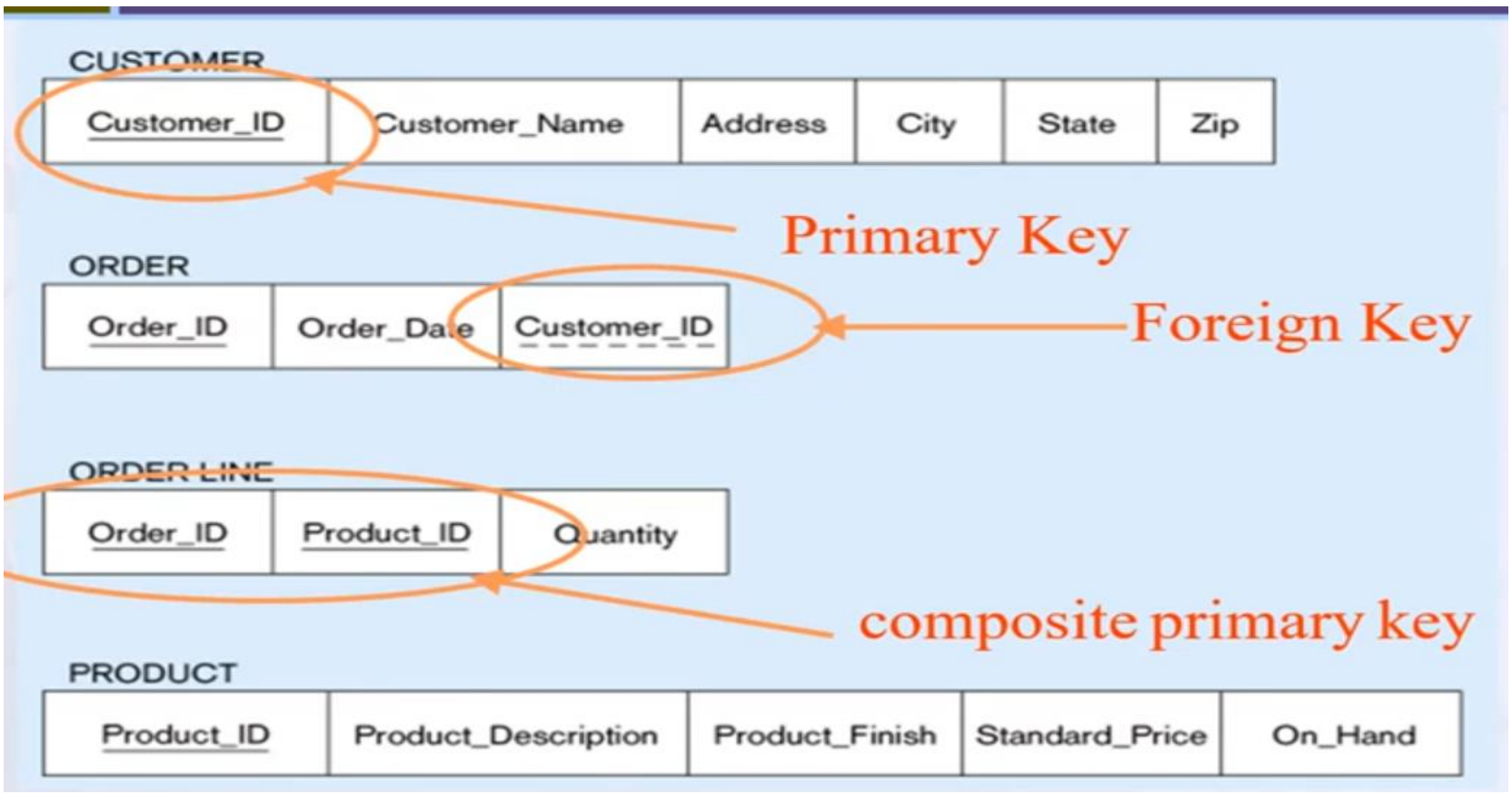
♣ Table Or Entity: A Collection Of Records
♣ Attribute Or Column Or Field: A Characteristic Of An Entity
♣ Row Or Record Or Tuble: The Specific Characteristics Of One Entity
♣ Database: A Collection Of Tables



♣ Characteristics Of Column :

1. Domain
2. Datatype
3. Quality
4. Size
5. Tinyint
6. 1b
7. -128:127
8. Constraints & Rules

2. Mapping Convert To Db Table :



3. ERD-To-Relational Mapping

Step 0: Relation 1:1 Total Participation

Step 1: Mapping Of Regular(Strong) Entity Types

Step 2: Mapping Of Weak Entity Types

Step 3: Mapping Of Binary 1:1 Relation Types

Step 4: Mapping Of Binary 1:N Relationship Types.

Step 5: Mapping Of Binary M:N Relationship Types.

Step 6: Mapping Of N-Ary Relationship Types.

Step 7: Mapping Of Unary (Self) Relationship.

Step 1. Mapping Of Regular Entity Types :

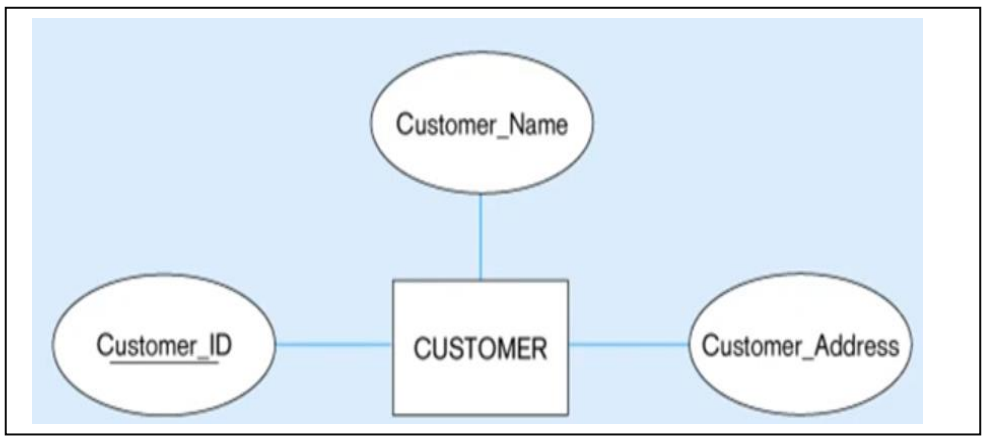
- Create Table For Each Entity Type -> If There Is No 1-1 Relationship Mandatory (Total) From 2 Sides
- Choose One Of Key Attributes To Be The Primary Key

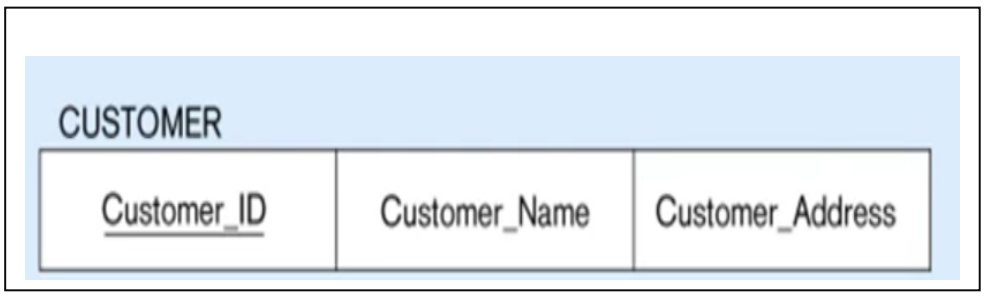
Example :

1. Mapping Simple Attribute :

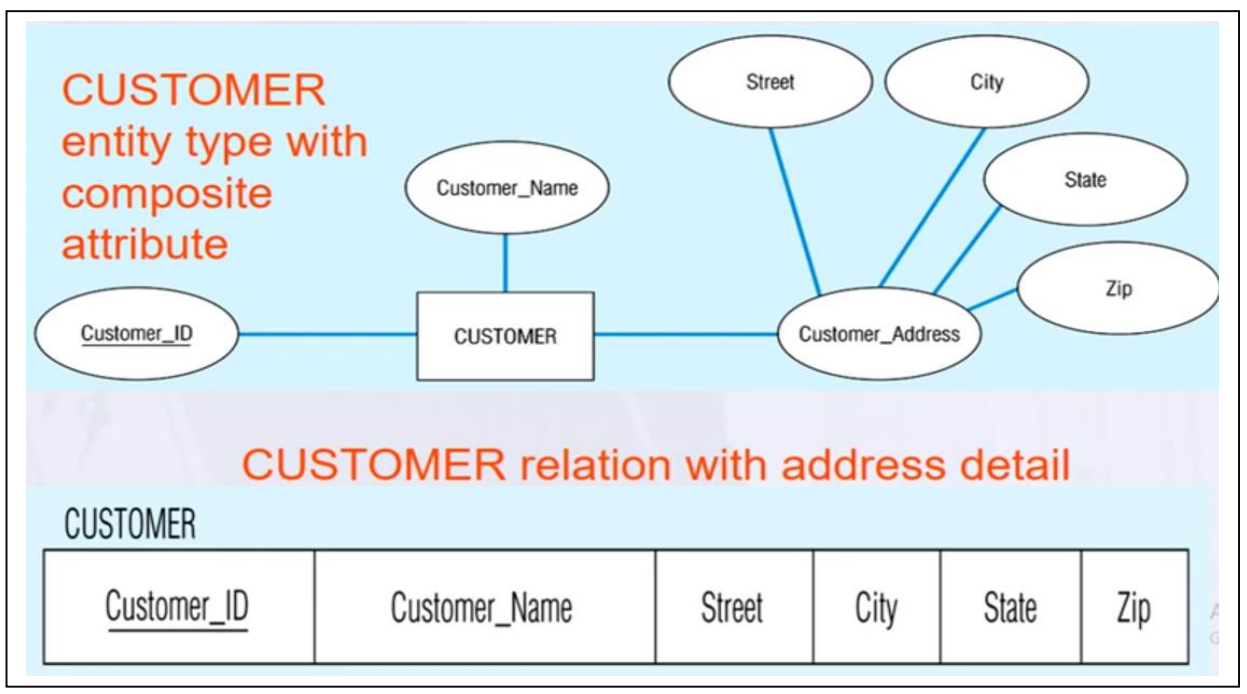
A. Customer Entity Type With Simple Attributes:

(B) Customer Relation

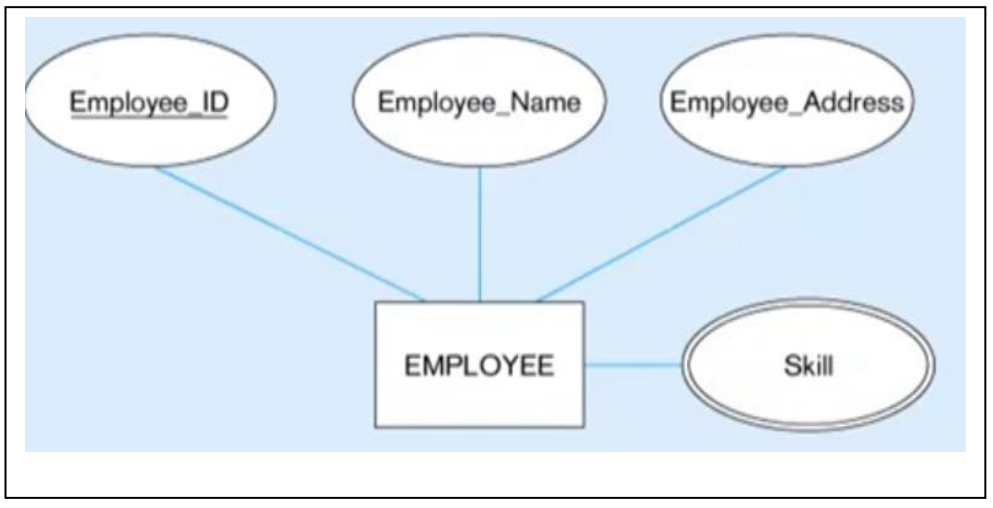


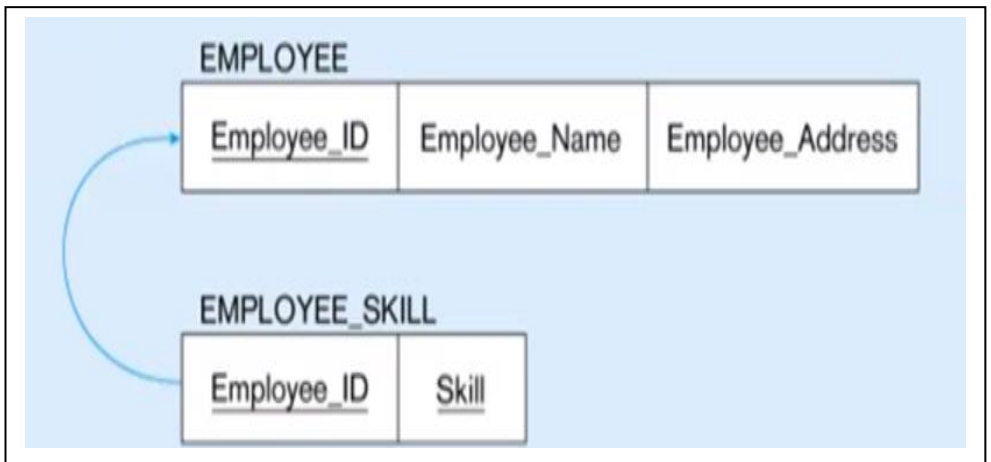


2. Mapping Composite Attribute :



3. Mapping Multivalued Attribute :





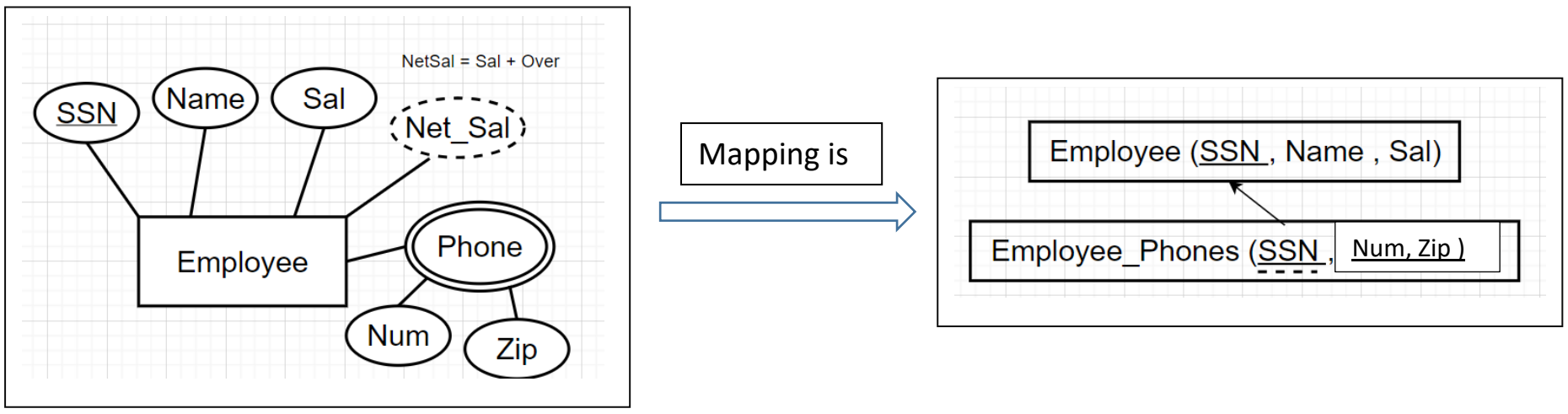
4. Mapping Derived & Complex

- In The Most Cases **Derived** Attribute **Not** Be Stored In Db.
- Mapping Complex Like Mapping Multivalued Attribute Then Including Parts Of The Multivalued Attributes As Columns In Db

Note :

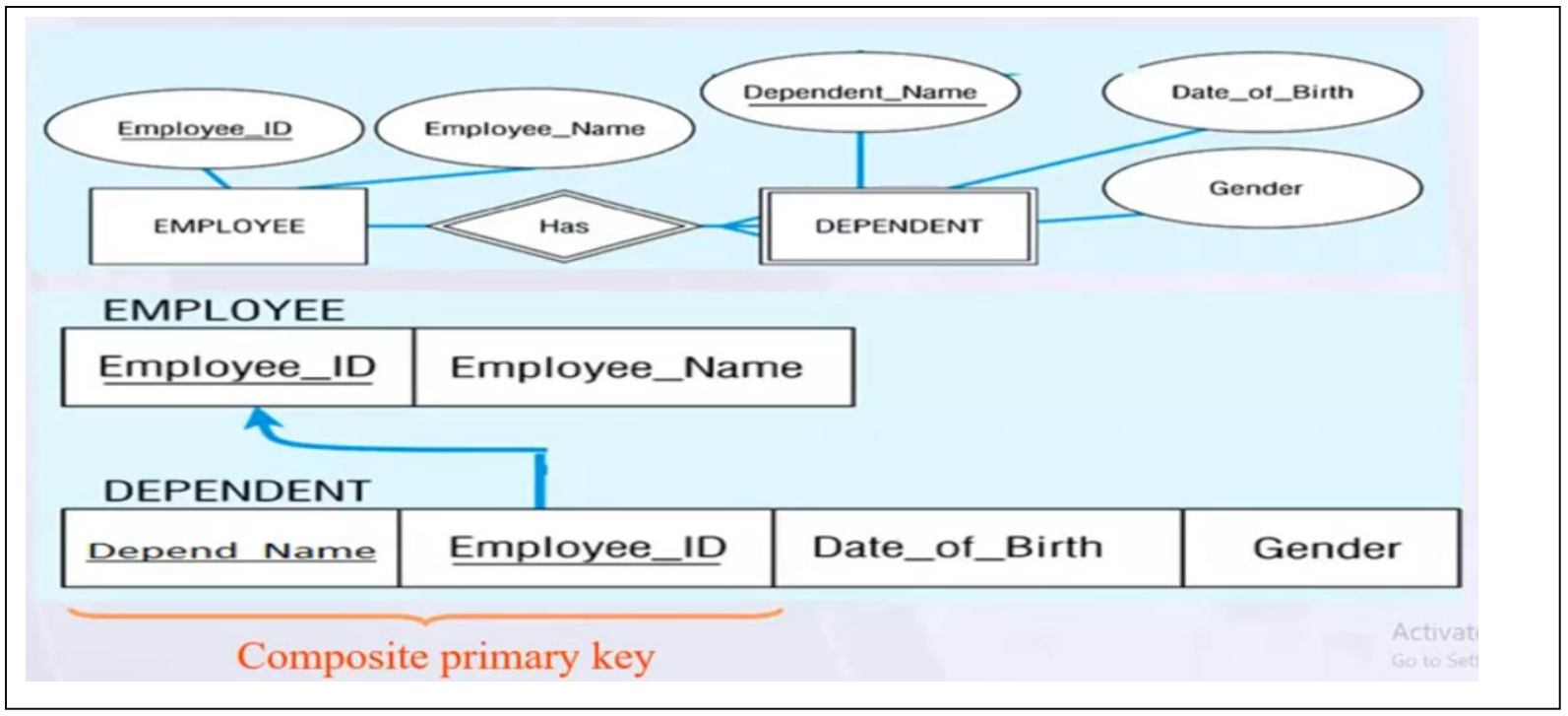
- When I Store Derived Attribute In Db :
  - If It Has A Relation With Another Table .

- If Number Of Records Is Huge And To Calculate This , It Will Take A Large Time .



Step 2 , Mapping Of Weak Entity Types

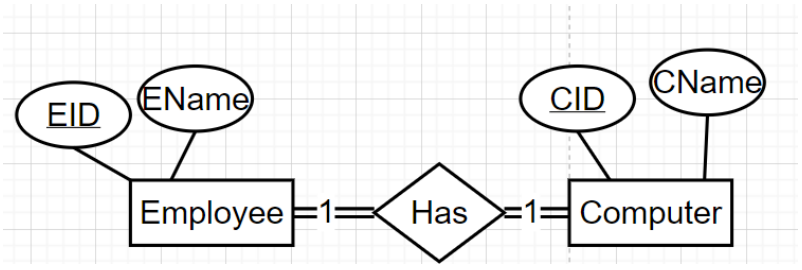
- Create Table For Each Weak Entity.
- Add Foreign Key That Correspond(تتوافق مع)To The Owner Entity Type.
- Primary Key Composed Of:
    - Partial Identifier Of Weak Entity
    - Primary Key Of Identifying Relation (Strong Entity)



Step 3: Mapping Of Binary 1:1 Relation Types :

- Merged Two Tables If Both Sides Are Mandatory(Total).
- Add Fk Into Table With The Total Participation Relationship To Represent Optional Side.
- Create Third Table If Both Sides Are Optional.

1. Mandatory - Mandatory :



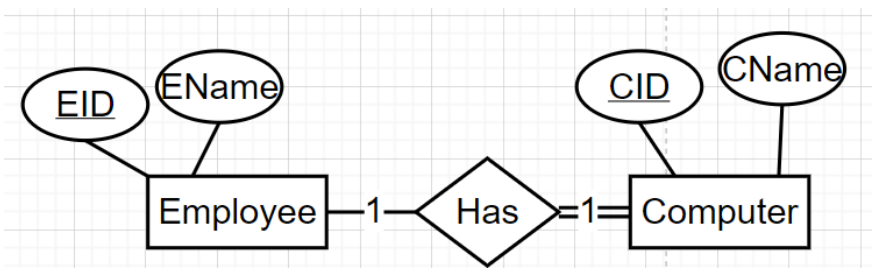
Mapping ⟹

Emp (EID , EName , CID , CName)

Create one table

tbl-xy(PK , ...,..), PK=PKx or PKy

2. Optional – Mandatory :

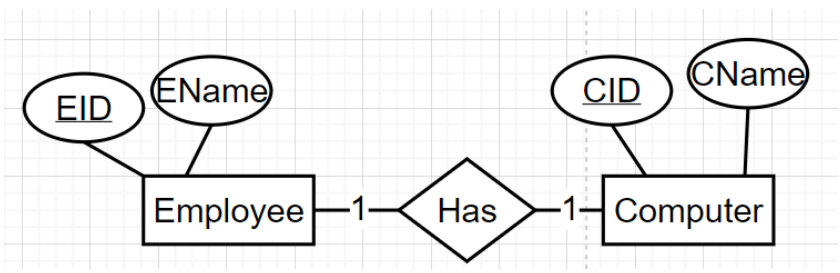


Mapping ⟹

Emp (EID , EName)

PC(CID , CName,EID_FK)

3. Optional – Optional:



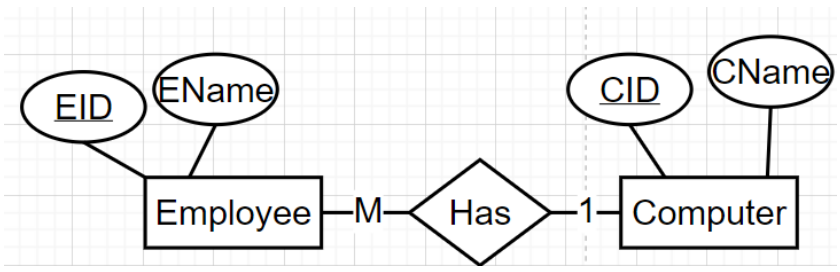
Mapping ⟹

Emp (EID , EName)

PC(CID , CName)

Emp_PC(EID , CID)-> FK=PKx or PKy

4. Many Is Optional:



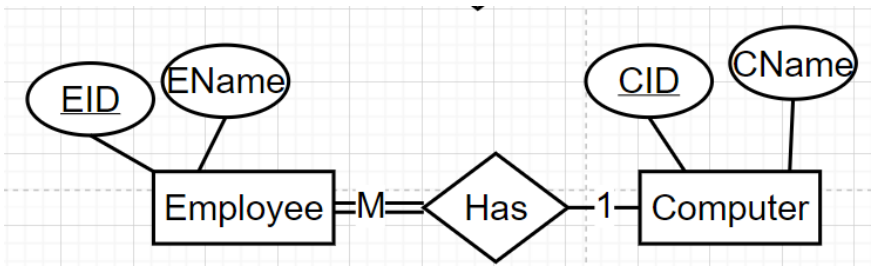
Mapping ⟹

Emp (EID , EName)

PC(CID , CName)

Emp_PC(EID , CID)-> FK=PKx

5. Many Is Mandatory:



Mapping ⟹

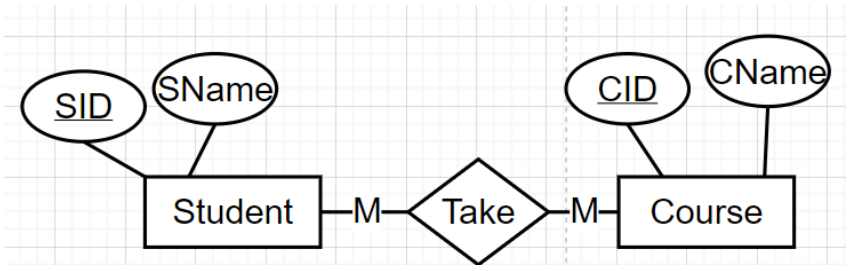
Emp (EID , EName, CID_FK)

PC(CID , CName)

Step 5: Mapping Of Binary Relationship Types :

- Create A New Third Table
- Add Fks To The New Table For Both Parent Tables
- Add Simple Attributes Of Relationship To The New Table If Any .

1. M To N :



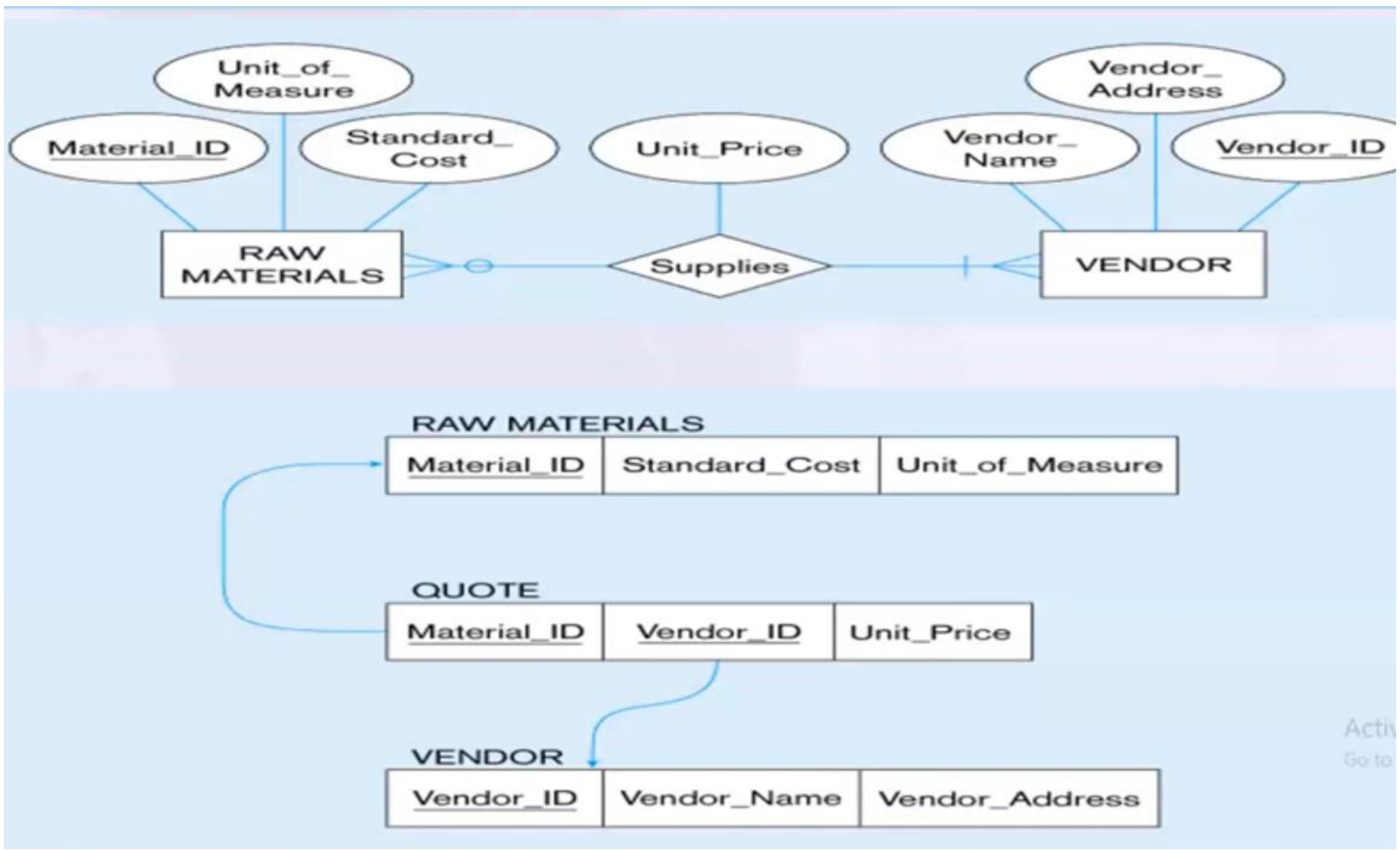
Mapping

Student (SID , SName)

Course(CID , CName)

2. M To N

Std_Crs(SID,CID)



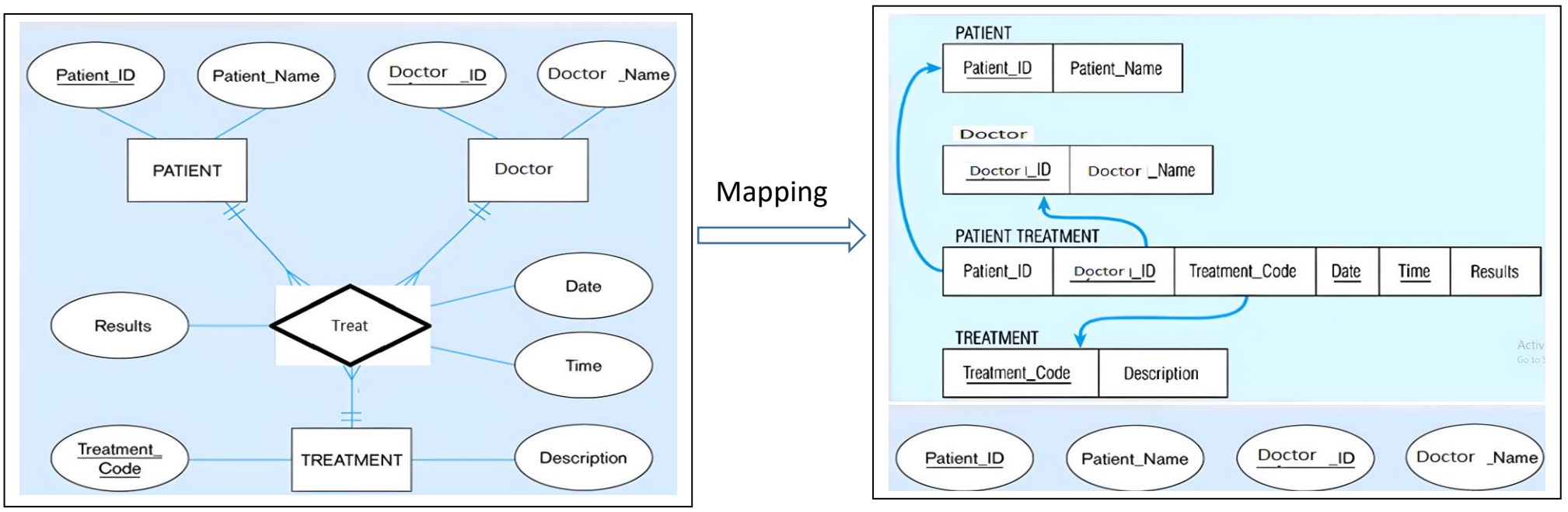
With Attributes :

Step 6: Mapping Of N-Ary Relationship Types.

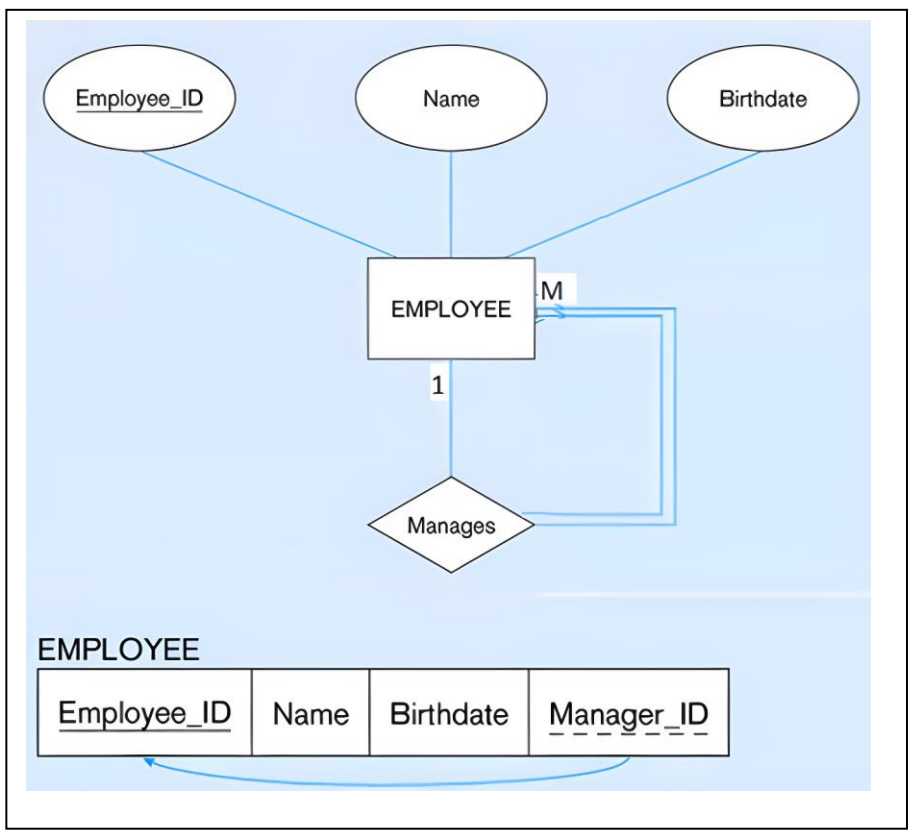
> If N > 2 Then :

- Create A New Third Table (# Col = # Of Parents + # Attributes)
- Add Fks To The New Table For All Parent Tables
- Find A Pattern For Primary Key.

Example :



Step 7'.Mapping Unary Relationship :



Search For :

if Relationship is self M : N

Note :

Ensure That “MSSQLServer” Is Running :

To Running It , Go To Search Icon -> Services -> MSSQLServer -> Run It

4. ANSI SQL:

♣ Ansi :

- Is : American National Standards Institute (معهد)
- Adopted The Sql Standards In 1986.
- Software Vendors Use The Ansi Sql Standards To Build Sql Database Software For Developers.

♣ Kinds :

1. Microsoft Transact-Sql
2. Oracle -> Pl-Sql
3. Ibm -> Ibm-Pl-Sql
4. Open Source -> Mysql

1. Microsoft Transact-Sql :

Types Are :

1) Data Definition Language (**DDL**):
   - Use To Design The Database Structure(MetaData + Structure).
   - Use To Create And Modify Database Objects Based On The Business Requirements.
   - For Example:
     [Create Table - Create View - Create Function – Alter – Drop – Select Into]

2) Data Query Language (DQL) :
   - Consists Of Instructions For Retrieving Data Stored In Relational Databases.
   - For Example :
     [Select - Aggregation Function – Grouping – Union – Joins – Subqueries]

3) Data Manipulation Language (DML) :
   - Statements Write New Information Or Modify Existing Records In A Relational Database.
   - For Example [Insert – Update – Delete – Merge]

4) Data Control Language (DCL) :
   - Database Administrators Use (DCL) To Manage Or Authorize Database Access For Other Users.
   - For Example :
     - Grant(وهب) :Gives Permission On A Specified Securable To The Principal.
     - Revoke (الغاء) : Remove The Previously Granted Or Denied Permissions
     - Deny : Denies Permission To A Principal For Accessing The Securable

5) Transaction Control Language (TCL)
   - The Relational Engine Uses (TCL) To Automatically Make Database Changes.
   - For Example :
     - **Rollback** : To Undo An Erroneous Transaction.
     - **Commit** : Lets A User Save Any Changes Or Alterations On The Current Transaction. These Changes Then Remain Permanent.
     - **Begin Transaction** .

♣ **Database Files :**

1. **.Mdf : Meta Data File** , Contains Tables , Data ,Metadata
2. **.Ldf :Log File** : Contains Transactions, Archive About Db

♣ **Create Database :**

1. Using Code Sql : Create Database [*Database Name]*;
2. Using Manual : Click Right On Database -> Create New Db

♣ **Back Up Db :**

1. Click Right On Your Db Name -> Task -> Back Up

♣ **Restore Db :**

1. Click Right On Database -> Restore Db

♣ **Create Table :**

1. Way Wizard :
   Click Right On Your Db Name -> Create New Table
2. Using Code :

```
Create Table Table_Name (
    Column1 Datatype,
    Column2 Datatype,
    Column3 Datatype,
   ....
);
```

♣ **Alter after create a Table :**

- Add Column :
  Alter Table [Table Name] Add [Column Name] [Data Type]
- Alter Data Type Of Column :
  Alter Table [Tablename] Alter Column [Column Name] [New Dt]
- Drop Column :
  Alter Table [Tablename] Drop Column [Column Name]
- Add Constraint After Create table:
  Alter Table [Tablename] add constraint [constraint name] [type]
- drop Constraint After Create table:
  Alter Table [Tablename] drop constraint [constraint name]

♣ **Drop A Table :** Drop Table [Table Name]

♣ **Insert Data In A Table :**

1. Insert Into *Table_Name* (*Column1*, *Column2*, *Column3*, ...)
   Values (*Value1*, *Value2*, *Value3*, ...),
   (*Value1*, *Value2*, *Value3*, ...),
   (*Value1*, *Value2*, *Value3*, ...);

♣ **Update Records In Db :**

1. Update *Table_Name*
   Set *Column1* = *Value1*, *Column2* = *Value2*, ...
   Where *Condition*;

♣ **DELETE Statement :** is used to delete existing records in a table.

1. DELETE FROM *table_name* WHERE *condition*;

Note :

1. To Enable Alter Table As Metadata :

   Tools -> Options -> Designers -> Check Off “Prevent Saving Changes That Require Table Re-Creation”

2. When U To Change Datatype :
   - Must Source And Destination Dt At The Same Category
   - Range Dt
   - Are Exist Data Matching With New Dt

Day 03 :

1. Types Of Joins

1. Cross Join
   - Cartesian Product
2. Inner Join
   - Equi Join
3. Outer Join
   - Left Outer Join
   - Right Outer Join
   - Full Outer Join
4. Self Join :
   - (Unary Relationship)

First : Cross Join -> Cartesian Product

- Matches Each Row Of One Table To Every Other Row Of Another Table,
- Only When The **Where** Condition Is Not Specified In The Query.
- In Case The **Where** Condition Is Specified Then The Cartesian Join Works As An **Inner Join**.
- **Code** :

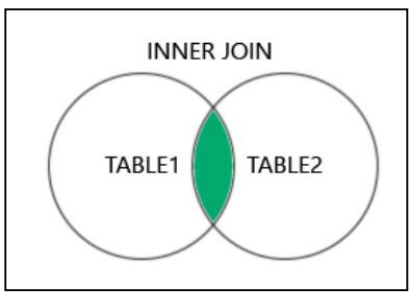
```
Select * From Table1 Cross Join  Table2
Select * From Table1 , Table2
```

- Testers And Quality Assurance Use This Join To Produce A Big Date For Testing

Second : Inner Join :

- Selects Records That Have Matching Values In Both Tables.
- **Code** :

```
Select Column_Name(S)
From Table1inner Join Table2
On Table1.Column_Name = Table2.Column_Name;
```

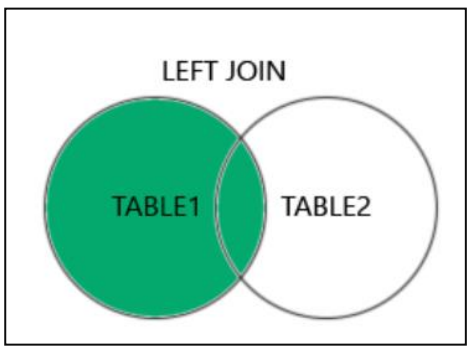


Third : Outer Join :

1. Left Outer Join :

- Returns All Records From The Left Table (Table1), And The Matching Records From The Right Table (Table2).
- The Result Is 0 Records From The Right Side, If There Is No Match.
- **Code** :

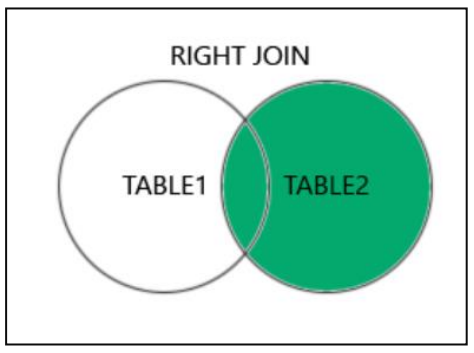
```
Select Column_Name(S)
From Table1
Left Join Table2
On Table1.Column_Name = Table2.Column_Name;
```



2. Right Outer Join :

- Returns All Records From The Right Table (Table2), And The Matching Records From The Left Table (Table1).
- The Result Is 0 Records From The Left Side, If There Is No Match.
- **Code** :

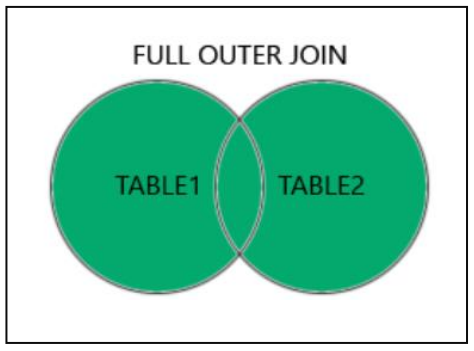
```
Select Column_Name(S)
From Table1
Right Join Table2
On Table1.Column_Name = Table2.Column_Name
```



3. Full Outer Join Or Full Join:

- Returns All Records When There Is A Match In Left (Table1) Or Right (Table2) Table Records.
- **Code** :

```
Select Column_Name(S)
From Table1
Full Outer Join Table2
On Table1.Column_Name = Table2.Column_Name
Where Condition;
```



Forth : Self Join:

- Is A Regular Join, But The Table Is Joined With Itself.
- Code :

```
Select Emp.Ename , Super.Ename
From Employee Emp , Employee Super
Where Super.Eid = Emp.Superid
```

Actual Table in DB is Child Table ->Called Emp

Eid	Ename	SuperId
1	Ahmed	2
2	Ali	1
3	Mona	1
4	Hala	Null

New Table in ram in DB is Parent Table -> Called Super

Eid	Ename	SuperId
1	Ahmed	2
2	Ali	1
3	Mona	1
4	Hala	Null

Note :

- How Can U Set As Owner For Database:

  Right Click On You Db->Properties->Choose General->Set Owner As “Sa”

- Number Of Joins Less Than 1 From Number Of Tables Using In Joins

2. Queries :

♣ Isnull(Expression, Replacement_Value):
- Return The Specified Value If The Expression Is Null, Otherwise Return The Expression
- Ex : Select Isnull(St_Fname,'') From Students

♣ Coalesce (*Val1*, *Val2*, ..., *Val_N*)
- Return The First Non-Null Value In A List:
- Ex : Select Coalesce(St_Fname,St_Lname,St_Address, 'No Data') From Students

♣ Convert()
- Converts A Value (Of Any Type) Into A Specified Datatype.
- **Syntax** : Convert(Data_Type(Length), Expression)

♣ Concat() :
- Adds Two Or More Strings Together.
- **Syntax** : Concat(String1, String2, ..., String_N)
- If Any String Is Null Replace It And Set Is As Empty String

♣ Like Operator
- Is Used In A Where Clause To Search For A Specified Pattern In A Column.
- There Are Two Wildcards Often Used In Conjunction With The Like Operator:
  - % Represents Zero, One, Or Multiple Characters
  - The Underscore Sign _ Represents One, Single Character.
- **Syntax** :

```
Select Column1, Column2, ...
From Table_Name
Where Column Like Pattern;
```

- Some Patterns :
  - 'A%H' : String Start With ‘A’ And End With ‘H’
  - ‘%A_’ : String Before Last Char Is ‘A’
  - ‘Abd%’ : String Start With ‘Abd’
  - ‘[Ahm]%’ : String Start With ‘A’ Or ‘H’ Or ‘M’
  - ‘[^Ahm]%’ : String Not Start With ‘A’ Or ‘H’ Or ‘M’
  - ‘[A-H]%’ : String Start With Rang ‘A’ To ‘H’
  - ‘[^A-H]%’ : String Not Start With Rang ‘A’ To ‘H’
  - ‘[(am)(gh)]%’ -> string start of am or gh
  - ‘%[%]’ : String End With ‘%’
  - ‘%[_]%’ : String Contain With Char ‘_’
  - ‘[_]%[_]’ : String Start With ‘_’ , End With [_]

♣ Order By :

- Order By [Column Name]
- Order By [# Col] : Order By With [# Col] Column In Select

3. Database Normalization:

- ✓ **Normalization**: The Process Of Structuring Data To Minimize Duplication And Inconsistencies(التناقضات).
- ✓ The Process Usually Involves Breaking Down A *Single Table* Into Two Or Tables And Defining More Relationships Between Those Tables.
- ✓ **Normalization** Is Usually Done In Stages, With Each Stage Applying Some Rules To The Types Of Information Which Can Be Stored In A Table.

♣ Well-Structured Relations

- Goal Is To Avoid Anomalies(الشذوذ)
  - Insertion Anomaly : Adding New Rows Forces User To Create Duplicate Data
  - Deletion Anomaly : Deleting Rows May Cause A Loss Of Data That Would Be Needed For Other Future Rows
  - Modification Anomaly : Changing Data In A Row Forces Changes To Other Rows Because Of Duplication

Note : A Table Should Not Have More Than One Entity Type

- **Example :**

SID	Sname	Bdate	City	ZipCode	Subject	Grade	Teacher
1	Ahmed	1/1/1980	Cairo	1010	DB	A	Hany
1	Ahmed	1/1/1980	Cairo	1010	Math	B	Eman
1	Ahmed	1/1/1980	Cairo	1010	WinXP	A	khalid
2	Ali	1/1/1983	Alex	1111	DB	B	Hany
2	Ali	1/1/1983	Alex	1111	SWE	B	Heba
3	Mohamed	1/1/1990	Mansoura	1210	NC	C	Mona

Question—**What's** The Primary Key? **Answer**—Composite: Sid ,Subject

Why Do These Animalizes Exist?

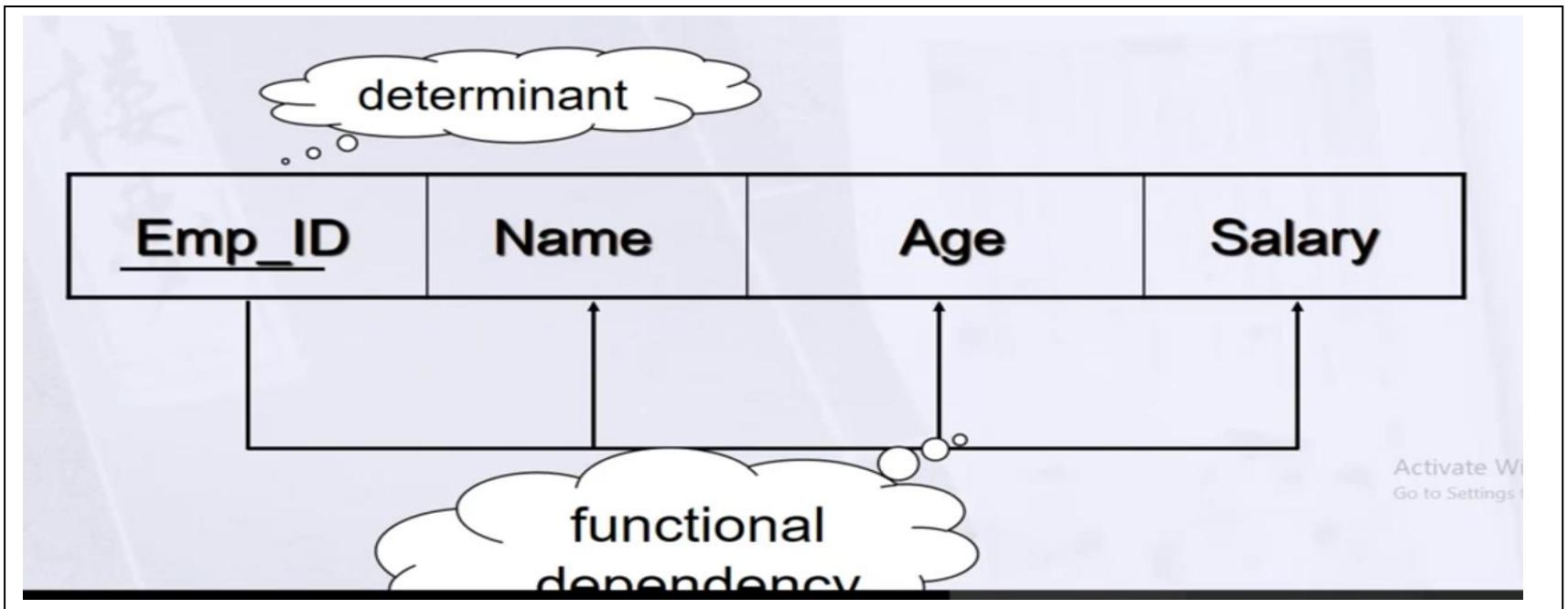
Because We've Combined Two Themes (Entity Types) Into One Relation. This Relation Duplication, And An Unnecessary Dependency Between Two Entities.

1.1 Functional Dependency :

- A Constraint Between Two Attributes (Columns) Or Two Sets Of Columns
- A -> B If "For Every Valid Instance Of A, That Value Of A Uniquely Determines The Value Of B"
- Or A -> B If "Existing Of B Depending On A Value Of A"
- **Some Examples :**
  - Social Security Number Determines Employee Name
    - SSN -> Ename
  - Project Number Determines Project Name And Location
    - Pnumber -> {Pname, Plocation}
  - Employee SSN And Project Number Determines The Hours Per Week That The Employee Works On The Project
    - { SSN , Pnumber } -> Hours

♣ keys and dependencies

EMPLOYEE (Emp_ID, Name, Age, Salary)



♣ Types of functional dependency

❖ Full Functional Dependency :

Attribute is fully Functional Dependency on a PK if its value is determined by the whole PK

- EX : { SSN , Pnumber } -> Hours

❖ Partial Functional Dependency:

Attribute if has a Partially Functional Dependency on a PK if its value is determined by part of the
PK(Composite Key)

- EX : SSN -> Ename

❖ Transitive Functional Dependency :

Attribute is Transitively Functional Dependency on a table if its value is determined by anther non-key attribute which it self determined by PK

- EX : SSN -> Pnumber
  Pnumber -> PName

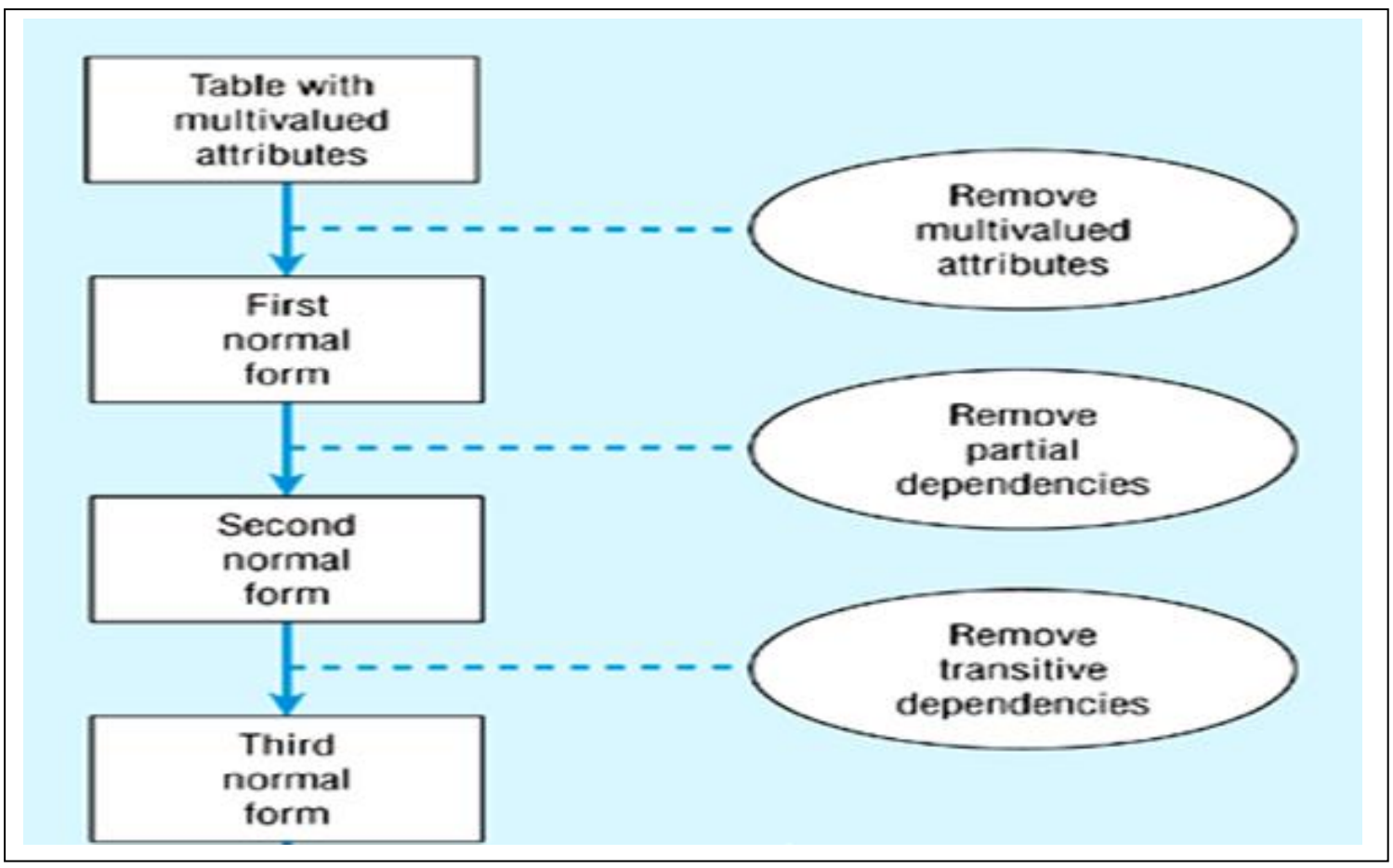
SNN -> PName

Example :

SID	SName	Birthdate	City	Zip Code	Subject	Grade	Teacher
1	Ahmed	1/1/1980	Cairo	1010	DB	A	Hany
1	Ahmed	1/1/1980	Cairo	1010	Math	B	Eman
1	Ahmed	1/1/1980	Cairo	1010	WinXP	A	khalid
2	Ali	1/1/1983	Alex	1111	DB	B	Hany
2	Ali	1/1/1983	Alex	1111	SWE	B	Heba
3	Mohamed	1/1/1990	Cairo	1010	NC	C	Mona

- Full Functional Dependency :
  - Sid,Subject -> Grade
- Partial Functional Dependency :
  - Sid -> SName
  - Subject -> Teacher
- Transitive Functional Dependency:
  - ZipCode -> City

♣ Steps in normalization :



➢ INF

- relation is in first normal form if it contains no multivalued or composite attributes
- remove repeating groups to a new table as already demonstrated, "carrying" the PK as a FK
- All columns (fields) must be atomic

- Means : no repeating items in columns

SID	SName	Birthdate	City	Zip Code	Subject	Grade	Teacher
1	Ahmed	1/1/1980	Cairo	1010	DB	A	Hany
					Math	B	Eman
					WinXP	A	khalid
2	Ali	1/1/1983	Alex	1111	DB	B	Hany
					SWE	B	Heba
3	Mohamed	1/1/1990	Cairo	1010	NC	C	Mona

Repeating Groups
Or multivalued

Student(SID, Sname, Birthdate, City, Zip Code)

SID	SName	Birthdate	City	Zip Code
1	Ahmed	1/1/1980	Cairo	1010
2	Ali	1/1/1983	Alex	1111
3	Mohamed	1/1/1990	Cairo	1010

Stud_Subject (SID, Subject, Grade, Teacher)

SID	Subject	Grade	Teacher
1	DB	A	Hany
1	Math	B	Eman
1	WinXP	A	khalid
2	DB	B	Hany
2	SWE	B	Heba
3	NC	C	Mona

➢ 2NF

- a relation is in second normal form if it is in first normal form AND every non key attribute is fully functionally dependent on the primary key
- i.e. remove **partial functional dependencies**, so no non key attribute depends on just part of the key

Student(SID, Sname, Birthdate, City, Zip Code)

SID	SName	Birthdate	City	Zip Code
1	Ahmed	1/1/1980	Cairo	1010
2	Ali	1/1/1983	Alex	1111
3	Mohamed	1/1/1990	Mansoura	1210

Stud_Subject (SID, Subject, Grade)

SID	Subject	Grade
1	DB	A
1	Math	B
1	WinXP	A
2	DB	B
2	SWE	B
3	NC	C

Subject (Subject, Teacher)

Subject	Teacher
DB	Hany
Math	Eman
WinXP	khalid
SWE	Heba
NC	Mona

➢ 3NF :

- 2NF PLUS no transitive dependencies(one attribute functionally determines a second, which functionally determines a third)

Student(SID, Sname, Birthdate,)

SID	SName	Birthdate	ZipCode
1	Ahmed	1/1/1980	1010
2	Ali	1/1/1983	1111
3	Mohamed	1/1/1990	1010

Stud_City(City, Zip Code)

City	Zip Code
Cairo	1010
Alex	1111

Stud_Subject (SID, Subject, Grade)

SID	Subject	Grade
1	DB	A
1	Math	B
1	WinXP	A
2	DB	B
2	SWE	B
3	NC	C

Subject (Subject,Teacher)

Subject	Teacher
DB1	Hany
Math	Eman
WinXP	khalid
DB2	Hany
SWE	Heba
NC	Mona

Day 04 :

1. Aggregation Function :

♣ Define :

- is a function that performs a calculation on a set of values, and returns a single value.
- are often used with the GROUP BY clause of the SELECT statement.
- The GROUP BY clause splits the result-set into groups of values and the aggregate function can be used to return a single value for each group.

Note :

- Aggregate functions ignore null values(except for COUNT()).
- If u use another columns when u using Aggregate Fun , u should use group by using another columns

  Ex :

```
SELECT AVG(st_age), st_address, dept_id
FROM students
GROUP BY st_address, dept_id;
```

  Note : Groups the data by **unique combinations** of st_address and dept_id, so the average age is calculated per address and department.
- WHERE applied for rows , HAVING applied on Aggregation group
-

♣ Examples:

- MIN() :returns the smallest value within the selected column
- MAX() : returns the largest value within the selected column
- COUNT() : returns the number of rows in a set
  - Count(*) → Cal number of rows
- SUM() : returns the total sum of a numerical column
- AVG() : returns the average value of a numerical column
- HAVING filters groups based on aggregated values.

  Ex :

```
SELECT Department, COUNT(*) AS EmployeeCount
FROM Employees
GROUP BY Department
HAVING COUNT(*) > 10;
```

♣ SQL Subqueries :

A subquery is a SQL query nested inside a larger query.

- ✓ The subquery can be nested inside a SELECT, INSERT, UPDATE, or DELETE statement or inside another subquery.
- ✓ A subquery is usually added within the WHERE Clause of another SQL SELECT statement.
- ✓ You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, or ALL.

- ✓ A subquery is also called an inner query or inner select, while the statement containing a subquery is also called an outer query or outer select.
- ✓ The inner query executes first before its parent query so that the results of an inner query can be passed to the outer query.

You can use a subquery in a SELECT, INSERT, DELETE, or UPDATE statement to perform the following tasks:

- Compare an expression to the result of the query.
- Determine if an expression is included in the results of the query.
- Check whether the query selects any rows.

Syntax :

```
SELECT select_list
FROM table
WHERE expression operator (SELECT select_list FROM table);
```

- The subquery (inner query) executes once before the main query (outer query) executes.
- The main query (outer query) use the subquery result.

2. Union Family :

➢ Union && Union all :

The UNION operator selects only distinct values by default and order values.

To allow duplicate values, use UNION ALL:

Syntax :

```
SELECT column_name(s)
FROM table1
UNION ALL
SELECT column_name(s)
FROM table2;
```

```
SELECT column_name(s)
FROM table1
UNION
SELECT column_name(s)
FROM table2;
```

➢ Intersect :

is used to retrieve the common records **between** two **SELECT queries.**

Syntax :

```
SELECT column1 , column2 ….
FROM table1
WHERE condition
INTERSECT
SELECT column1 , column2 ….
```

FROM table2
WHERE condition

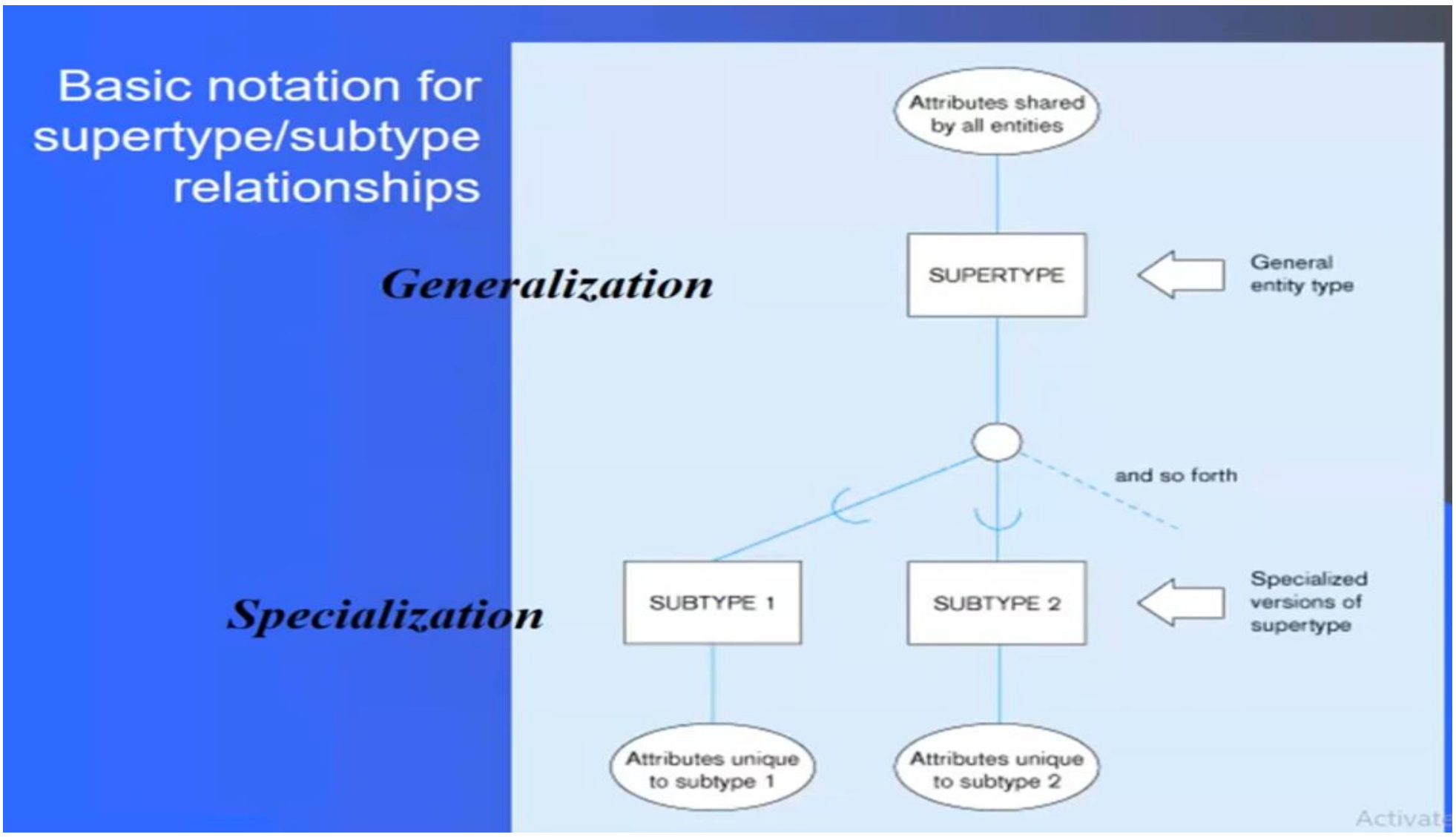
- **Except :**
  - is used to return the rows from the first **SELECT statement** that are not present in the second SELECT statement
  - **Syntax :**

```
SELECT column1 , column2 ….
FROM table1
WHERE condition
Except
SELECT column1 , column2 ….
FROM table2
WHERE condition
```

3. Enhanced ERD (EERD) :

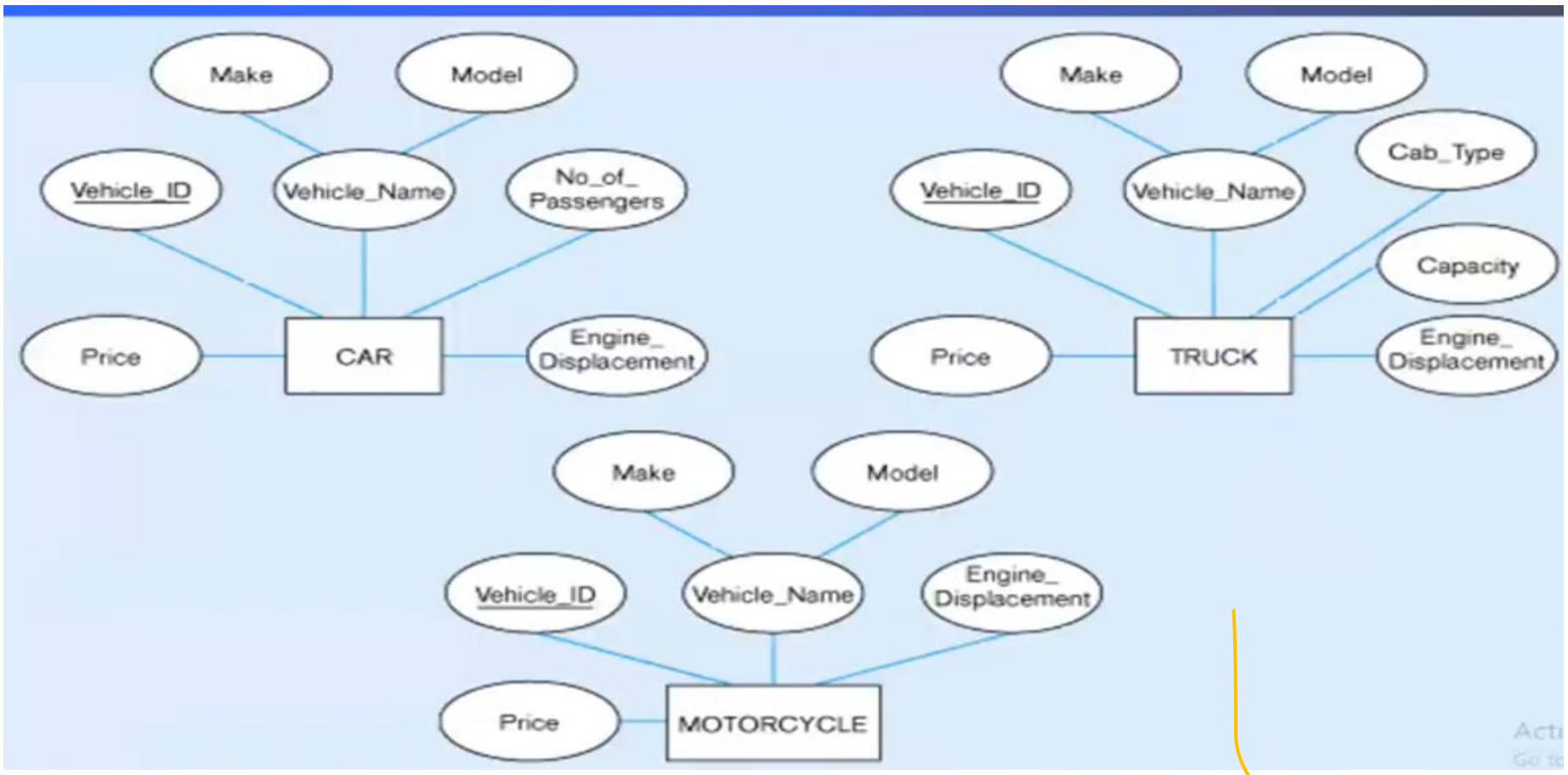
➢ Supertypes and Subtypes

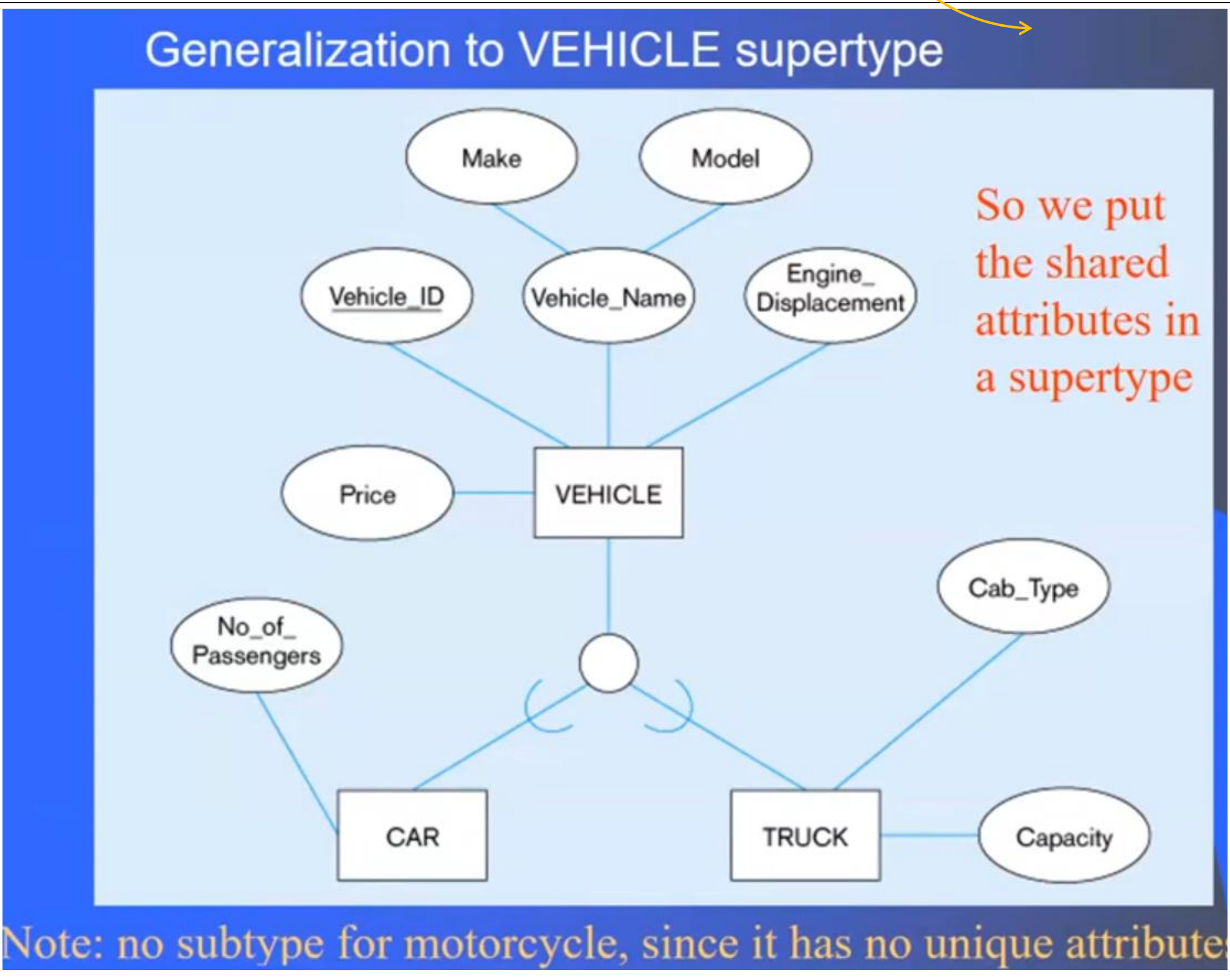
- ***Subtype**: A subgrouping of the entities in an entity type which has attributes that are distinct from those in other subgroupings*
- ***Supertype**: An generic entity type that has a relationship with one or more subtypes*
- ***Inheritance**:*
  - *Subtype entities inherit values of all attributes of the supertype*
  - *An instance of a subtype is also an instance of the supertype*



Example of generalization :

1. *Three entity types: CAR, TRUCK, and MOTORCYCLE*

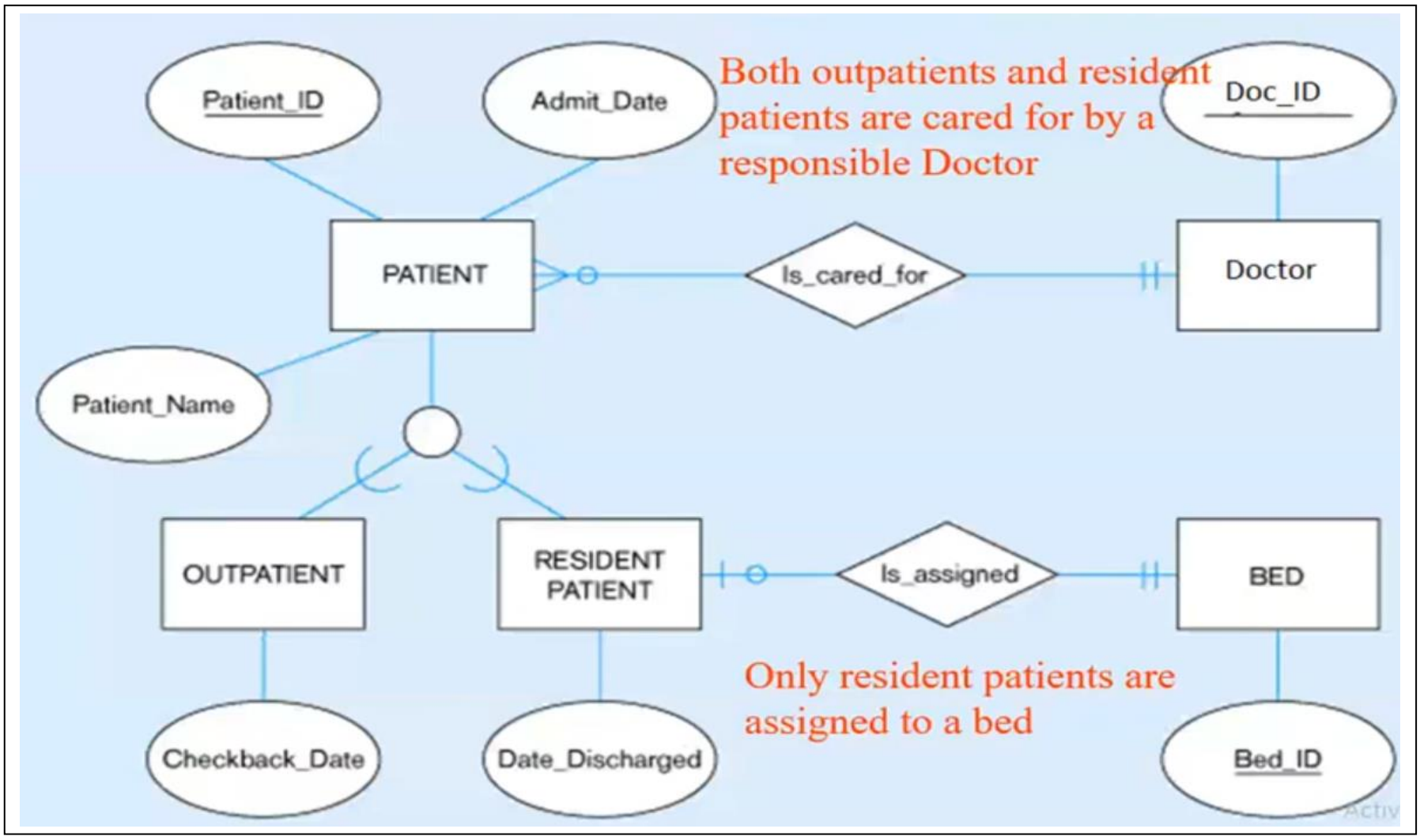




➢ *Relationships and Subtypes*

- *Relationships at the supertype level indicate that all subtypes will participate in the relationship*
- *The instances of a subtype may participate in a relationship unique to that subtype. In this situation, the relationship is shown at the subtype level*

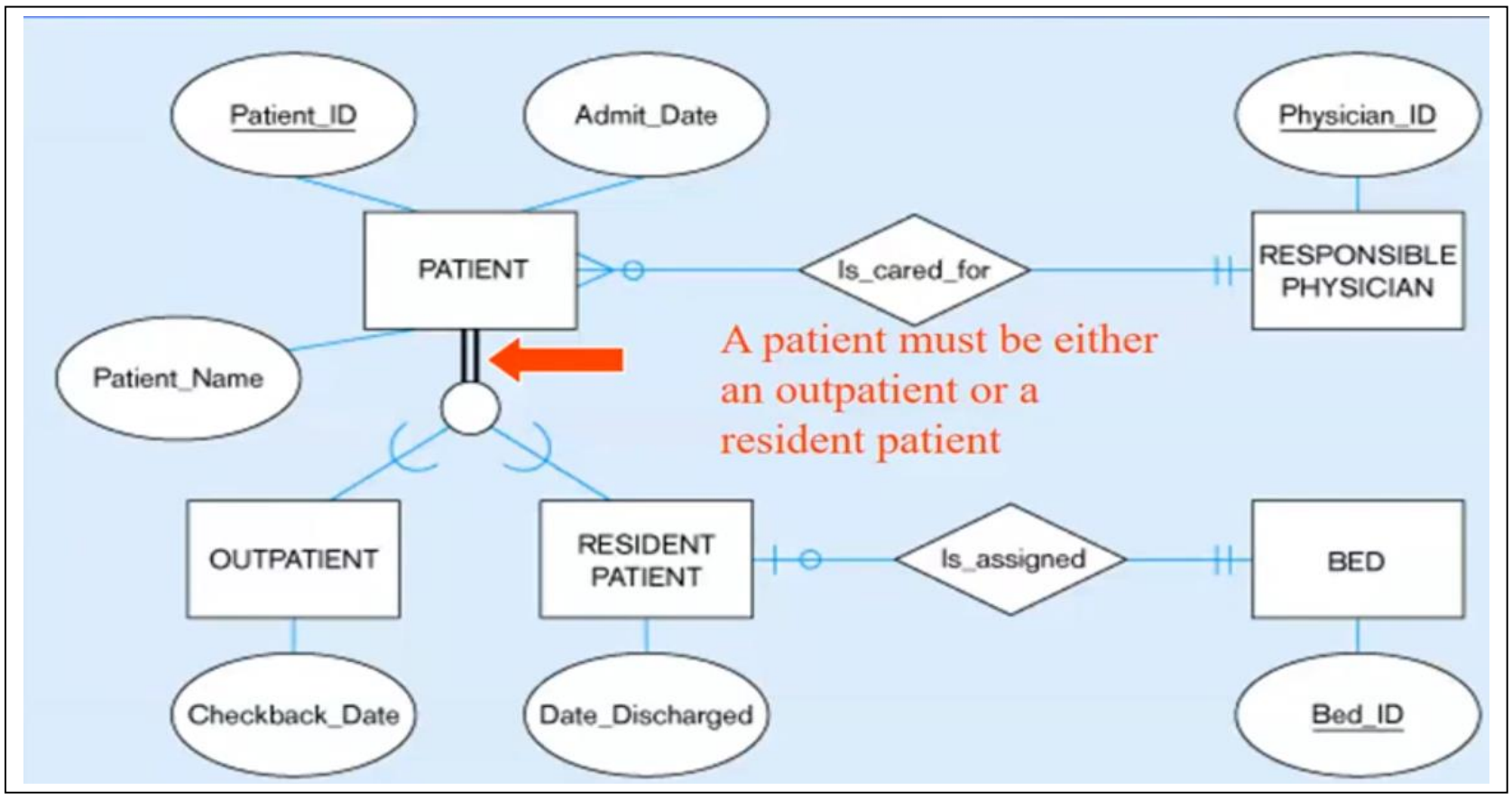
Example : Supertype/subtype relationships in a hospital



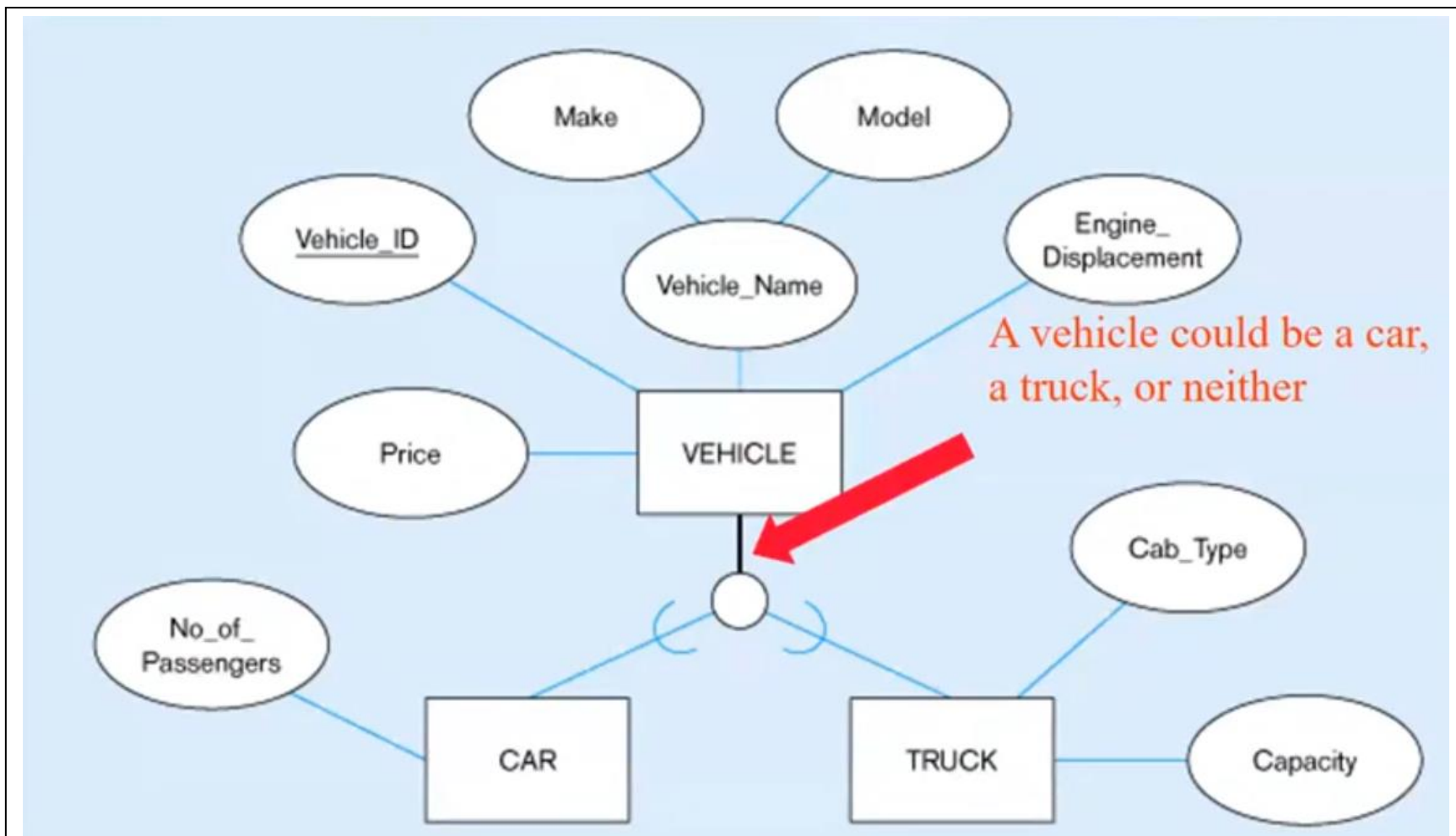
➢ *Constraints in Supertype*

- Completeness Constraints:
  - Total Specialization Rule: Yes (double line)
  - Partial Specialization Rule: No (single line)
- Disiointness Constraints:
  - Total Disjoint (d)
  - Overlap Rule (o)

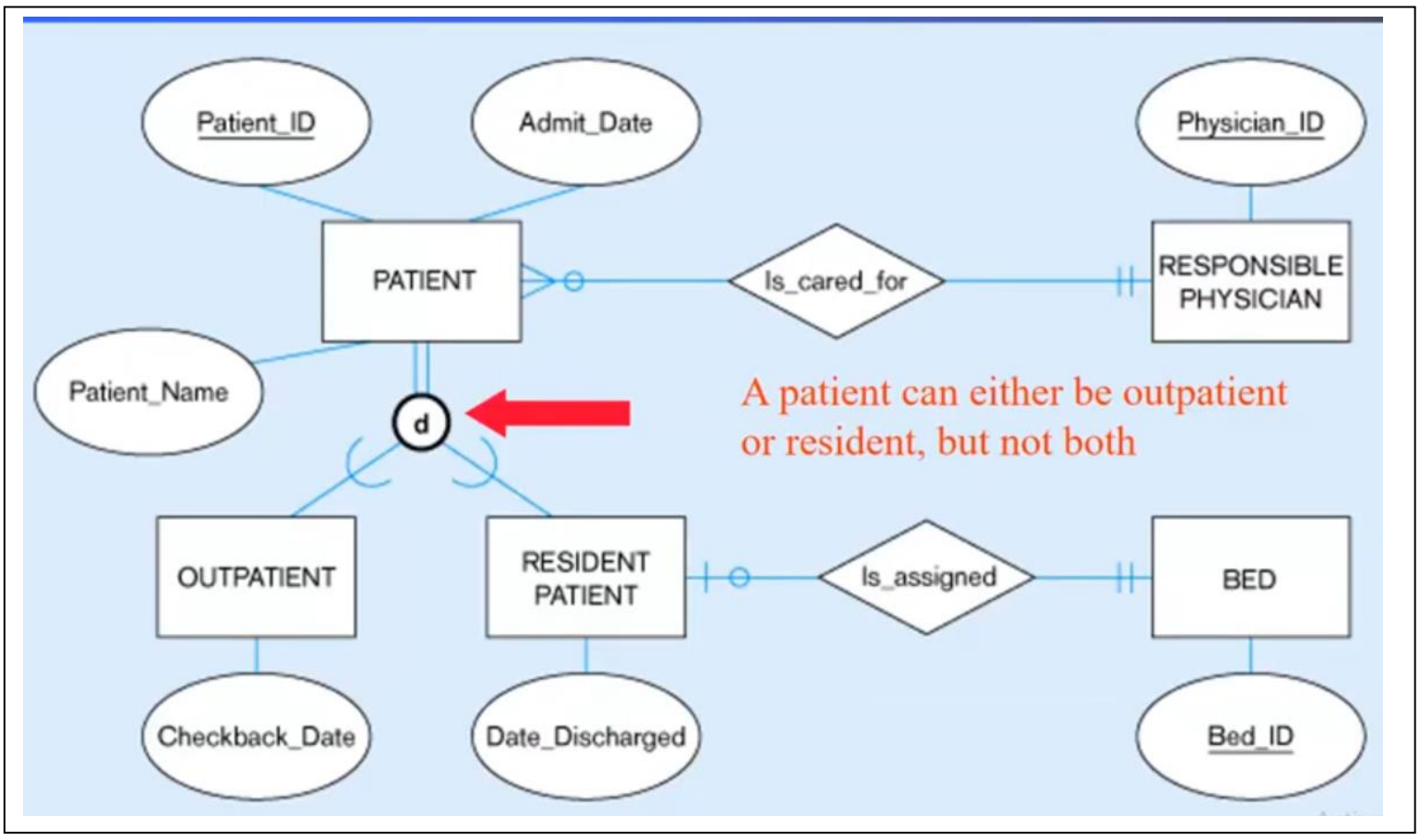
Example : Examples of completeness constraint Total specialization rule



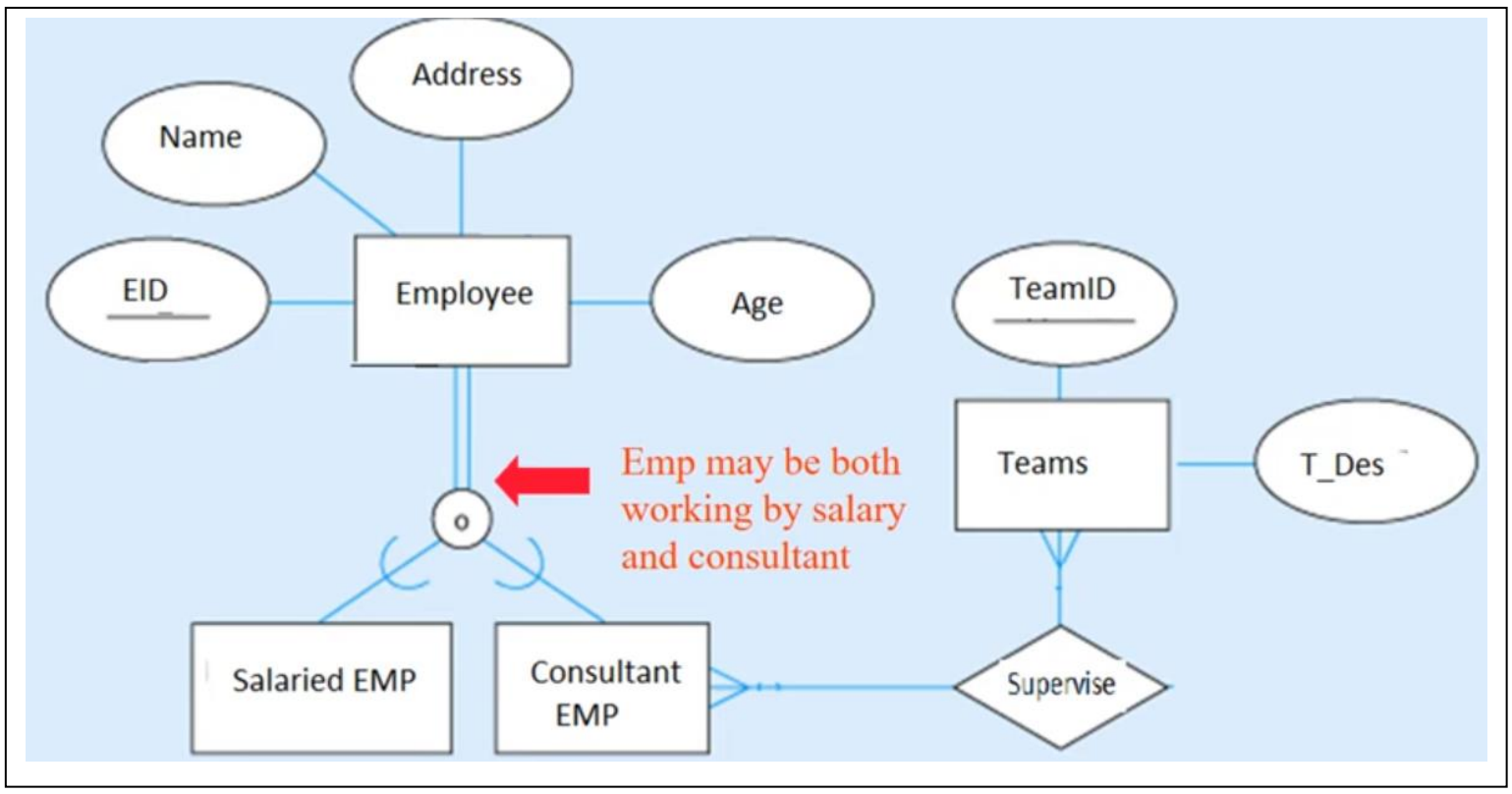
Example : Partial specialization rule :



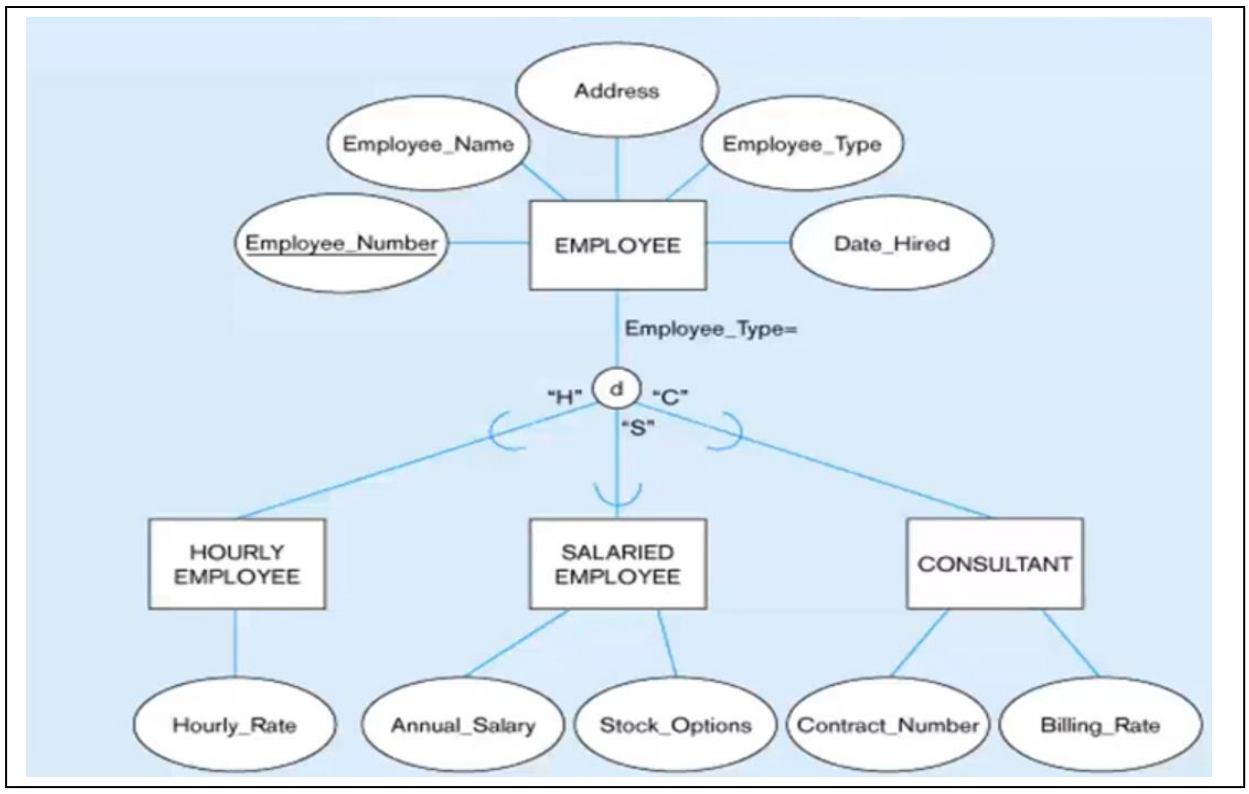
Example : disjointness constraints

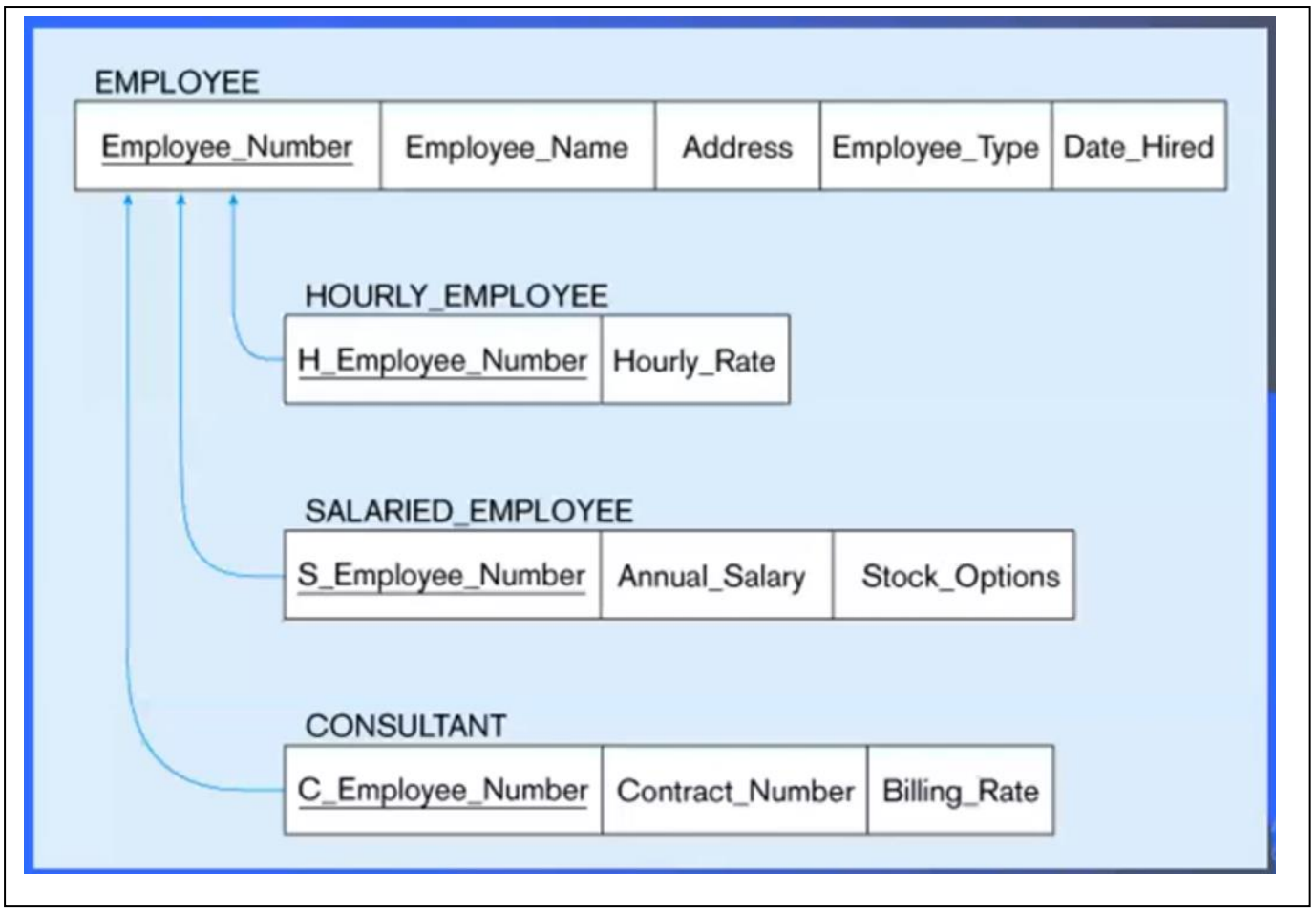


Example : Disjoint rule

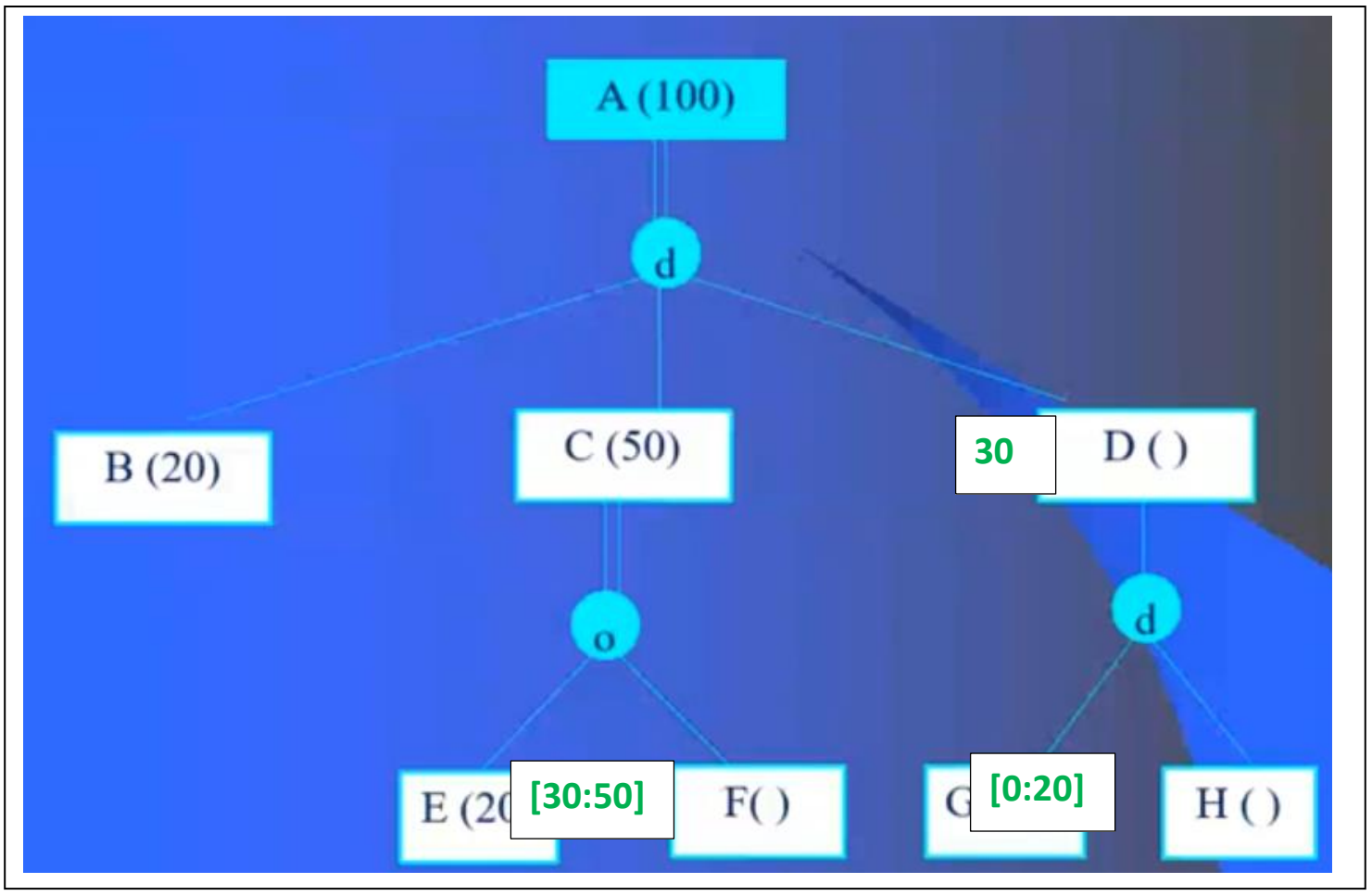


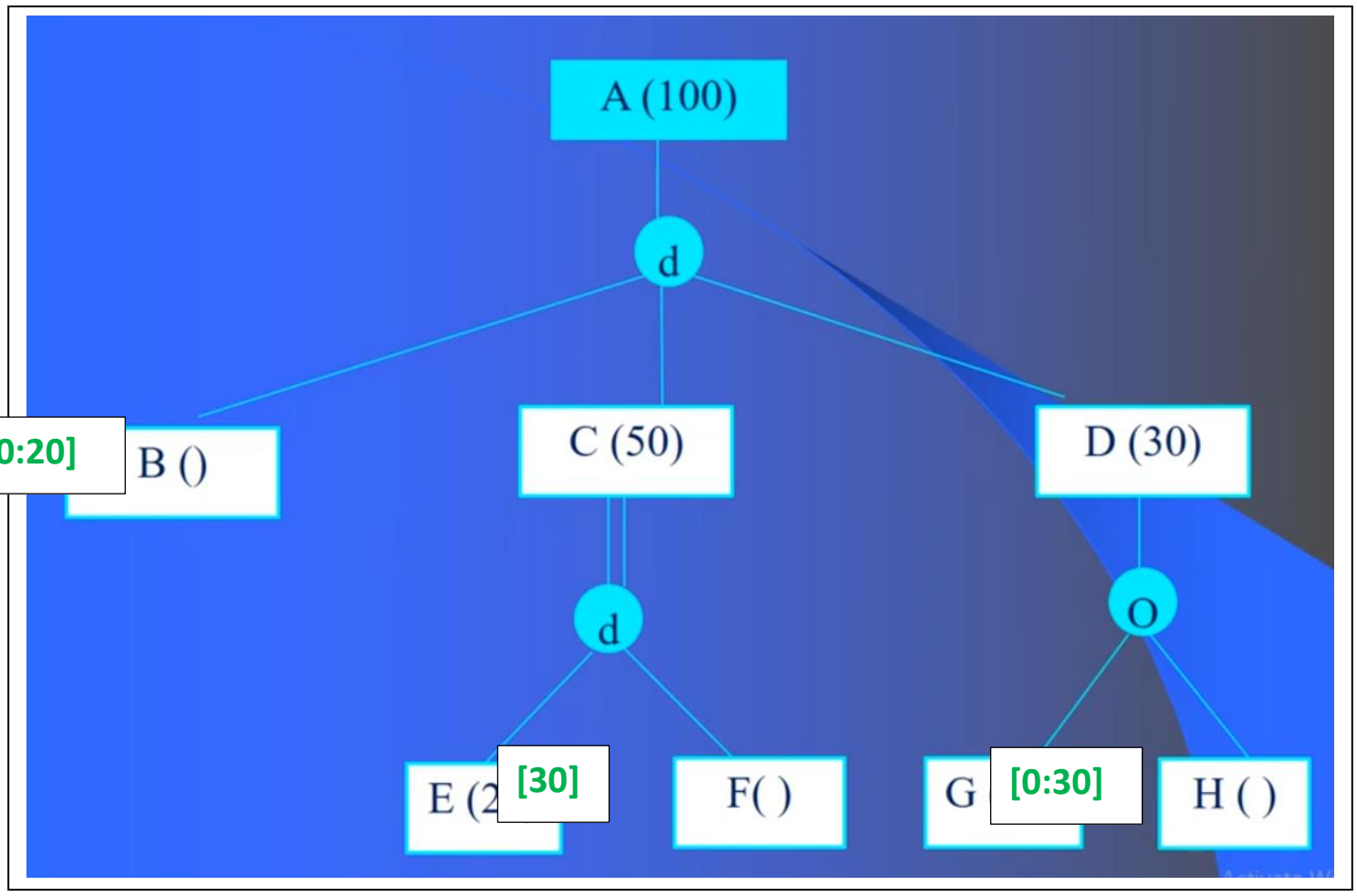
➢ *Mapping :*


Address
Employee_Name
Employee_Type
Employee_Number
EMPLOYEE
Date_Hired
Employee_Type=
"H"
d
"C"
"S"
HOURLY EMPLOYEE
SALARIED EMPLOYEE
CONSULTANT
Hourly_Rate
Annual_Salary
Stock_Options
Contract_Number
Billing_Rate


EMPLOYEE
Employee_Number
Employee_Name
Address
Employee_Type
Date_Hired
HOURLY_EMPLOYEE
H_Employee_Number
Hourly_Rate
SALARIED_EMPLOYEE
S_Employee_Number
Annual_Salary
Stock_Options
CONSULTANT
C_Employee_Number
Contract_Number
Billing_Rate

Examples :





4. Built in Function :

➢ *Aggregation Fun :*

- getdate() - isnull() - coalesce
- concat - Convert - year(getdate())
- substring(st_fname,1,3) : from index 1 to index 3
- db_name() : know name of db.
- suser_name() : know of user database

Note :

- Know any syntax for any thing in sql server open : view -> Template Explorer

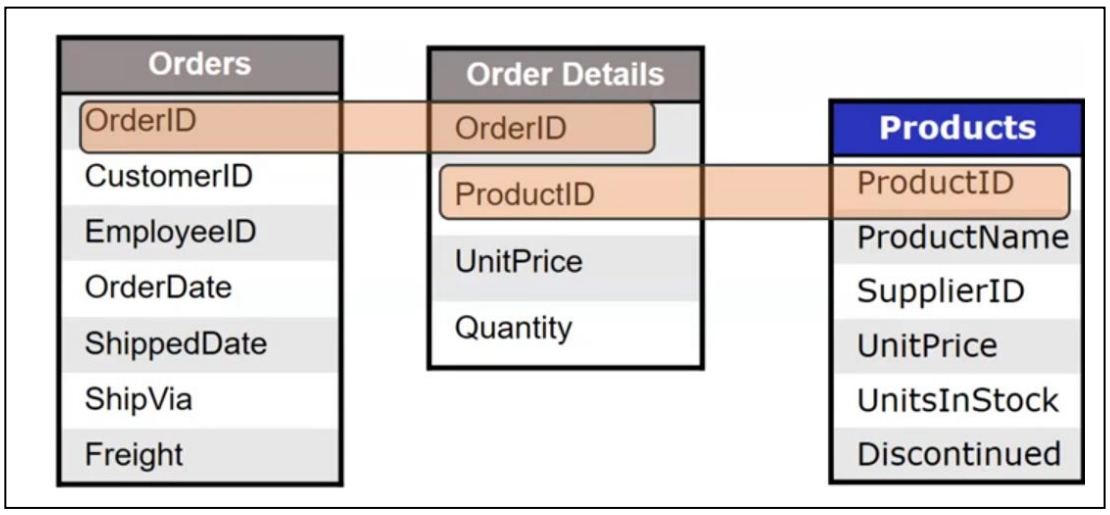
Day 05 :

1. Installation

- ✓ Writing Queries Using Microsoft SQL Server T-SQL
- ✓ Implementing a Microsoft SQL Server Database
- ✓ Maintain Microsoft SQL Server Database
- ✓ SQL Server Business Intelligence

2. Overview of Relational Databases

- ✓ SQL Server is a Fully RDBMS
- ✓ The tables have one-to-many relationships



3. SQL Server Versions History

1st Generation	
SQL Server Version	Features
SQL 6.0/6.5 (1995)	First version designed specifically for Windows NT Replication
SQL Server 4.2 (1992)	Developed for Windows NT 3.1
SQL Server 1.0 (1989)	Developed by Microsoft, Sybase, and Ashton-Tate for OS/2

2nd Generation	
SQL Server Version	Features
SQL2000	Focus on Performance and Scalability XML support Data Mining Reporting Services
SQL Server 7.0 (1999)	Restructure of Relational Server Data Transformation Services Online Analytical Processing

3rd Generation	
SQL Server Version	Features
SQL 2014,2017,2019	Security&Performance
SQL 2012	Always On Power View File Table Sequence Data Quality Service
SQL2008/SQL2008 R2	Power Pivot Enhance SharePoint Integration T-SQL (Ranking, Merge, Output) Improve and enhance for BI Tools
SQL2005	High Availability(includes DB Mirroring) Security Enhancements (DB Schema) Integration Services SQLCLR XML and Web services supports

4. SQL Server Editions :

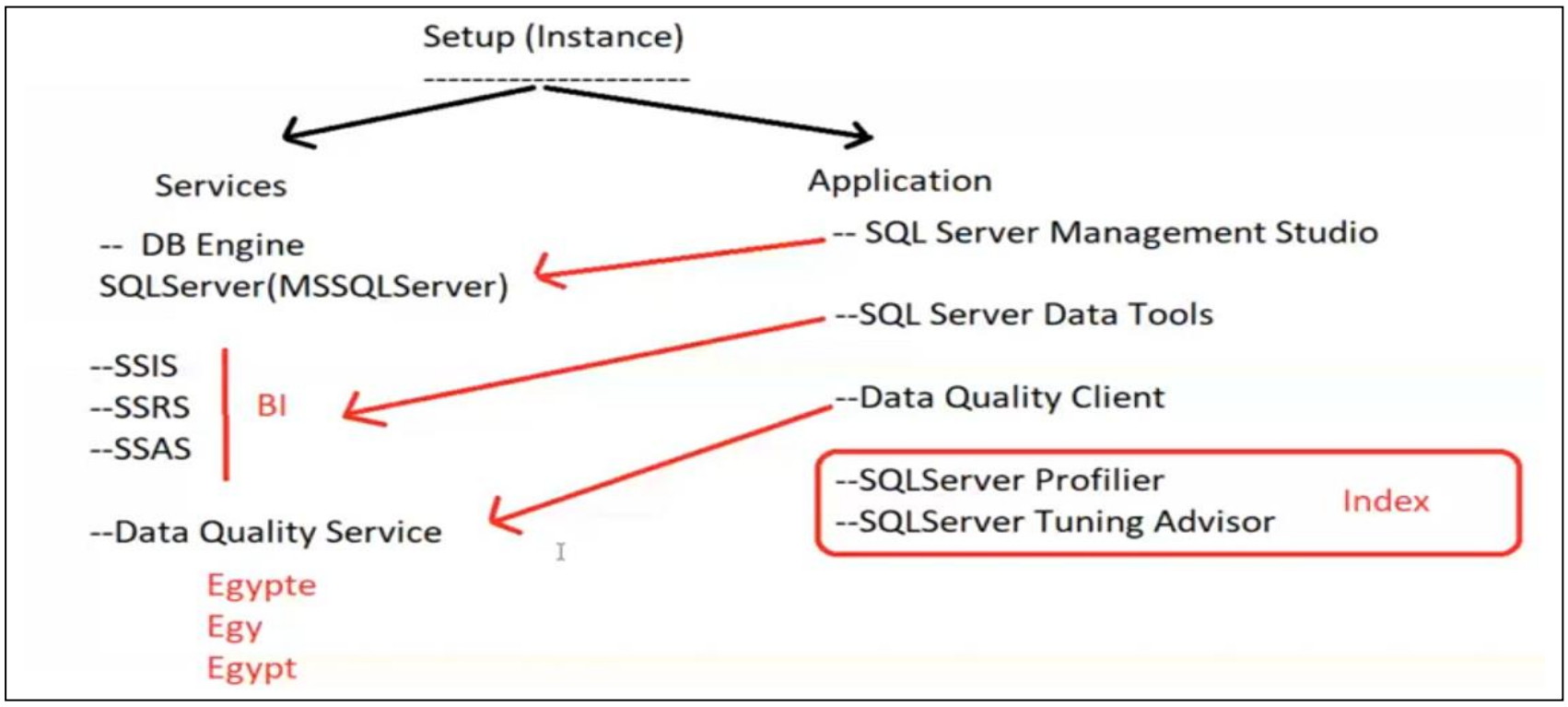
Edition	Description
Enterprise	For large scale, business-critical applications
Standard-Developer	For small/medium, departmental applications
BI Edition	For BI Services
Express	Entry level/learning edition
Azure	For Cloud

5. SQL Server Components

Server Component	Description
SQL Server Database Engine	Core service for storing and processing data
Analysis Services	Tools for creating and managing analytical processing
Reporting Services	Components for creating and deploying reports
Integration Services	Tools for moving, copying, and transforming data

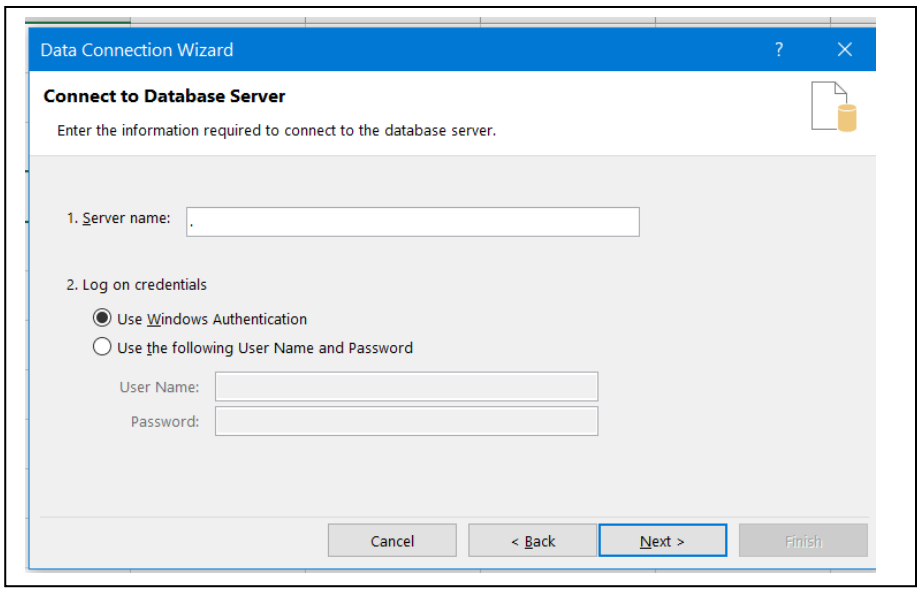
Data Quality Service & Master Data Service

6. Setup(instance):



7. Ways To Connect to Database :

- ♣ If u have many Instance (DB Engine)
  - Default Instance (DB Engine) :Service SQLServer(MSSQLServer)
    - ✓ Connect
      - .
      - Local
      - Pc-Name
      - IP (Current PC)
  - Named Instance (DB Engine) -> name “Cairo”
    - ✓ Connect
      - ./Cairo
      - local/Cairo
      - pc-Name/Cairo
      - IP/Cairo
- ♣ If u want to connect from Excel to Db server
  - Open Excel -> Data Tab -> Get External Data -> From Other Sources -> From SQL Server ->



♣ Connect within CMD :

Command :

- Connect using SQL Authentication :
  - sqlcmd –s [servername] –u [username] –p [password]
- Connect using Windows Authentication :
  - sqlcmd –s [servername]
  - sqlcmd –s [servername] –d [DBName] –q [your query]
- to run a query and output result in file
  - sqlcmd –s [servername] –d [DBName] –q [your query] –o [file name]

8. Security :

➢ *Authentication :*

♣ Authentication (User Name & Password)

--> Windows authentication

- Windows Admin ===> SQL Admin

--> SQLServer authentication

Username : password

- Dev_ahmed : 123
- Dev_ali : 444

♣ How to allow any person to remote my DB Server :

✓ Right click on server name -> properties -> Security tab -> Server authentication section -> **active** SQL Server and Windows Authentication mode

♣ How to create new login (new user) on SQL Server :

✓ Part “Security”-> click on it -> new login -> Create new user

♣ How to create new user on Database :

✓ Choose U DB -> open Folder “Security” -> click right on folder “User” -> choose new user

9. Some Queries :

1.Top(number) :

✓ The SELECT TOP clause is used to specify the number of records to return.

✓ Syntax :

```
SELECT TOP (number|percent) column_name(s)
FROM table_name
WHERE condition;
```

2.With ties :

✓ *Used when you want to return two or more rows that tie for last place in the limited results set.*

✓ **Examples :**

```
select top(8) with ties *
from Students
order by st_age
```

→ Return 9 rows

3.NewId():

✓ **Examples :**

```
--return random rows from table
    select *
    from Students
    order by newid()
```

4. SELECT INTO

- copies data from one table into a new table.
- **Example :**
  - ```
    select * into student
    from Students
    ```
  - Create new table in Db PhysioCare
    ```
    select * into [PhysioCare].[Actors].student
    from Students
    ```
  - To create empty table same original table
    ```
    select * into emptystudents
    from students
    where 1=2
    ```

5. Insert Based on select :

```
insert into emptystudents
select * from students
```

Note : columns in two tables must are matching

6. Having with only aggregation is special case :
   - **Example :**

```
select sum(salary)
 from Instructor
 having count(ins_id) > 100
```

10. execution order

1. from
2. join
3. on
4. where
5. group
6. having[aggregation]
7. select [distinct-aggregation]
8. order by
9. top

Examples :

1. Execution Correct :

```
select st_fname+' '+ st_lname as fullname
from Students
order by fullname
```

2. Execution Failed :

```
select st_fname+' '+ st_lname as fullname
from Students
where fullname = 'Ahmed Hassan'
```
```

11. DB Objects [table - view - function - Rule - SP] :

Defualt Path : [ServerName].[DBName].[schemaName].[objectName]

12. Ranking Function :

1. Row_Number()
2. Dense_rank()
3. NTiles(Group)
4. Rank()

Example :

```
Select *
From (
      Select * , Row _ Number()
      over(order by esal desc) as RN,
      Dense_rank() over(order by esal desc) as DR,
      Ntiles(5) over(order by esal desc) as G
      From employee
      ) as Newtable
where RN=1     RN=3
DR=1      DR = 2
G = 1
```

eid	ename	esal	did	RN	DR	G
15	ahmed	10000	10	1	1	1
14	ali	10000	10	2	1	1
12	eman	9000	10	3	2	1
1	nada	9000	10	4	2	1
2	reem	9000	10	5	2	1
3	khalid	8000	10	6	3	2
7	mohamed	7000	20	7	4	2
8	sayed	7000	20	8	4	2
6	hassan	6000	20	9	5	2
5	omar	6000	20	10	5	2
9	sally	5000	30	11	6	3
10	shimaa	4000	30	12	7	3
11	hana	4000	30	13	7	3
12	lama	3000	30	14	8	3

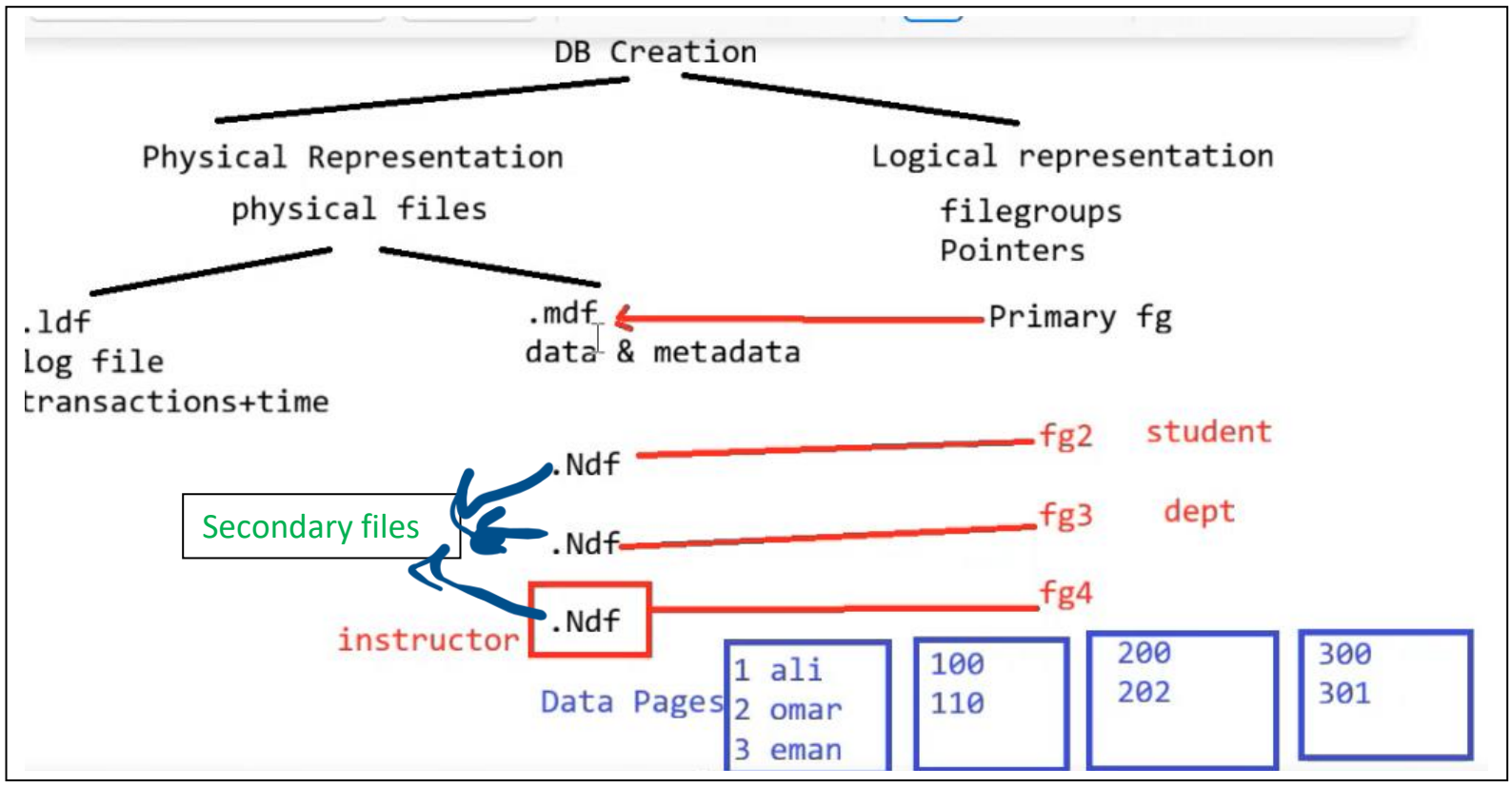
Example :

```
Select *
From (
Select * , Row _ Number()
      over(partition by did order by esal desc) as RN
, Dense_rank() over(partition by did order by esal desc)
as DR
From employee ) as Newtable
where RN=1     RN=3
DR=1      DR = 2
```

eid	ename	esal	did	RN	DR
15	ahmed	10000	10	1	1
14	ali	10000	10	2	1
12	eman	9000	10	3	2
1	nada	9000	10	4	2
2	reem	9000	10	5	2
3	khalid	8000	10	6	3
7	mohamed	17000	20	1	1
8	sayed	17000	20	2	1
6	hassan	6000	20	3	2
5	omar	6000	20	4	2
9	sally	15000	30	1	1
10	shimaa	4000	30	2	2
11	hana	4000	30	3	2
12	lama	3000	30	4	3

Day 06 :

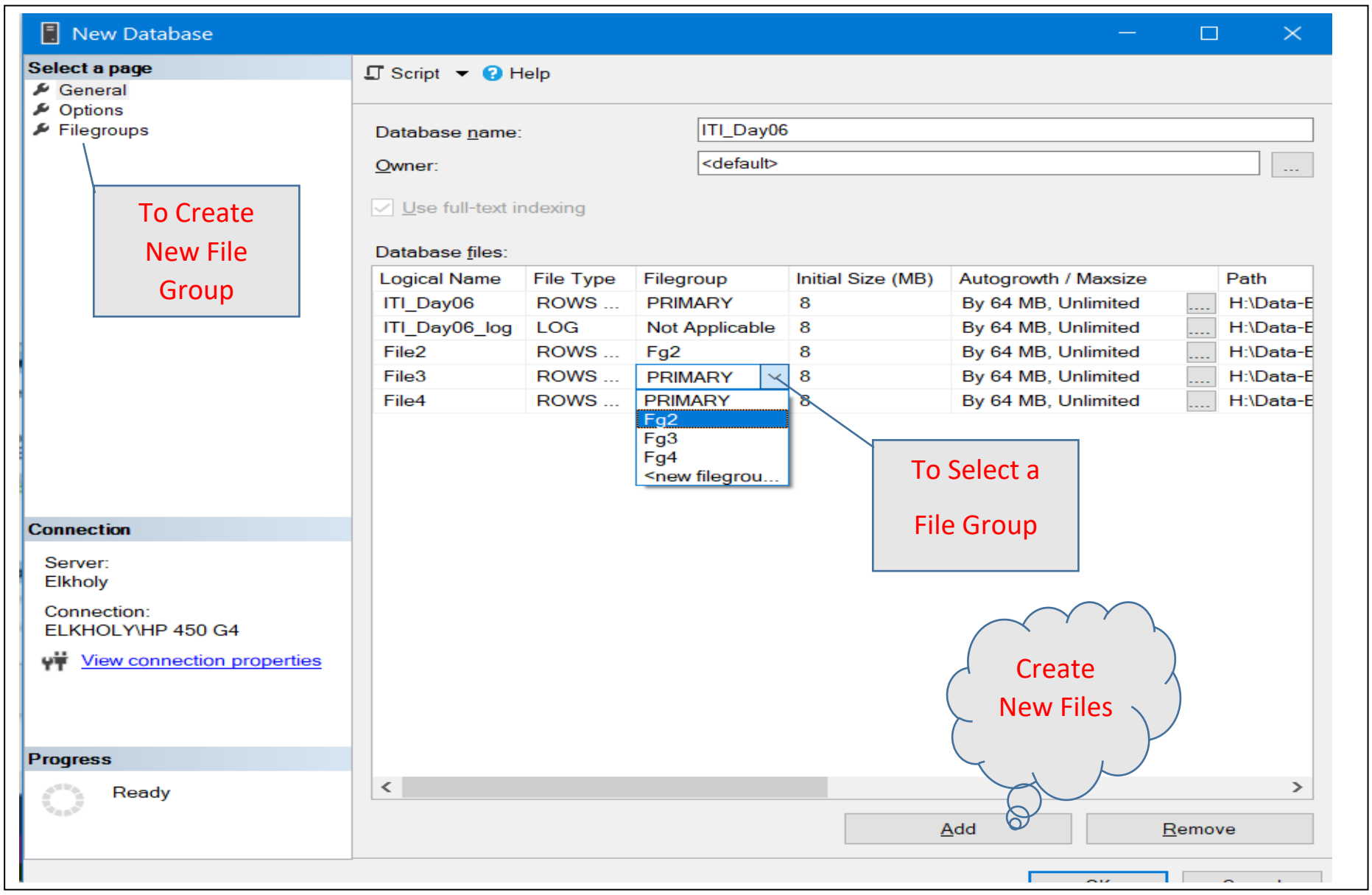
6.1 : DB Structure Files :



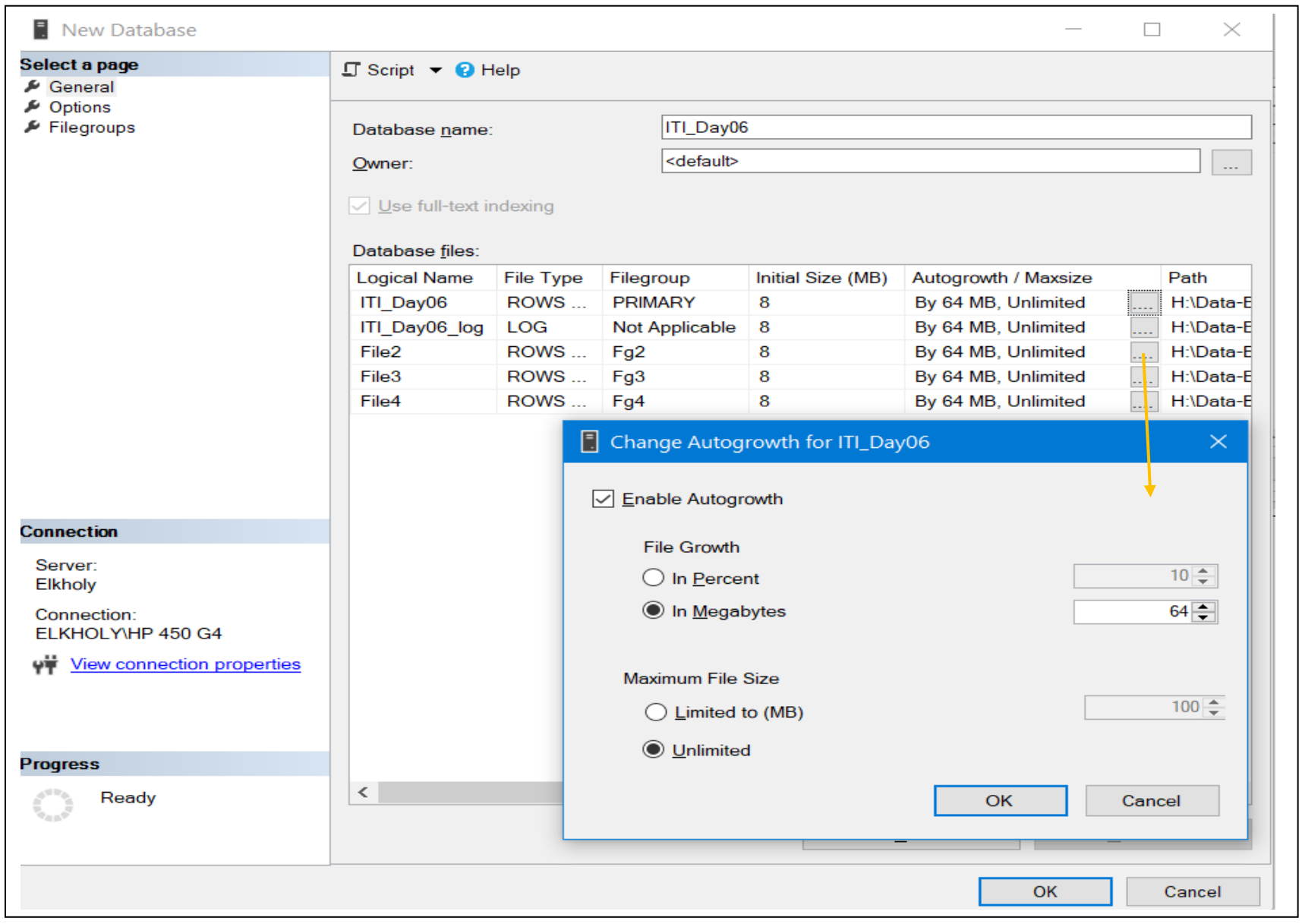
- **Tables :**
  a. store at MDF (Data & Metadata)
  b. store at data Pages(size = 8kb) sorted sequentially with primary key
  c. store default at file group called “Primary” ,I can change it later
- If I usually joins between tables , best practice store each table in different file group To improve reading , because it retrieve parallel
- File group is a pointer to refer to physical file on HDD

6.2 New DB :

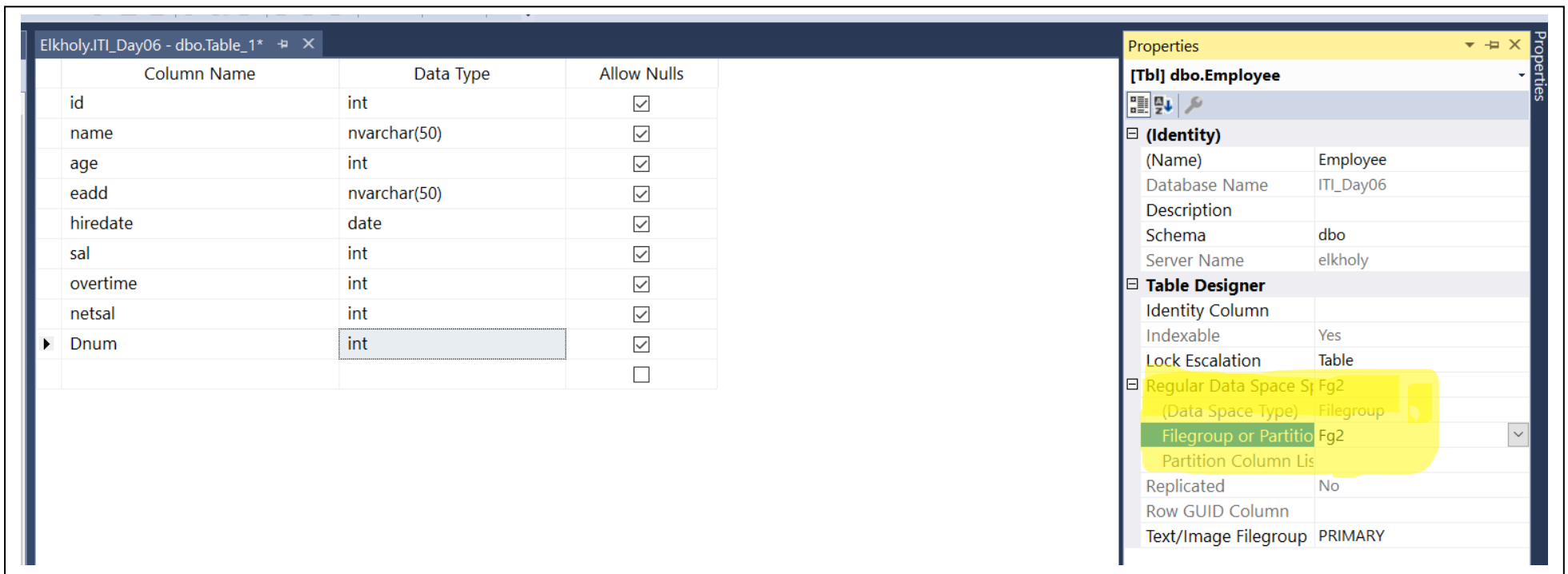
♣ Create Db File :



♣ Change Auto growth for ITI_Day06



♣ To change file group when u create new table :



6.3 Column and relationship Properties

♣ When I must create column Driven Attribute :

- If calculation for this is difficult and table more time
- If it use calculate another attribute
- If it affects performance

♣ Columns Properties :

- Computed Column Specification :
  - **Formula** : isnull(sal , 0)+ isnull(overtime , 0), each I call this computed column , it will call equation
  - **Is Persisted(yes/no)** : To store this Formula value on HD or no
- Is Sparse :
  - Set **yes** if this column take more null values , no store space in HDD
  - Set **no** if this column no take more null values
- Default Value or Binding : ex, “Menofia” , getdate()

♣ Relationships Properties :

- Insert & update specification :
  - **Delete Role :**
    - Set Null: When delete parent set null in foreign key
    - Cascade: when delete parent it will delete in child
    - Set default
    - No Action: I can`t delete parent if it have children
  - **Update Role :**
    - Set Null:
    - Cascade: when update in parent it will update in child
    - Set defual

6.4 Schema :

- Is a logical grouping for tables.
- Benefits :

1. اقدر اربط user ب schema كاملة
2. اقدر اخلي شوية tables في نفس ال schema

- Syntax :
  - Create : create schema [Name]
  - **Note :** Schema store in folder : Security->Schema
  - Transfer table in schema :
    alter schema [schema Name] transfer [Table Name]
  - Drop Scheme : alter schema [schema Name]

6.5 To Set Permission for a user :

2. Click on SQL server -> properties -> security tab -> SQL Server and Windows Authentication mode
3. Restart The Server

4. Open folder security overall Sql Server -> click right on login folder->new login -> create new user
5. Open folder security overall U DB -> click right on users folder-> new user -> enter name of user
6. Create Schema to assign permissions for a user
7. Set some of objects in this schema
8. To set permission :
   Open folder security overall U DB -> folder schemas->double click on u schema -> Permission Tab ->users or roles part -> click on search button -> browse button -> select user
9. Grant and deny for user : permissions for user part
10. Login using SQL authentication with this user
11. Run u queries

♣ To create a short cut for u table name
- Store at : Your Database -> Synonyms
- Examples :

```
create synonym HS
 for Hr.Students
 --
 alter synonym HS
 for Hr.Students
```

6.6 Delete & truncate & drop

- Drop : drop data and metadata
  - Ex : drop table course
- Delete :
  - Delete data only
  - May be use where condition
  - Slow than Truncate, As write in log file
  - Use rollback Concept
  - Don’t reset for identity
  - Ex : delete table course
- Truncate :
  - Delete data only
  - Don’t use where condition
  - faster than Delete, As may be write or not in log file
  - don’t use rollback Concept
  - Do reset for identity
  - Ex : truncate table course

6.7 : DB Integrity :

	Domain Integrity (Range of Values "Columns")	Entity Integrity (Uniqueness "Rows")	Referential Integrity (Relationships)
DB Constraint	-Datatype -Defualt Value -Allow null , not null -Check Constraint	-PK Constraint -unique constraint (allow one null value)	-foreign key constraint
DB Objects	-Rules -Triggers	-Index -Triggers	-Triggers

♣ DB Constraints :

1. Check Constraint.
2. Primary Constraint.
3. Unique Constraint.
4. Foreign Key Constraint.
5. Custom Constraint (use : Stored Procedure - Triggers).

Example :

1. **Constraints :**
   - Composite primary key :
     - constraint c1 primary key (eid,ename) ,
   - Primary key :
     - constraint c2 primary key (eid) ,
   - Foreign key :
     constraint c8 foreign key(did) references dept(deptid)
     on delete set NULL on update cascade
   - unique value (applied on rows) :
     - constraint c3 unique (eid) -> No Duplication eid
     - constraint c4 unique (eadd,ename) -> No Duplication eadd and ename together.
   - Check Value :
     - constraint c5 check(eadd in('cairo','mansoura','alex', 'menofia'))
2. **Rule :**
   - I u want to create :
     - constraint apply on new data only
     - constraint shared between two tables and columns
     - new datatype apply on it :constraint and defualt Val
   - Syntax :
     - create rule [rule name] as @x > 1000
   - Where created rule ?

- “Programmability 🗁” -> “Rules 🗁”
- To assign this rule to already column in table :
  - Sp_bindrule [rule name], ‘[table name].[column name]’
- To Un assign this rule to already column in table :
  - Sp_unbindrule ‘[table name].[column name]’
- To drop rule
  - Drop rule [rule name]

Note :

- Each column has only one rule ,col may have many constraints.
- Rule create overall database , while constraint create overall table
- Constraint not shared(specific),while rule is shared
- Role apply only new data(insert or update data) , while constraint apply on old and new data
- Constraint apply before rule
- I can I apply rule to new_Datatype, while constraint I can’t

3. Defualt :

- Syntax :
  - create default [default name] as Val
- Where created default ?
  - “Programmability 🗁” -> “Defualt 🗁”
- To assign this default to already column in table :
  - Sp_bindefault [defualtname],‘[table name].[column name]’
- To Un assign this default to already column in table :
  - Sp_unbindefault ‘[table name].[column name]’
- To drop default
  - Drop default [default name]

4. Create New DataType :

```
create rule r2 as @x>1000
create default d2 as 5000
sp_addtype ComplexDT ,'int'   -- ComplexDT inherit from integer
sp_bindrule r2,ComplexDT
Sp_bindefault d2,ComplexDT
```

- Where created newDataType ?
  - “Programmability 🗁” -> “Types 🗁” -> “User-Defined Data Types🗁”

Day 07 :

a. Variables :

1) Local Variable :
   - Declare Variable :
     - declare @[Name] [Datatype]
     - declare @x int = 100
   - Assign Value :
     - set @x = 50
     - select @x = 70
     - Select and update in one statement :
       update student set st_fname='ali' , @x = st_age where st_id = 1
   - To Print : select @x

Note :

- must declare , assign and select as a bash
- if query return value :

```
declare @x int = 200
select @x = st_Age from student where st_id = 1
select @x  -- print 20
```

- if select query does not return any value , variable keep last assign value in it

```
declare @x int = 500
select @x = st_Age from student where st_id = 10000
select @x  -- print 500
```

- if select query return an array , variable will take last value from this array

```
declare @x int = 500
```

```
        select @x = st_Age from student where st_address = 'cairo'
        select @x  -- print  21
```

- select two value form a query

```
    declare @y int, @name varchar(20)
  select @y=st_age ,@name=st_fname from student where st_id = 1
        select @y as age ,@name as name
```

- variable as an array (table) , this table in Memory

```
        declare @tbl  table(age int)
        insert into @tbl
        select st_Age from student where st_address = 'cairo'
        select * from @tbl
        --
        declare @x int = 5
        select top(@x) *
        from student
```

♣ **Dynamic Query :**

--==> **execute** : take string and convert to a query and try to run it

```
execute('select * from student')
--
declare @col varchar(20) ='*',@tbl varchar(20) ='student'
execute('select '+@col + ' from ' +@tbl)
```

2) Global Variable :

- can't declare Global Variable
- can't assign Global Variable
- Select with '**@@**'
- Examples :
  - Select @@servername -> To Return server name
  - Select @@rowcount -> To know number of rows was affected after last statement
  - Select @@version -> To Return latest version of SQL server
  - Select @@error -> To know type of error (0:no errors , another number: # of error)
  - Select @@identity -> To know last identity added in table , if your column is not identity it will return NULL

---

b. Controls Of flow Statement :

➢ If , begin , end

Example :

```
declare @x int
update student
      set st_age +=1
select @x = @@rowcount
```

```
if @x > 0  -- and , or
	begin
		select 'multi row affected'
	end
--else if
else
	begin
		select 'no row affected'
		end
```

- If exists , if not exists

Example :

```
if EXISTS (select name from sys.tables where name ='student1')
	begin -- return true
		select 'This table is exist'
	end
else
	begin
		create table studentss
		(
			id int,
			name varchar(50)
		)
		end

---
if not exists	(select dept_id from student where dept_id = 10)
  and not exists (select dept_id from instructor where dept_id = 10)
		begin
			delete from department where dept_id  = 10
		end
else
	begin
		select 'Table Department has a relationship with another table'
	end
```

sys.tables : get all tables in my db

- Try , catch

```
	begin try
	delete from department where dept_id  = 10
  end try
  begin catch
	select 'Table Department has a relationship with another table'
	select error_line() , error_number() , error_message()
  end catch
```

➢ While , continue , break

```
declare @i int = 1
while @i <= 10
        begin
                set @i +=1
                if @i = 5
                        continue
              if @i = 7
                        break
                select @i
end
```

------Search for :
--iff
--wait for
--choose

****

c. Batch V Script VS Transaction :

➢ batch :

-- some queries run together , not affected each other
--if statement not execute,if another statement is execute if true

Ex :

```
insert
update
delete
```

➢ script :

-- some queries not execute with other
-- Ex : create and drop table
-- to do this use keyword 'go' between statements

Ex :

```
create table
go
drop table
go
create rule
go
sp_bindrule
```

➢ Transa

Ex :

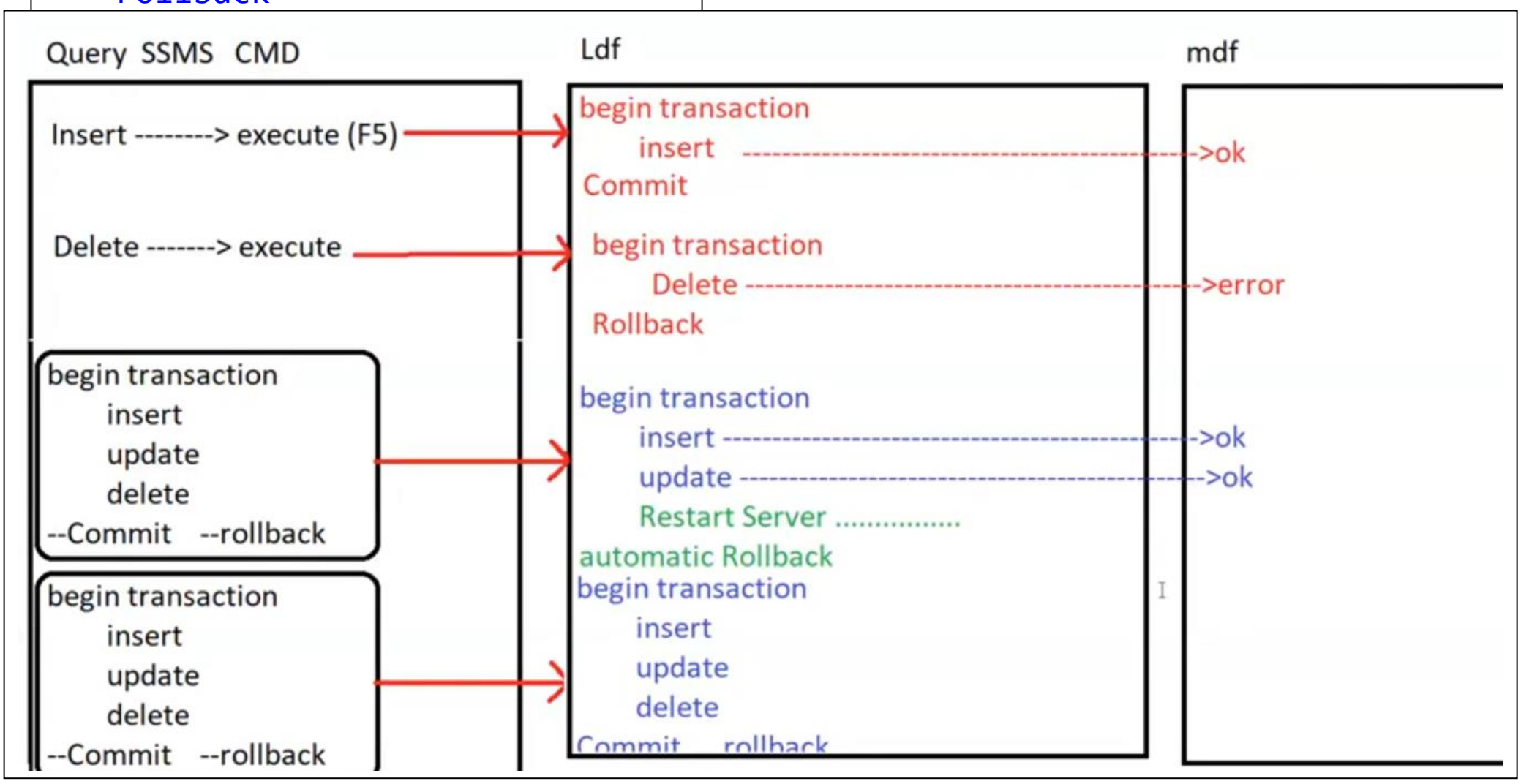
```
create table parent(pid int primary key)
create table Child(cid int foreign key references parent(pid))
```

```
begin transaction
    insert into Child values(1)
    insert into Child values(1)
    insert into Child values(4)
rollback
```

```
If happen error when inserting
, I will rollback , does not
insert in table
```

```
begin try
  begin transaction
    insert into Child values(1)
    insert into Child values(1)
    insert into Child values(40)
  commit
end try
begin catch
    rollback
```

I use try and catch to prevent error , when transaction is successful , it will add values in table . otherwise it will roollbaj



d. Functions :

Types :

1. Built in Function :

- Called Scaler Function : mean return one value
- Null : isnull() , Coalesce() , Nullif()
- System : db_name(), suser_name()
- Convert : Convert, cast ,format
- String : Substring ,Upper, lower, len

- Date : getdate(), year, Month ,Day
- Agg : Count, Max ,Min ,Avg ,Sum
- Math : power ,log ,sin ,cos ,tan
- ranking : Row_Number ,rank_Dense ,rank_Ntile
- Logical : iif , Choose
- Windowing: Lead ,Lag ,First_Value ,last_value

Note :

- **Function** must return value
- **Function** does not contain [insert – delete - update],only contain select query
- **Function** may be take parameter or no

2. User Defined Functions

♣ Scalar Function

- Return One Value
- When call it , u must write name of schema
- In Programmability 🗁 -> Functions 🗁 -> Scalar-valued Functions 🗁
- Ex :

```
create function getname(@id int)
returns varchar(20)
      begin
            declare @name varchar(20)
            select @name = st_fname from student  where st_id = @id
            return @name
      end

select dbo.getname(1)
```

♣ Inline table Function

- Return Table, if body contain only Select query ,like as **view**
- In Programmability 🗁 -> Functions 🗁 -> Table-valued Functions 🗁
- Ex :

```
CREATE FUNCTION GetInstructorsByDept (@did INT)
RETURNS TABLE
AS
RETURN
(
   SELECT ins_name, salary * 2 AS multiply_sal
   FROM Instructor
   WHERE dept_id = @did
```

```
);

select * from dbo.GetInstructorsByDept(10)
select ins_name from dbo.GetInstructorsByDept(10)
select sum(multiply_sal) from dbo.GetInstructorsByDept(10)
```

♣ Multi statement table valued Function

- Return Table, if body contain Select query with [if ,declare, while]
- Called : **insert based on select**
- **Ex :**

```
create function getSudents(@format varchar(20))
returns @t table ( id int , name varchar(20))
as
 begin
        if @format = 'first'
              insert into @t
              select st_id , st_fname from student
        else if @format = 'last'
              insert into @t
              select st_id , st_lname from student
        else if @format = 'full'
              insert into @t
              select st_id , st_fname + ' ' + st_lname from student
        return
 end

select * from getSudents('first')
```

Note : insert statement , for insert data in variable , not in table

Day 08 :

8.1 Index :

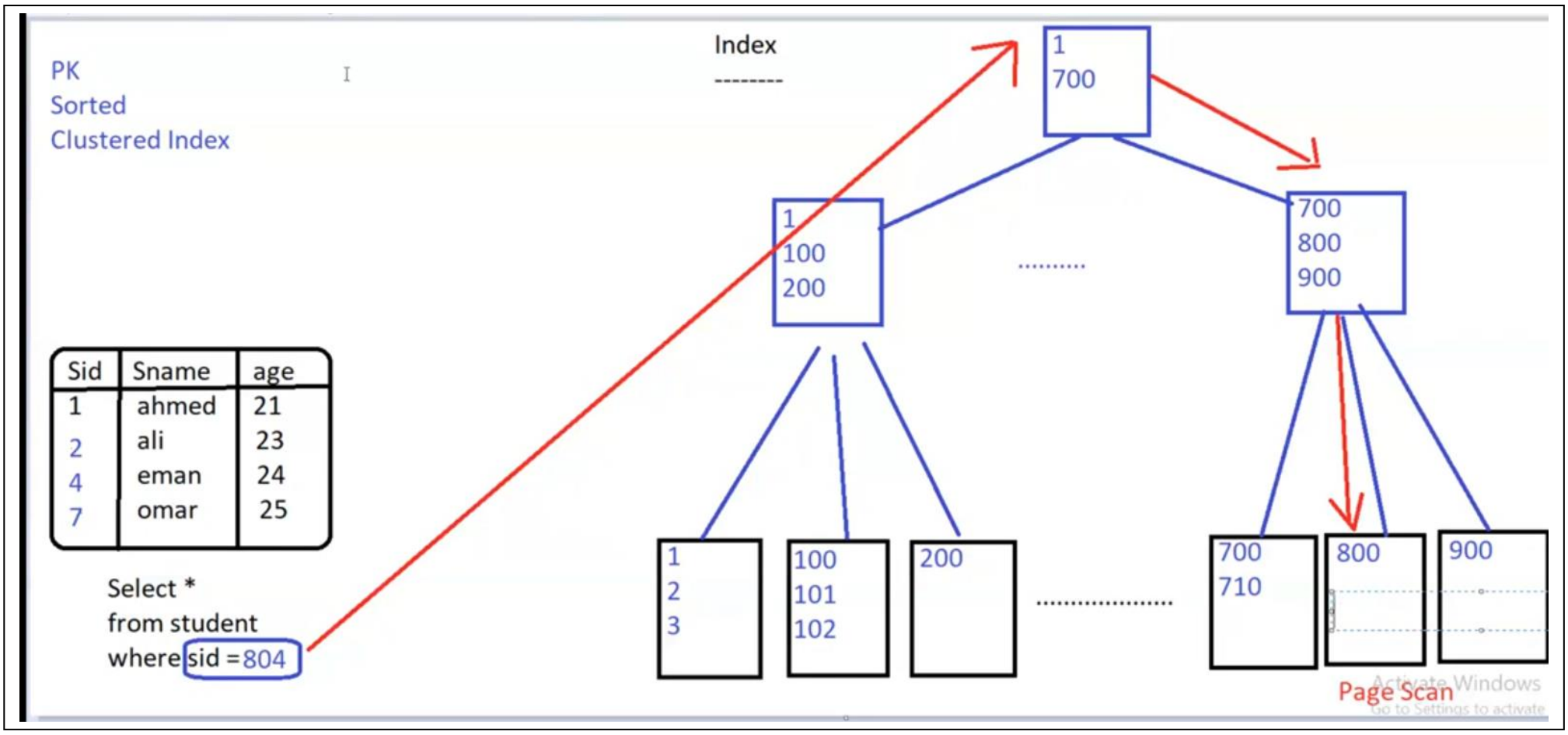
Note :

- When create table without primary key :
  - data is store in heap(data not sorted).
  - insert statement is fast ,as it will put data in last.
  - Table scan meaning when u searching for a data , it will scan all data in this table
- When create table with primary key :
  - data is store in u disk(data sorted with **PK**).
  - Insert is slow شوية

- There is a Clustered index(better for searching)
- Where : Columns 🗁 -> Indexes

8.1.1 Clustered Index :

- Cannot create more than one clustered index on table
- Data Pages : Clustered Index using PK

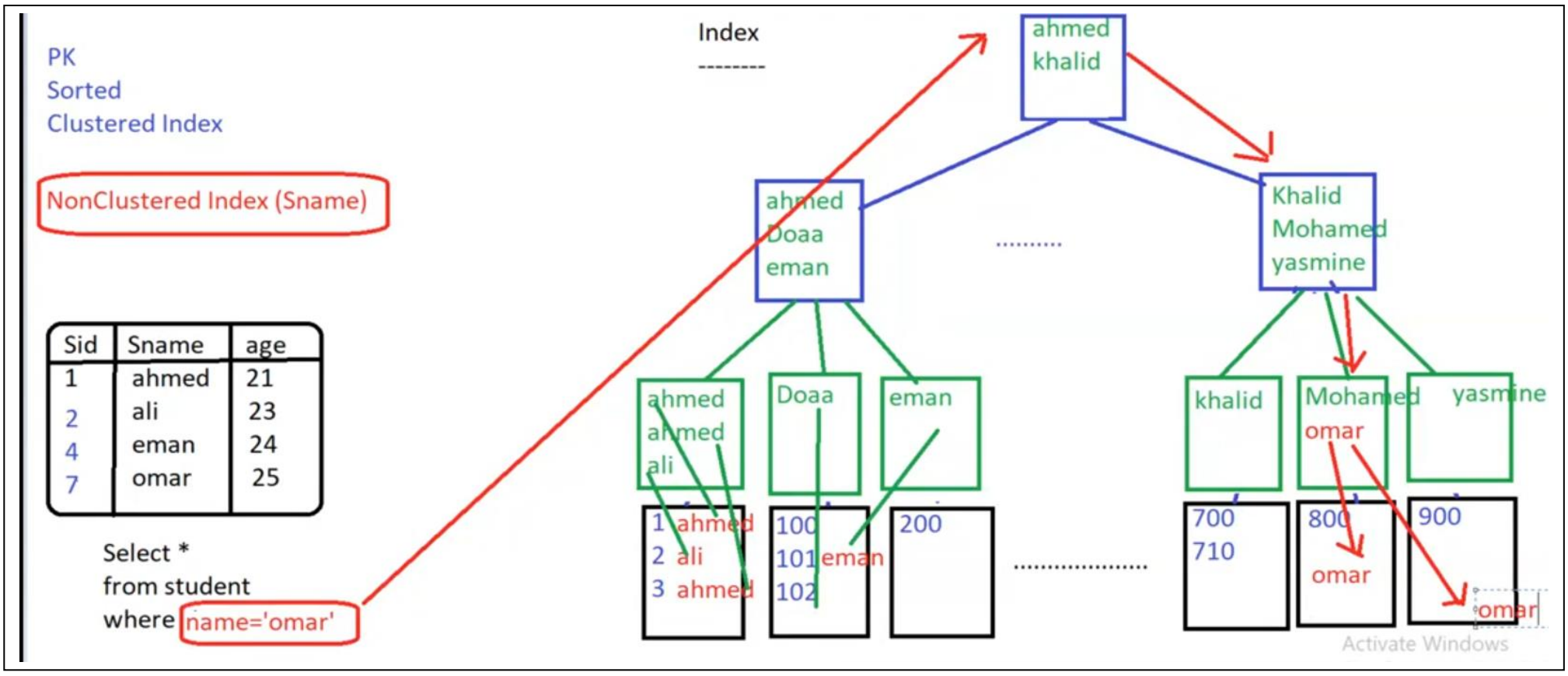


- Syntax :

```
create clustered index PK_Student
on student(st_id)
```

8.1.2 Non Clustered Index

- Data Pages : Non Clustered Index using SName



- Syntax :

```
create nonclustered index Search_BY_LName
on student(st_lname)
```

Note :

- PK is a Constraint , then create clustered index
- unique is a Constraint , then create nonclustered index

8.1.3 SQL Server Profiler :

- To open : start menu -> sql server profiler .

8.1.4 SQL Server Tuning :

- To open :
  - start menu -> sql server Tuning .
  - Tools -> sql server Tuning .
- pass for Tuning file.trc (Profiler) عشان تعرف مين اكتر كولم تقدر تعمل عليه index

8.2 System Database :

1. Master :

- Contain All Server`S Configuration Or Server`S Metadata
  Ex: Users Name , Users Password , Location Of My Own Dbs

2. Mode :

- Contain templates for any db is created take

3. msdb :

-

4. tempdb :

8.3 View :

- Is a select Statement
- Specify user View of data
- hide DB Objects
- Limit access of data
- Simplify Construction of complex queries
- has no Parameter
- has no **DML** queries inside its body
- standard view can be considered as Virtual table
- Only index view can increase Performance
- Folder : View 🗁
- Call : select * from [View Name]

Types of views :

1. Standard View :

Syntax :

```
Create view VCairo
As
      Select id ,name ,address
      From student
      Where address = 'cairo'
--
Create view VAlex
As
      Select id ,name ,address
      From student
      Where address = 'Alex
```

2. Partitioned View :

- Collect data from different server
- Syntax:

```
Create view VStuds
as
      select *
      from Mans Server.iti.dbo.students
      union all
      Select *
      From SohagServer.iti2.HR.studs
```

3. Indexed View : [Searching]

- [Used For Increase Performance of Queries]

Notes :

- To know code of view : sp_helptext '[view name]' , to encrypt U View , user "with encryption" ,
  Syntax :

```
create view [view name ]
```

```
with encryption
as
    select statement
```

- If u want to check for data when u insert it using view ,use “With check option"
  Syntax :

```
create view [view name ]
with encryption
as
    select statement
    with check option
```

- When I must write name of schema for object :
  1. Call Scaler Function
  2. Indexed View
  3. call object from another DB or server

DML With View :

1. One Table using in View :

- I can insert data in view table , if remain columns allow [identity – null – calculated(driven) – defualt value]

2. multiple Table using in View :

- U can not use delete
- U can use insert and update statement , must these statement affect only one table

8.4 Merge:

- **MERGE** statement to update data in a table based on values matched from another table.

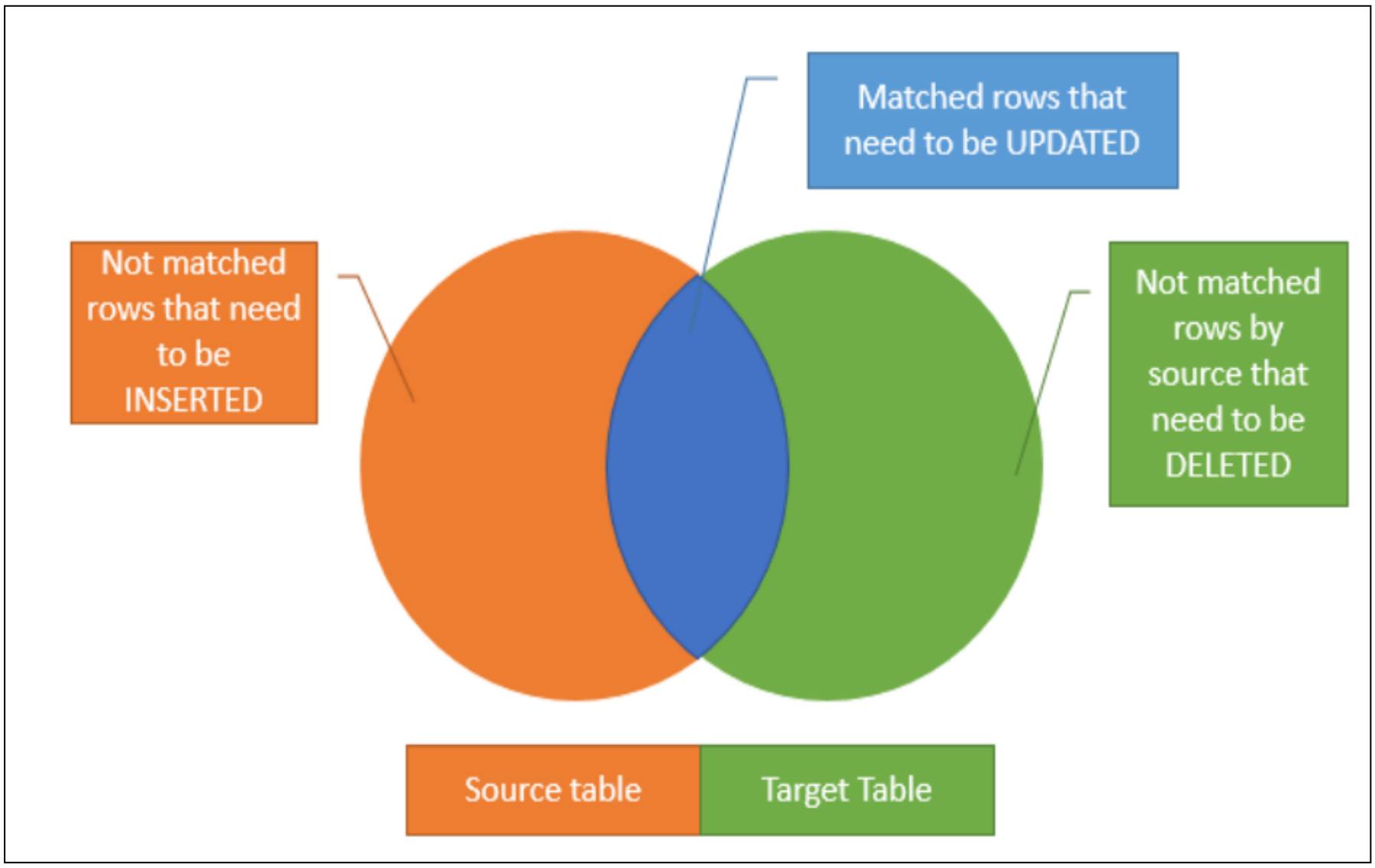
Introduction SQL Server MERGE Statement

Suppose, you have two table called source and target tables, and you need to update the target table based on the values matched from the source table. There are three cases:

1. The source table has some rows that do not exist in the target table. In this case, you need to insert rows that are in the source table into the target table.
2. The target table has some rows that do not exist in the source table. In this case, you need to delete rows from the target table.

3. The source table has some rows with the same keys as the rows in the target table. However, these rows have different values in the non-key columns. In this case, you need to update the rows in the target table with the values coming from the source table.

The following picture illustrates the source and target tables with the corresponding actions: insert, update, and delete:



Syntax :

```
    MERGE target_table
USING source_table
    ON merge_condition
    WHEN MATCHED
        THEN update_statement
    WHEN NOT MATCHED
        THEN insert_statement
    WHEN NOT MATCHED BY SOURCE
     THEN DELETE;
```

Full Example :

```
    MERGE sales.category t                          –- target table
        USING sales.category_staging s   -- source table
    ON (s.category_id = t.category_id)
    WHEN MATCHED
        THEN UPDATE SET                          -- update in target table
            t.category_name = s.category_name,
            t.amount = s.amount
    WHEN NOT MATCHED BY TARGET                   -- update in target table
        THEN INSERT (category_id, category_name, amount)
```

```
        VALUES (s.category_id, s.category_name, s.amount)
WHEN NOT MATCHED BY SOURCE
   THEN DELETE;                                  -- delete in target table
```

Day 09 :

9.1 Execution for a Query :

- Query -> Parsing (syntax) -> optimize (Metadata) -> Query Tree (from ->where ->select) -> Execution Plan (memory)

Note :

- I u execute the query multiple , it will go to execution query cycle

Problems for write Query :

- If u execute the query multiple , it will go to execution query cycle (Affect on performance)
- Network Wise :
  - كل لما حجم ال query كل لما عدد ال characters يكبر و ده بيأثر ع performance

9.3 Solution is Stored Procedure :

Cons :

- بتنادي ب اسم ال PROC فقط ولو بتاخد param بتحطه جنب الاسم
- مش باين للهاكر انت بتعمل (… - update – select – insert) , طبعا دا افضل من ناحية ال security
- انت خافي اسامي ال objects , طبعا دا افضل من ناحية ال security
- عدد ال Characters اقل بكتير من انك تكتب ال query (علي ال network)
- مع اول مرة وانت بتنادي ال PROC أي الي سيحصل :
  - بيمر بمرحلة ال execution query
  - لما يوصل لخطوة ال Query Tree بيعمل save داخلها, وبعدين بيعمل Execution Plan
- لو عملت call to PROC تاني مش هيكون مضطر انه يمشي كل خطوات ال execution query من الأول , هيروح يجيب ال query from query tree وبعدين ينفذها من Execution Plan

- إذا تمت إعادة تشغيل السيرفر، قد يتم **مسح ذاكرة التخزين المؤقت(Cache)** ، وبالتالي تحتاج بعض **Stored Procedures** إلى **إعادة التجميع (Recompile)** عند استدعائها لأول مرة بعد إعادة التشغيل.
- بتمنع ال errors انها تحصل علي السيرفر
- تقدر تخفي ال business role جوه ال PROC (مثل: معادلة حساب مرتب موظف)
- تقدر تكتب داخل الPROC :
(insert-update-delete-select-if-while-merge-try&catch)
- Programmability 🗁-> Stored Procedures🗁

=> Triggers (Special case from SP)

- Special of PROC
- Can`t Call it
- Can`t send parameter it
- is explicit code on server , listen on action(insert-update-delete) that happen on server
- triggers on table , and Db and server
- trigger called whether u query affected on rows or no
- trigger take name of schema automatic

Example :

```
1. Create trigger
create trigger t1
on student
after insert
as
	select 'Success Added'

--
create trigger t2
on student
for update
as
	select getdate()
--

create trigger t3
on student
instead of delete
as
	select 'not allowed for this user : ' + suser_name()
---- Set Table As ReadOnly
create trigger t4
on Department
instead of delete,insert,update
as
	select 'This table is readonly'

-- Use Update As a Function
alter trigger hr.t7
on hr.Students
after update
```

```
as
	if update(St_Age)
	select 'Not Allowed To increase age'

---- enable & disenable Trigger :
alter table department disable trigger t4
alter table department enable trigger t4

----------

Table inserted has history to inserting new data
Table deleted has history to deleting old data
create trigger t8
on course
after update
as
	select * from inserted
	select * from deleted
----------
--to prevent delete Friday
create trigger t6
on course
after delete
as
	if format(getdate(),'dddd') = 'friday'
	begin
		select 'Not Deleted'
		-- use : rollback
		-- or
		insert into course
		select * from deleted
	end
```

or :

```
--using instead of
create trigger t7
on course
instead of delete
as
	if format(getdate(),'dddd') <> 'friday'
	begin
		select 'allowed Deleted'
		delete from course where crs_id =( select crs_id from deleted)

	end

--------------------
```

```
--to store data about who update or delete row in my table :
create table history(
	_User varchar(50),
	_date date,
	_oldid int,
	_Newid int
)

create trigger t2
on topic
instead of update
as
	if update(top_id)
	begin
		declare @new int, @old int
		select @old = top_id from deleted
		select @new = top_id from inserted

		insert into history
		values (suser_name() , getdate() , @old , @new)
end
```

9.4 XML :

- Transform data from Db to another Db
- Keywords :
  --For XML raw
  --For XML auto
  --For XML Explicit
  --For XML Path

Type :

1. For XML :
    - Transform table to XML
2. open XML :
    - Transform XML to table

Day 10 :

Resources :

1. ITI Videos
2. Https://Aws.Amazon.Com/What-Is/Sql/
3. https://www.w3schools.com/sql/default.asp
4. https://www.programiz.com/sql/constraints
5. SubQueries
6. SQL Server MERGE