

Day1:

-----  
DB Life Cycle

DB Design

ERD

File Based System

DB Systems

Day2:

-----  
DB Mapping

DB Schema

SQL

Create DB

Day3:

-----  
joins

Normalization

Day4:

-----  
Aggregate Function  
grouping  
Union - Subqueries

EERD

Day5:

-----  
DB Engine  
Services  
Ranking Function

Transact-SQL

DB Course

-----  
Day6:

DB Constraints  
Create DB (SQLServer)  
Rules --Create DB

Day7:

-----  
Variables  
if /while  
functions

Day8:

-----  
View  
index  
Merger -Pivot tables

Day9:

-----  
Stored Procedures  
Triggers  
XML

Day10:

-----  
backup & restores  
Mirroring  
Cursor  
jobs  
SQLCLR

## DB Life Cycle

1- Analysis --> System Analyst

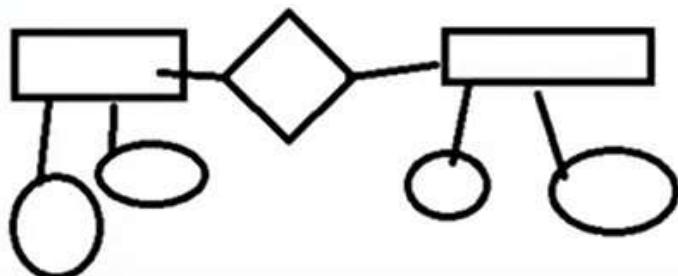
Scope (Req Doc)

HR System

2- DB Design --> DB designer

ERD

Entity Relationship Diagram



3- DB Mapping --> DB Designer

Set of Rules

Actual Schema

Tables

## 4- DB Implementation (shared DB)

Physical DB

Tool

RDBMS

Relational DB Management System

(SQL Server - oracle - mySQL --Access .....,)

SQL (Structure Query Lang)

DB Developer

DB Server

## 5-Application

GUI -- Interface

Web Site -- Mobile -- Desktop

Application Programmer (DB User)

MVC C# java React Pho

Application Server

## 6-Client --> EndUser --> browser --> URL

## File Based System

### Delimited File

1,ahmed,22, <b>10</b> ok
2,ali,24, <b>70</b>
3,eman,25
3,eman,25
3,eman,25

Student.txt

- 1-Diff Search
- 2-low Performance
- 3-Separated Copies
- 4-No relationship
- 5-No DB Integrity
- 6-DB Duplication
- 7-long Development Time

### Fixed width File

10 java 20 SD
30 admin 40 HR
50 IS

Dept.txt

- 8-Security & Permissions
- 9-Constraints & Rules
- 10-No Data Quality
- 11-Manual Backup & Restore
- 12-No Standard
- 13-Diff Integration

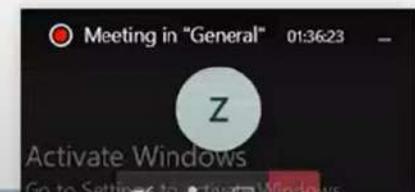
## DB System

### Tables & Relationship

Sid	Sname	
1	ahmed	10
2	ali	70 XXX
3	eman	
3	eman	XXXXX

did	dname
10	SD
20	IS
30	CS

- 1- One Standard
- 2-MetaData + Data
- 3-Column --> DataType
- 4-Primary key (Unique -- not NULL)
- 5-Centralized DB



# Database

Introduction To SQL  
Programming

# Day1

- DB Life cycle
- File Based System & its Disadvantages and Limitations
- DBMS Advantages & Disadvantages
- ERD Notations
- Entities & Attributes & relations
- Keys & Constraints
- Case Study

# File Based System

- Separation & Isolation Of data (each user has a copy) cause inconsistencies
- Incompatible File Formats
- Program-Data Dependence
  - All programs maintain metadata for each file they use
  - Each application program needs to include code for the metadata of each file
  - Non-standard file formats
- Lengthy Development Times
  - Programmers must design their own file formats (Metadata)
- Data Redundancy (Duplication of data)
  - Different systems/programs have separate copies of the same data
  - When data changes in one file, could cause inconsistencies
  - No Database integrity
- Limited Data Sharing
  - No centralized control of data

# Basic Definitions

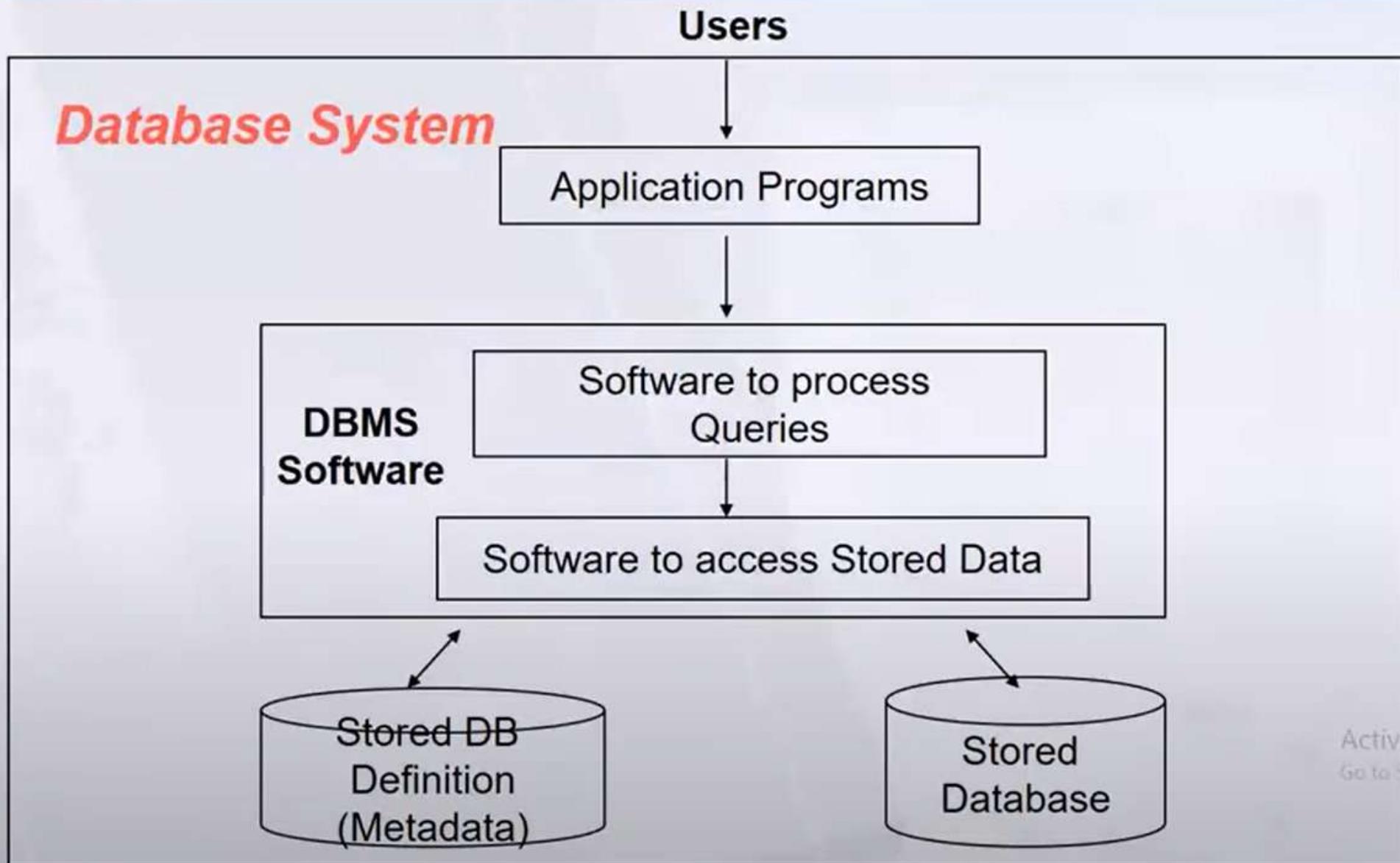
- **Database:** A collection of related data.
- **Database Management System (DBMS):** A software package/ system to facilitate the creation and maintenance of a computerized database.(model introduced in 1970 IBM but RDBMS appears in 1980)
- **Database System:** The DBMS software together with the data itself. Sometimes, the applications are also included. ( **Software + Database** )

 Company_SD.mdf	20/05/2014 10:35	SQL Server Database	7,168 KB
 Company_SD_log.ldf	20/05/2014 10:35	SQL Server Database	3,456 KB

Services (Local)					
SQL Server (MSSQLSERVER)	Name	Description	Status	Startup Type	Log On As
<a href="#">Stop the service</a>	Secondary Log-on	Enables start...	Running	Manual	Local System
<a href="#">Pause the service</a>	Secure Socket Tunneling Protocol Service	Provides sup...	Running	Manual	Local Service
<a href="#">Restart the service</a>	Security Accounts Manager	The startup ...	Running	Automatic	Local System
	Security Center	The WSCSVC...	Running	Automatic (De...	Local Service
	Sensor Data Service	Delivers dat...		Manual (Trigg...	Local System
	Sensor Monitoring Service	Monitors va...		Manual (Trigg...	Local Service
	Sensor Service	A service for ...		Manual (Trigg...	Local System
	Server	Supports file...	Running	Automatic (Tr...	Local System
	Shared PC Account Manager	Manages pr...		Disabled	Local System
	Shell Hardware Detection	Provides not...	Running	Automatic	Local System
	Smart Card	Manages ac...	Running	Manual (Trigg...	Local Service
	Smart Card Device Enumeration Service	Creates soft...		Manual (Trigg...	Local System
	Smart Card Removal Policy	Allows the s...		Manual	Local System
	SNMP Service	Enables Sim...	Running	Automatic	Local System
	SNMP Trap	Receives tra...		Manual	Local Service
	Software Protection	Enables the ...		Automatic (De...	Network Se...
	Spatial Data Service	This service i...		Manual	Local Service
	Spot Verifier	Verifies pote...		Manual (Trigg...	Local System
	SQL Server (ALEX)	Provides sto...	Running	Automatic	\Rami
	SQL Server (CAIRO)	Provides sto...	Running	Automatic	\Rami
	SQL Server (ISMAILIA)	Provides sto...	Running	Automatic	\Rami
	SQL Server (MANSOURA)	Provides sto...	Running	Automatic	\Rami
	SQL Server (MSSQLSERVER)	Provides sto...	Running	Automatic	\Rami
	SQL Server Agent (ALEX)	Executes job...	Running	Automatic	\Rami
	SQL Server Agent (CAIRO)	Executes job...		Automatic	\Rami
	SQL Server Agent (ISMAILIA)	Executes job...		Automatic	\Rami
	SQL Server Agent (MANSOURA)	Executes job...		Automatic	\Rami
	SQL Server Agent (MSSQLSERVER)	Executes job...	Running	Automatic	\Rami
	SQL Server Analysis Services (MSSQLSERVER)	Supplies onl...	Running	Automatic	\Rami
	SQL Server Analysis Services CEIP (MSSQLSERVER)	CEIP service ...	Running	Automatic	NT Service\...
	SQL Server Browser	Provides SQ...	Running	Automatic	Local Service
	SQL Server CEIP service (ALEX)	CEIP service ...	Running	Automatic	NT Service\...
	SQL Server CEIP service (CAIRO)	CEIP service ...	Running	Automatic	NT Service\...
	SQL Server CEIP service (ISMAILIA)	CEIP service ...	Running	Automatic	NT Service\...

Activate Windows  
Go to Settings to activate Windows.

# Database System



# DBMS Advantages

- **Standardization** and better Data accessibility and response (SQL)
- **Sharing data.**
  - Different users get different views of the data
- **Enforcing Integrity Constraints**
- **Improved Data Quality**
  - Constraints, data validation rules
- **Inconsistency can be avoided because of data sharing.**
- **Restricting Unauthorized Access.**
- **Providing Backup and Recovery.**
  - Disaster recovery is easier
- **Minimal Data Redundancy**
  - Leads to increased data integrity/consistency
- **Program-Data Independence**
  - Metadata stored in DBMS, so applications don't worry about data formats
  - Data queries/updates managed by DBMS

## DBMS Disadvantages

- It needs **expertise** to use
- DBMS itself is **expensive**
- The DBMS may be **incompatible** with any other available **DBMS**

# Database Users

- Database Administrator (DBA)
- System Analysts
- Database Designer
- Database Developer
- Application programmers
- BI & BigData Specialist (Data Scientist)
- End users

# Entity Relationship Modeling

## **Entity-Relationship Diagram (ERD)**

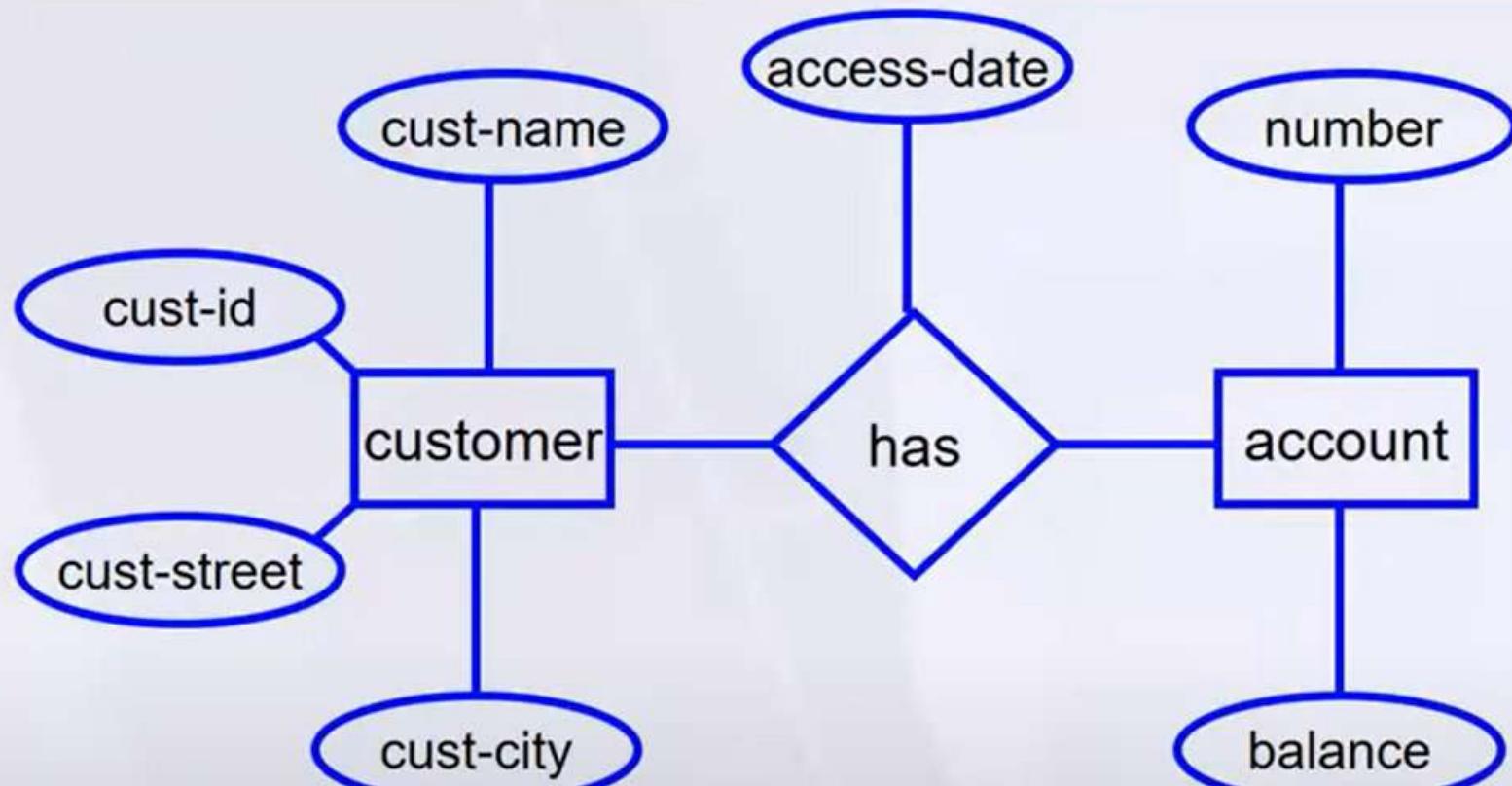
identifies information required by the business by displaying the relevant entities and the relationships between them.

# The ER Model

## Basic constructs of the E-R model:

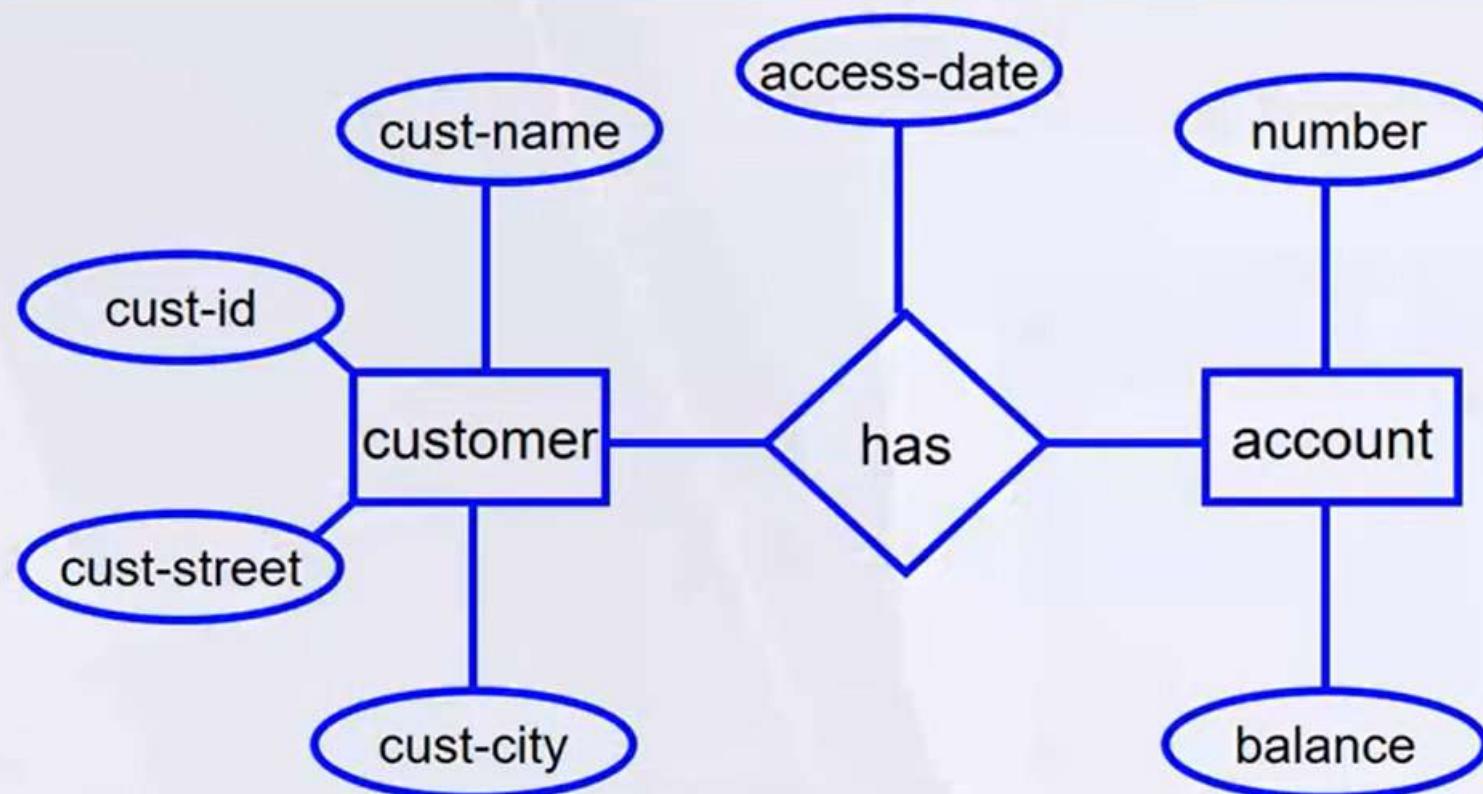
1. **Entities** - person, place, object, event, concept (often corresponds to a real time object that is **distinguishable** from any other object)
2. **Attributes** - property or **characteristic** of an entity type (often corresponds to a field in a table)
3. **Relationships** - **link** between entities (corresponds to primary **key-foreign key** equivalencies in related tables)

# ER Diagram: Starting Example



- Rectangles: entity sets
- Diamonds: relationship sets
- Ellipses: attributes

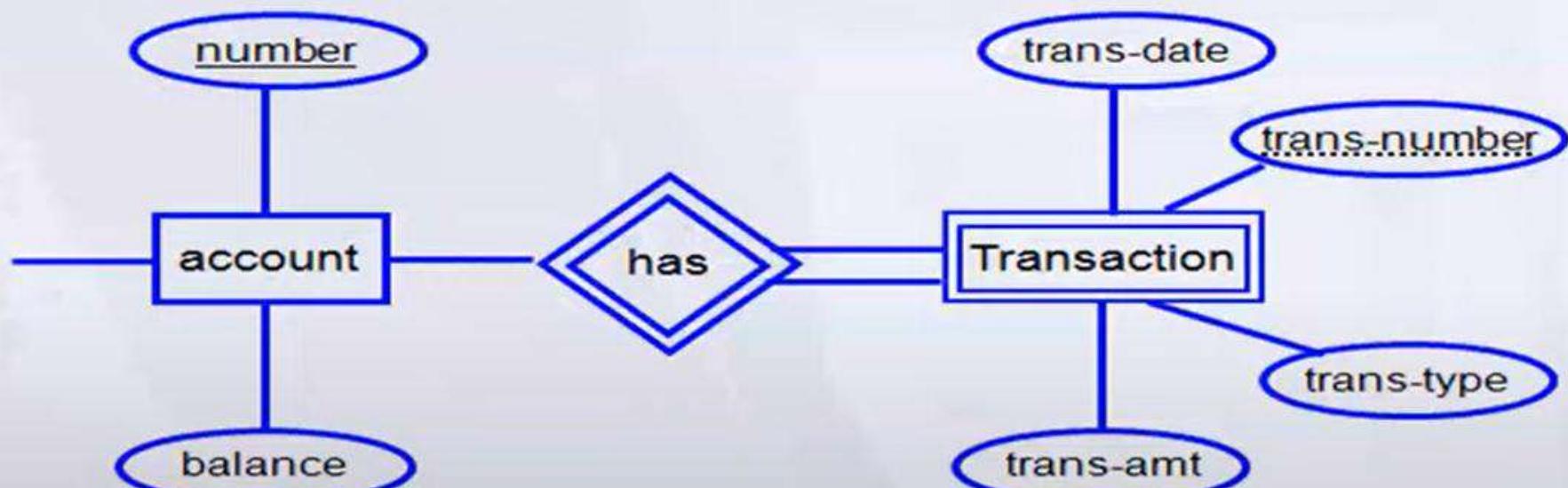
# ER Diagram: Starting Example



- ▶ Rectangles: entity sets
- ▶ Diamonds: relationship sets
- ▶ Ellipses: attributes

# Strong Entity Vs Weak Entity

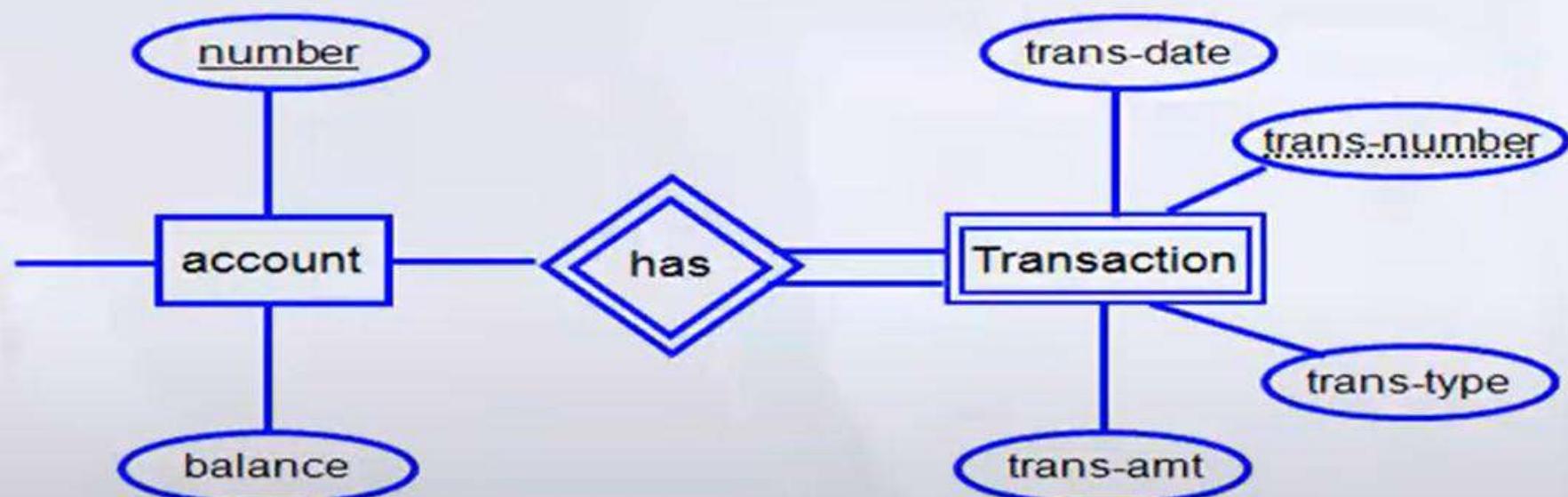
- › A **Strong Entity**- An Entity set that has a primary key.
- › A **Weak Entity**- An entity set that do not have sufficient attributes to form a primary key.



**Partial key:** A set of attributes that can be associated with P.K of an owner entity set to distinguish a weak entity.

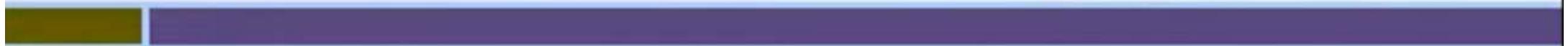
# Strong Entity Vs Weak Entity

- › A **Strong Entity**- An Entity set that has a primary key.
- › A **Weak Entity**- An entity set that do not have sufficient attributes to form a primary key.



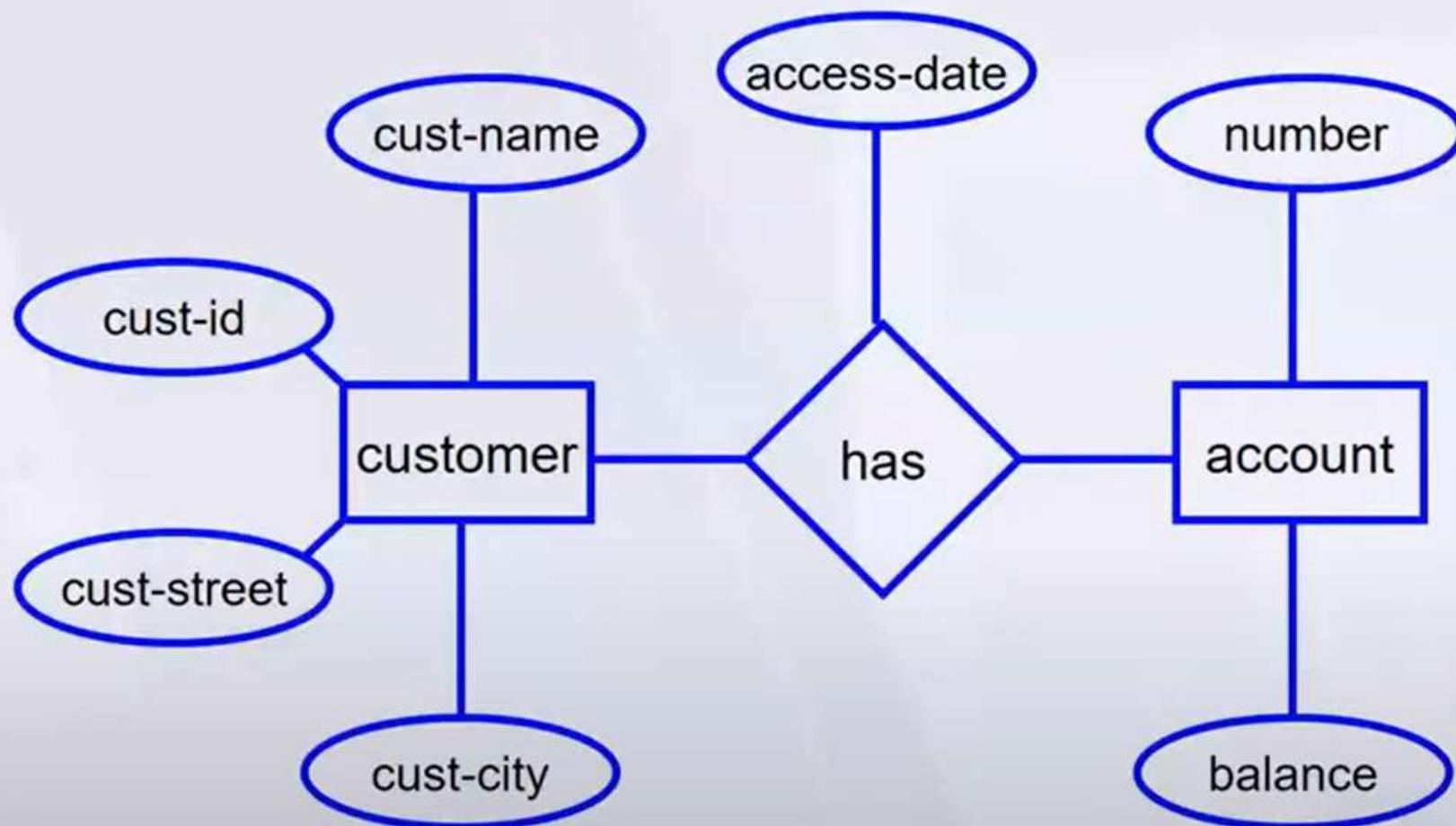
**Partial key:** A set of attributes that can be associated with P.K of an owner entity set to distinguish a weak entity.

## Next: Types of Attributes

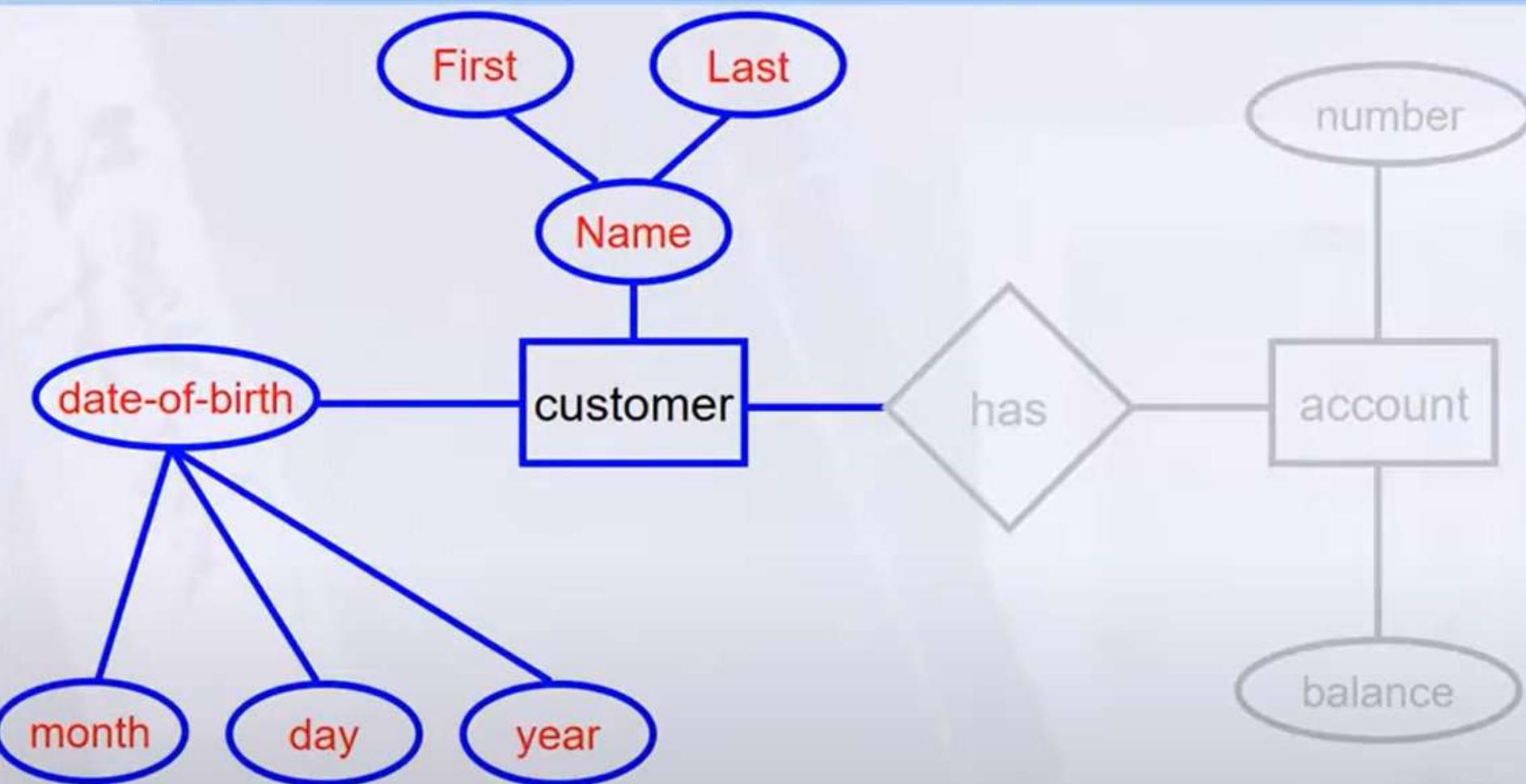


1. **Composite Attribute**
2. **Multi-valued Attribute**
3. **Derived Attribute**
4. **Complex Attribute**
5. **Simple Attribute**

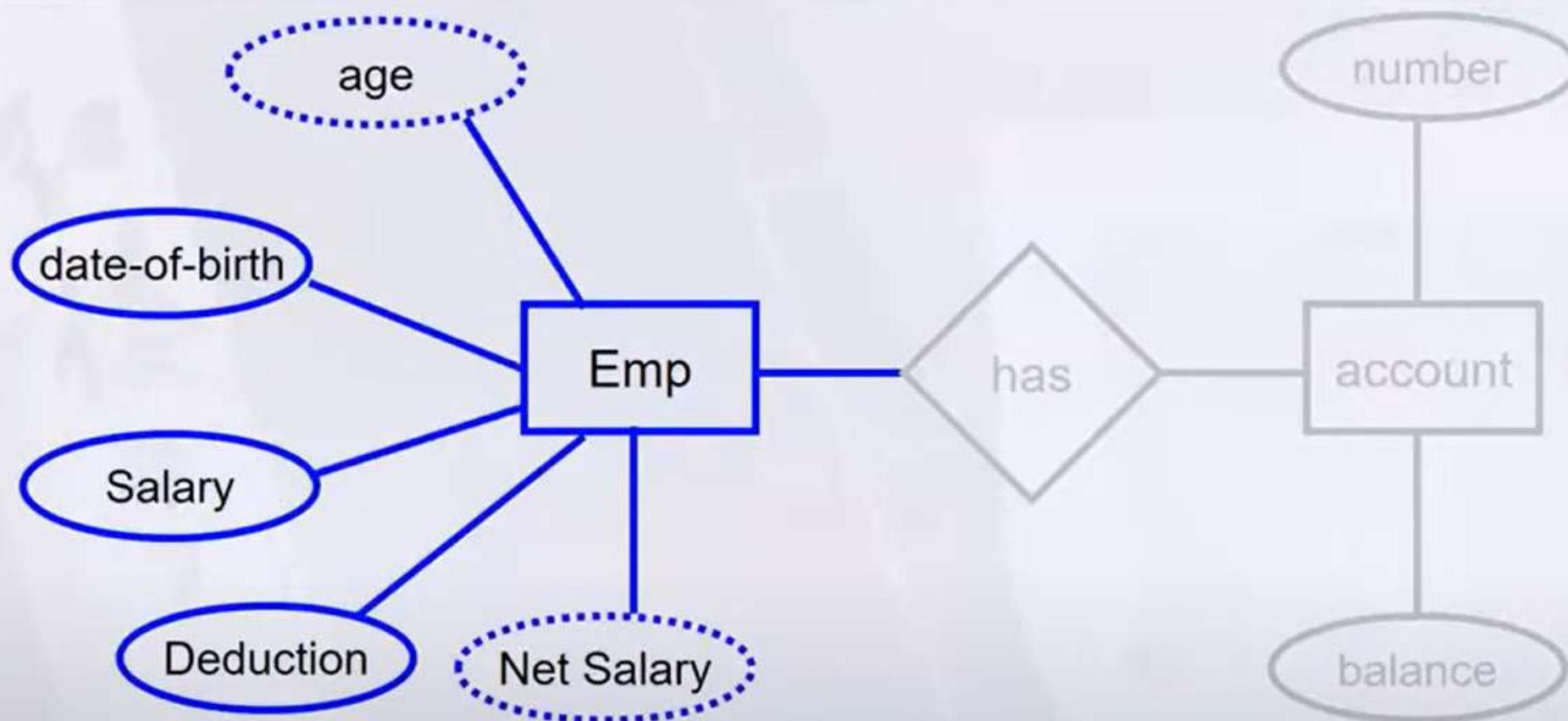
# Simple Attribute



# Composite Attribute

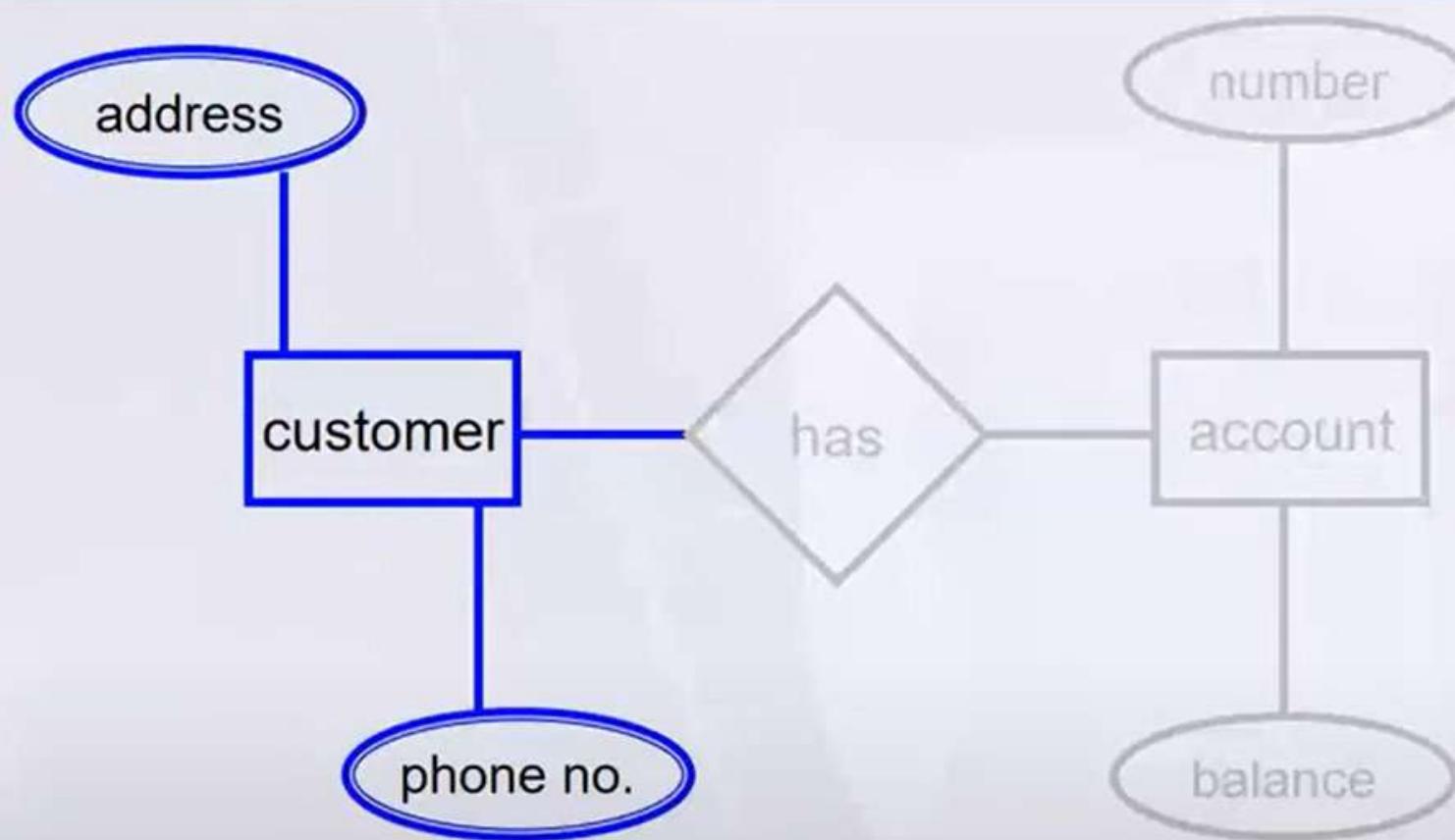


# Derived Attribute



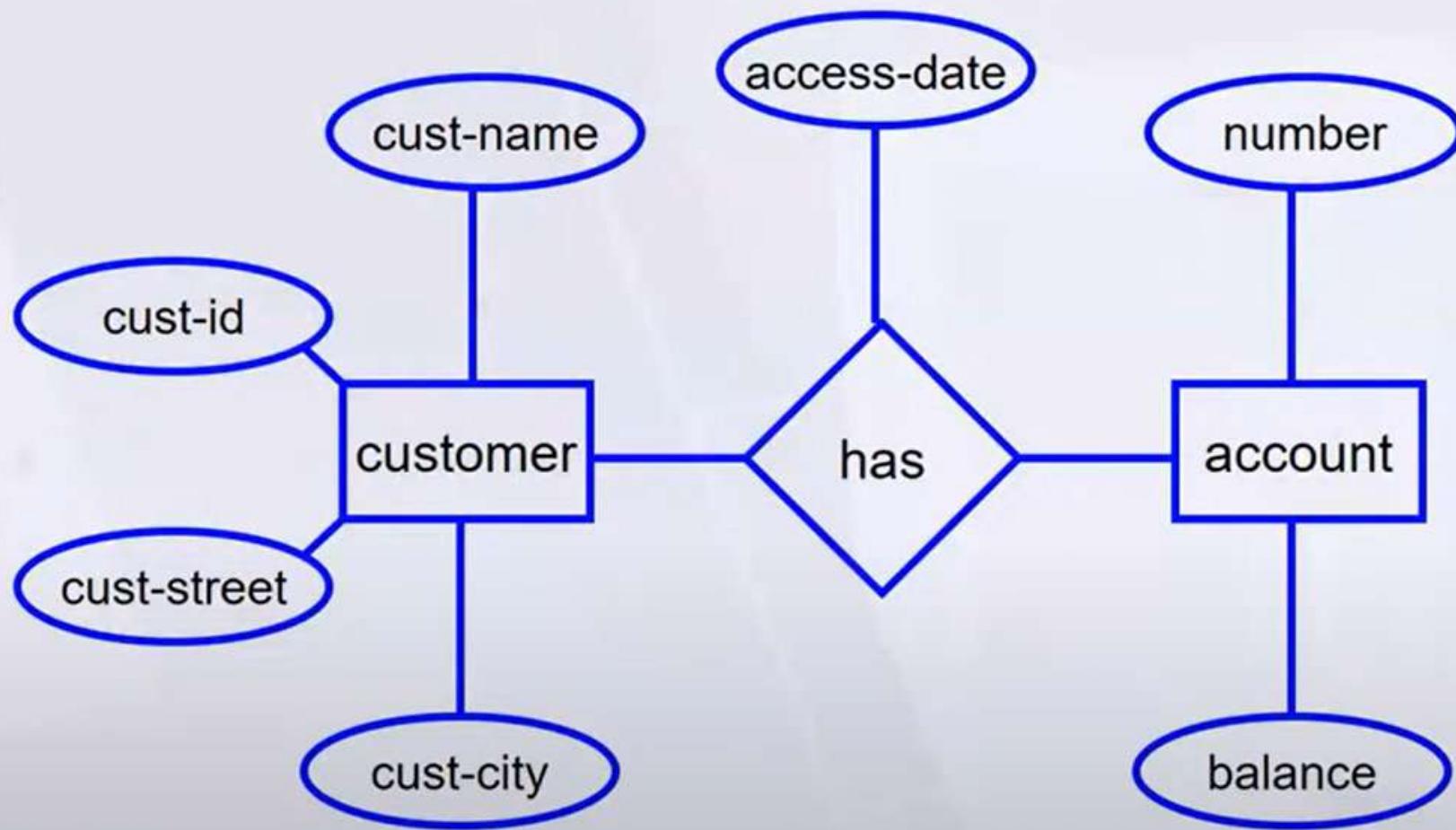
➤ derived (dashed ellipse)

# Multi-valued

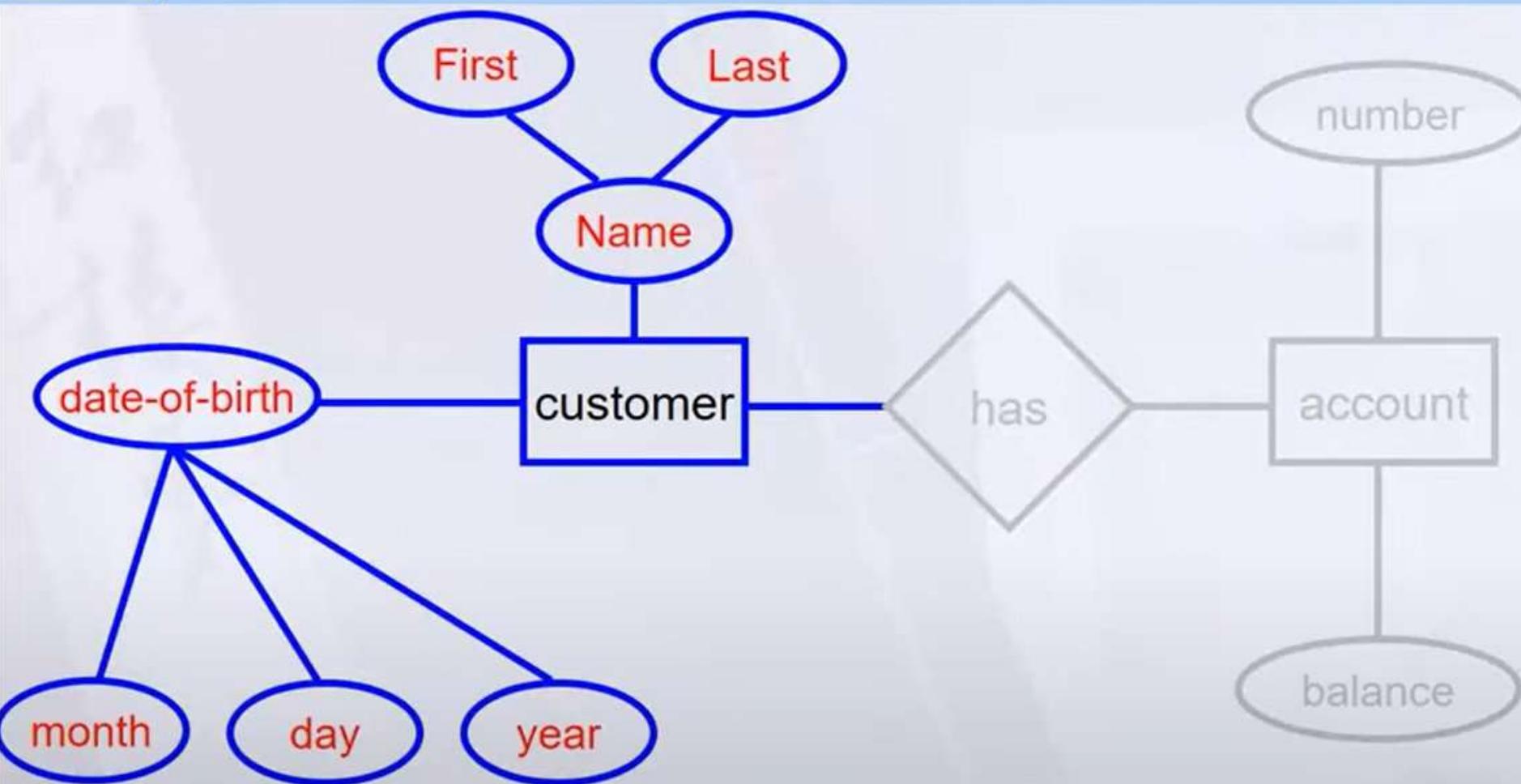


▶ multi-valued (double ellipse)

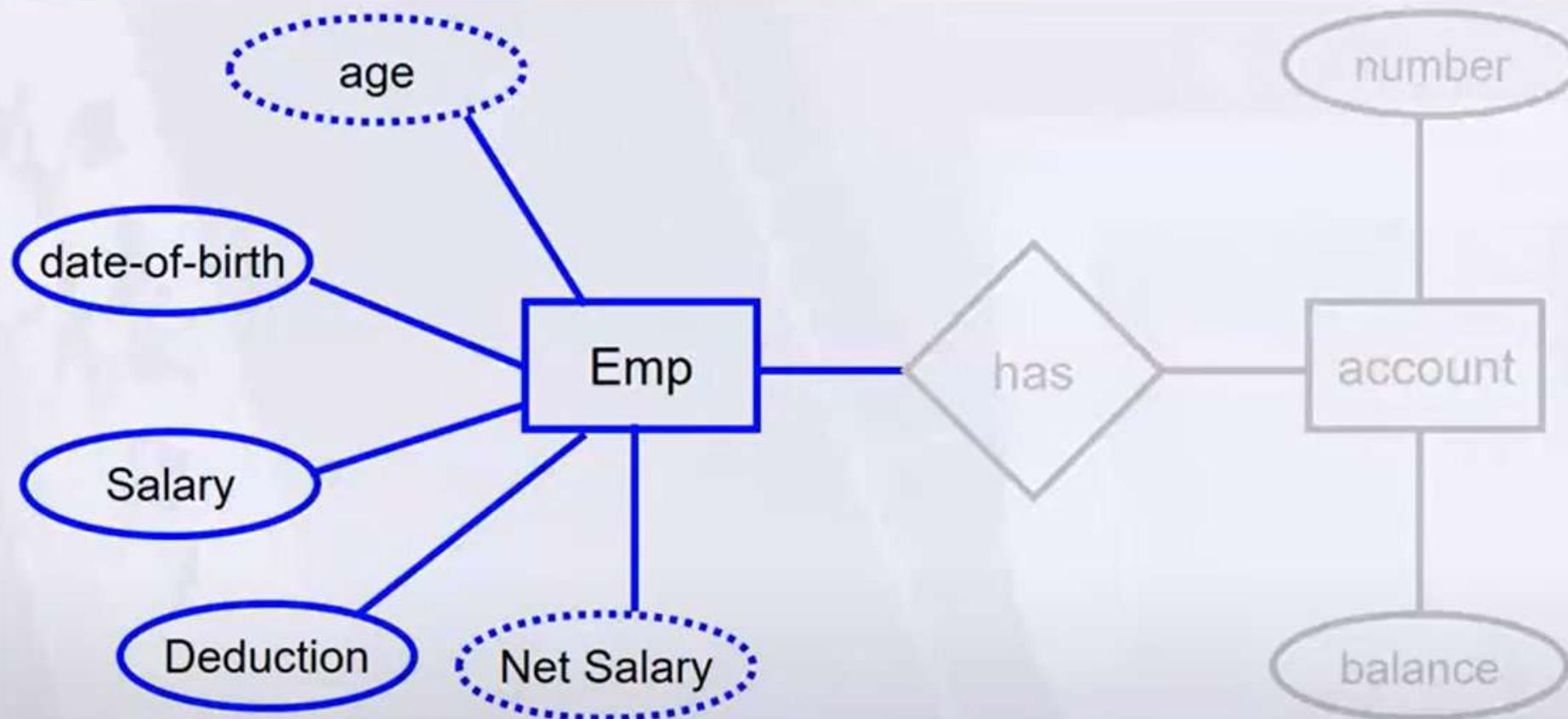
# Simple Attribute



# Composite Attribute

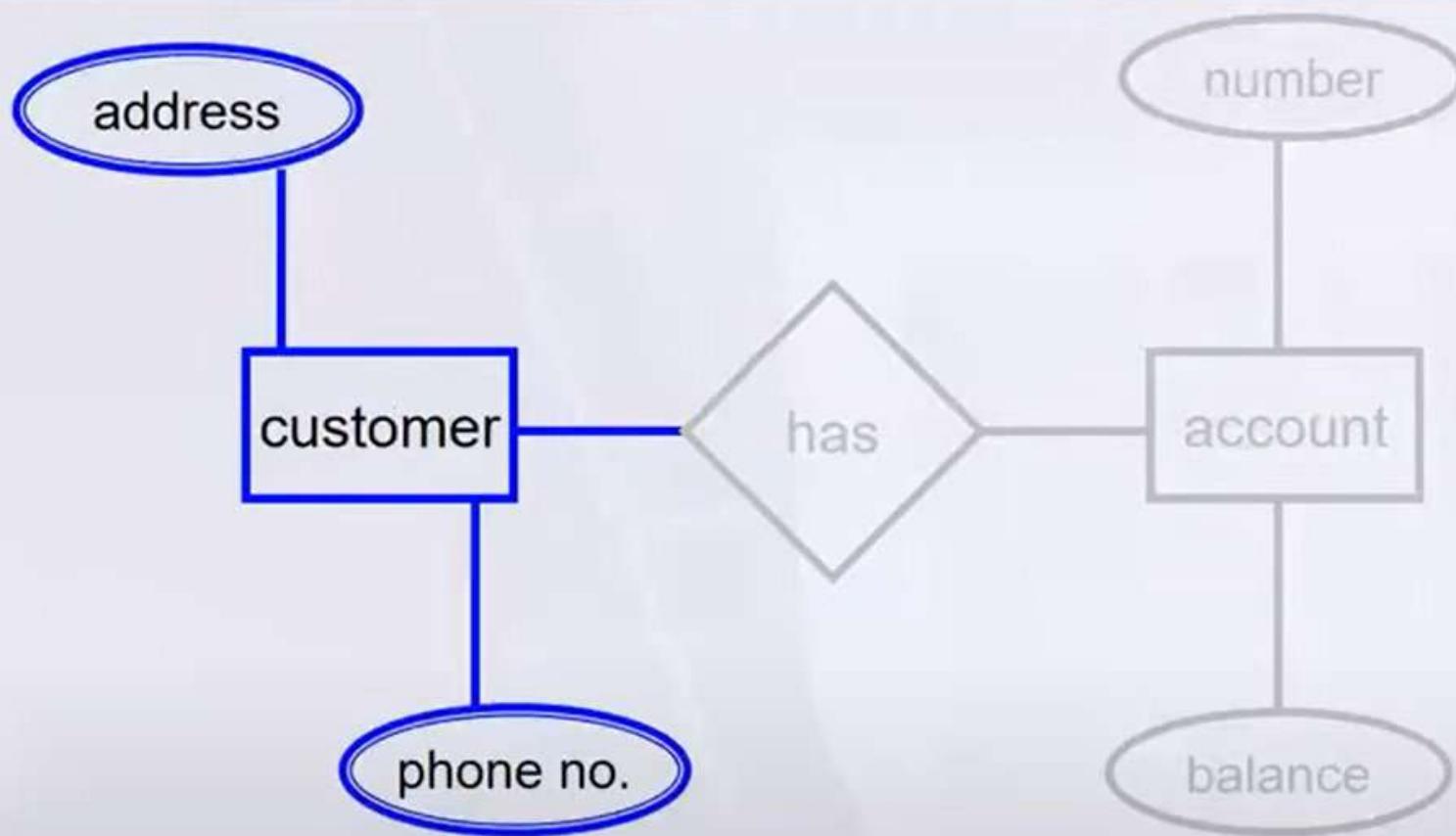


# Derived Attribute



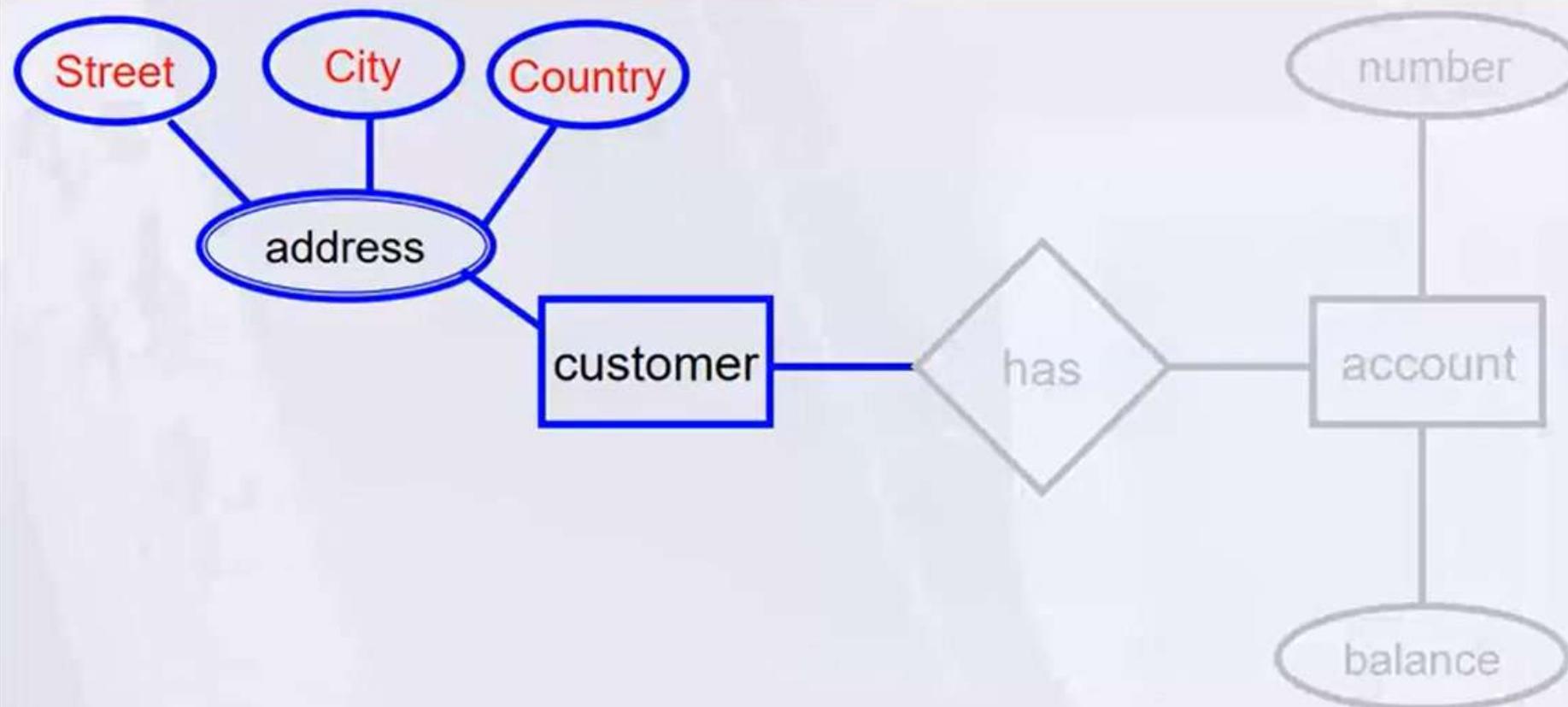
➤ derived (dashed ellipse)

# Multi-valued



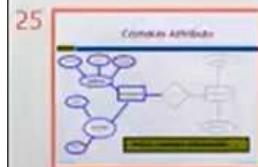
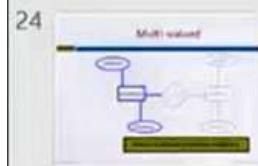
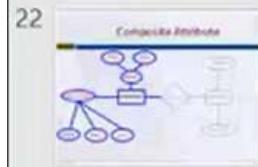
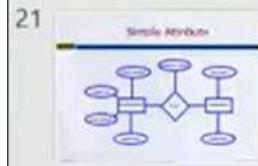
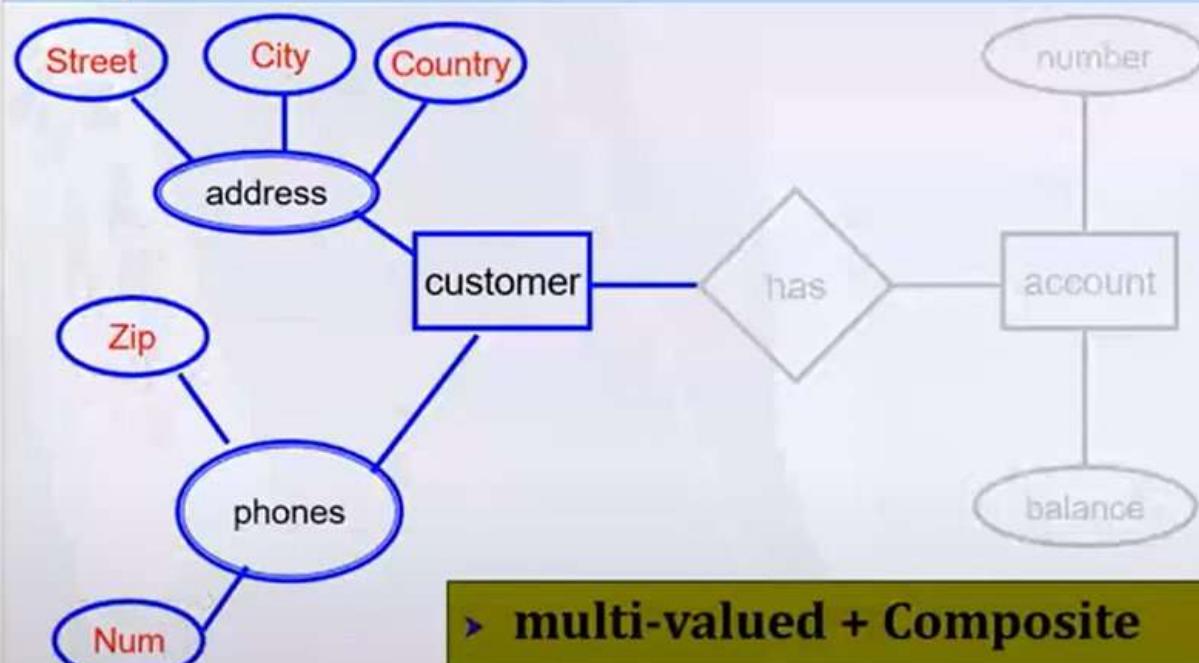
▶ multi-valued (double ellipse)

# Complex Attribute



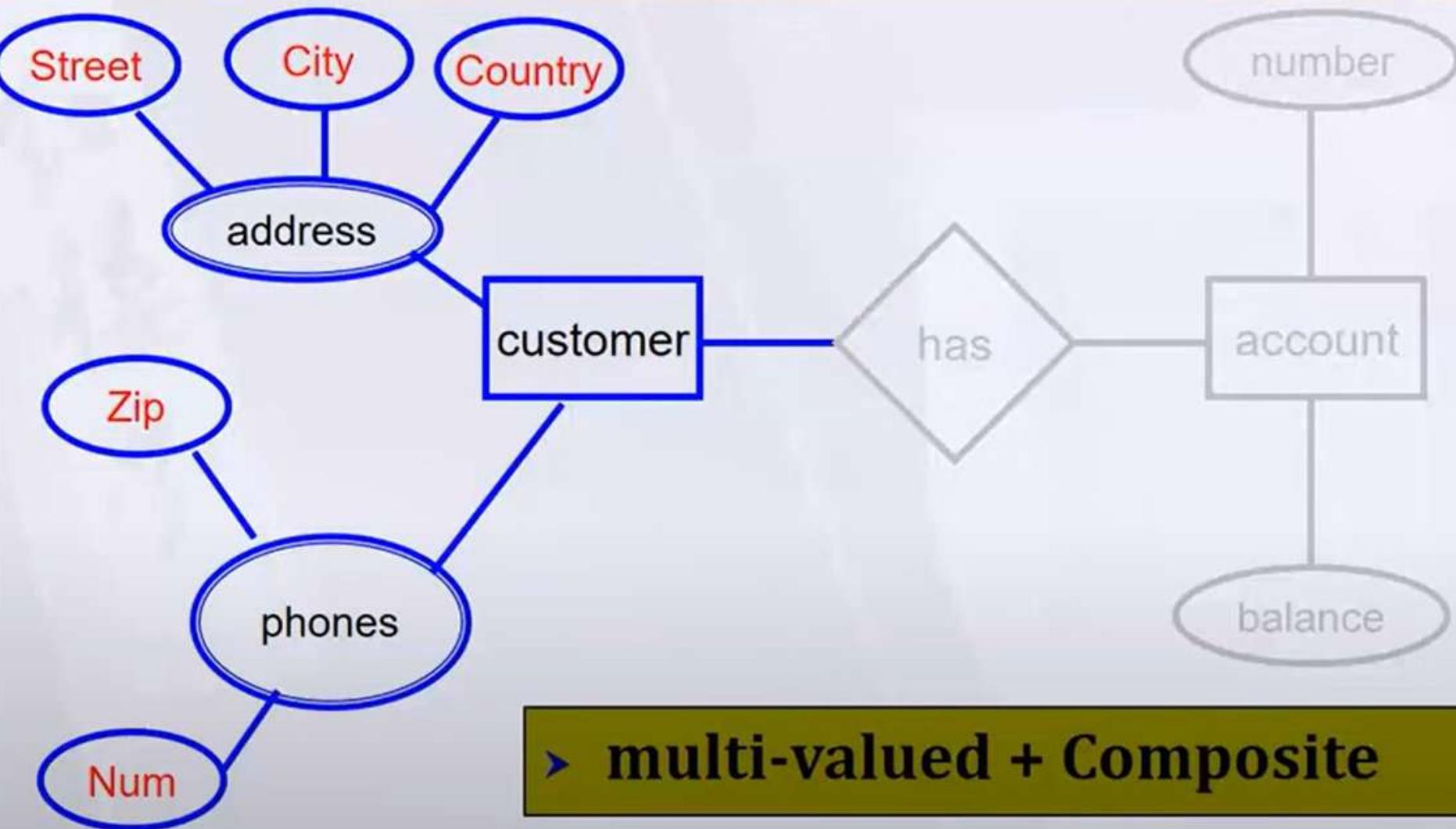
➤ multi-valued + Composite

## Complex Attribute



26 Aggregation

# Complex Attribute

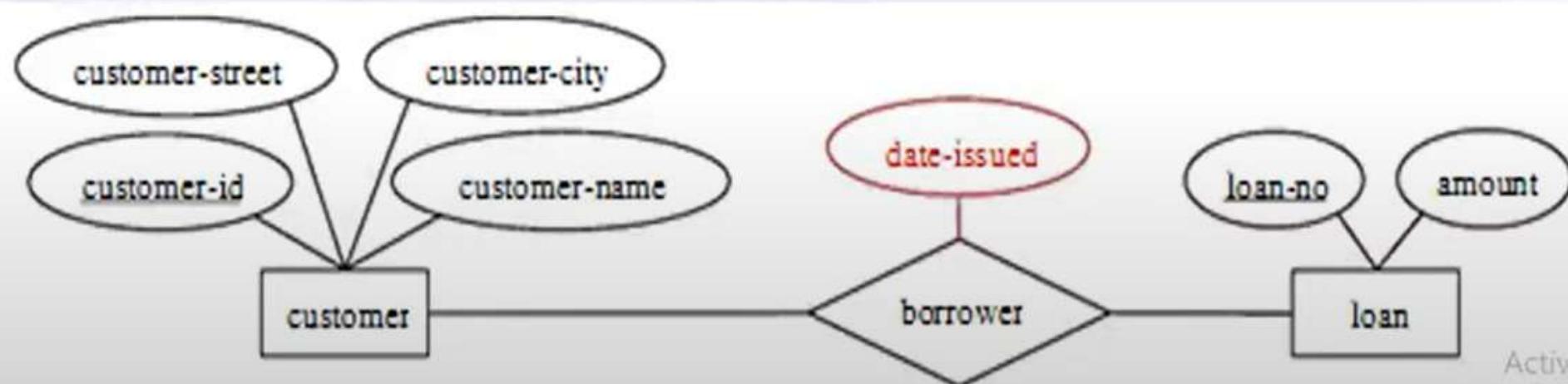


➤ multi-valued + Composite

# 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.



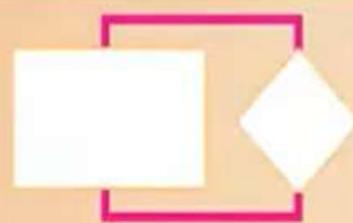
# **Relation**

**Relation has three Properties:**

- Degree of Relationships
- Cardinality Constraint
- Participation Constraint

# Degree of Relationships

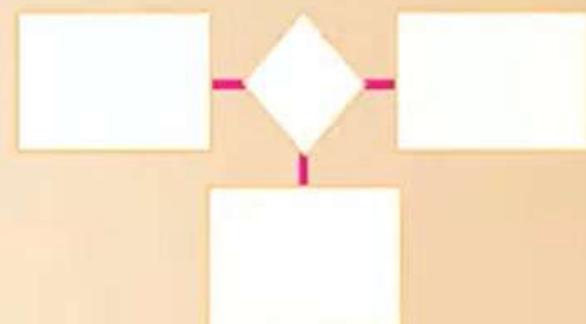
- Degree: number of entity types that participate in a relationship
- Three cases
  - **Unary:** between two instances of one entity type
  - **Binary:** between the instances of two entity types
  - **Ternary:** among the instances of three entity types



Unary



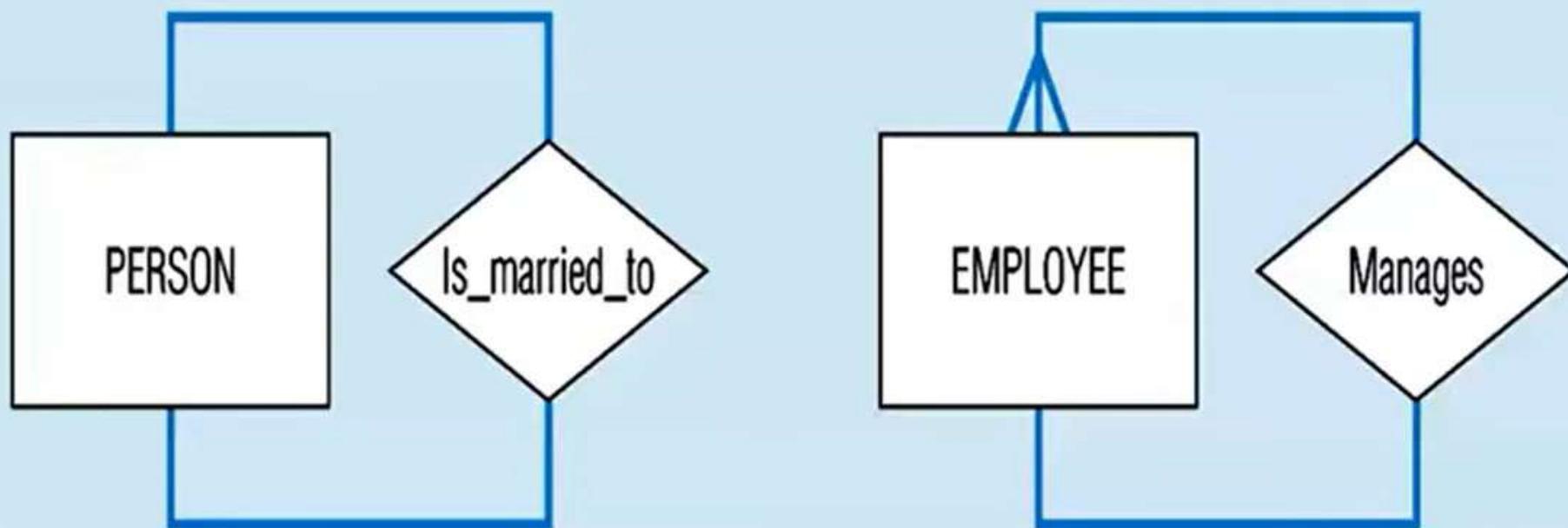
Binary

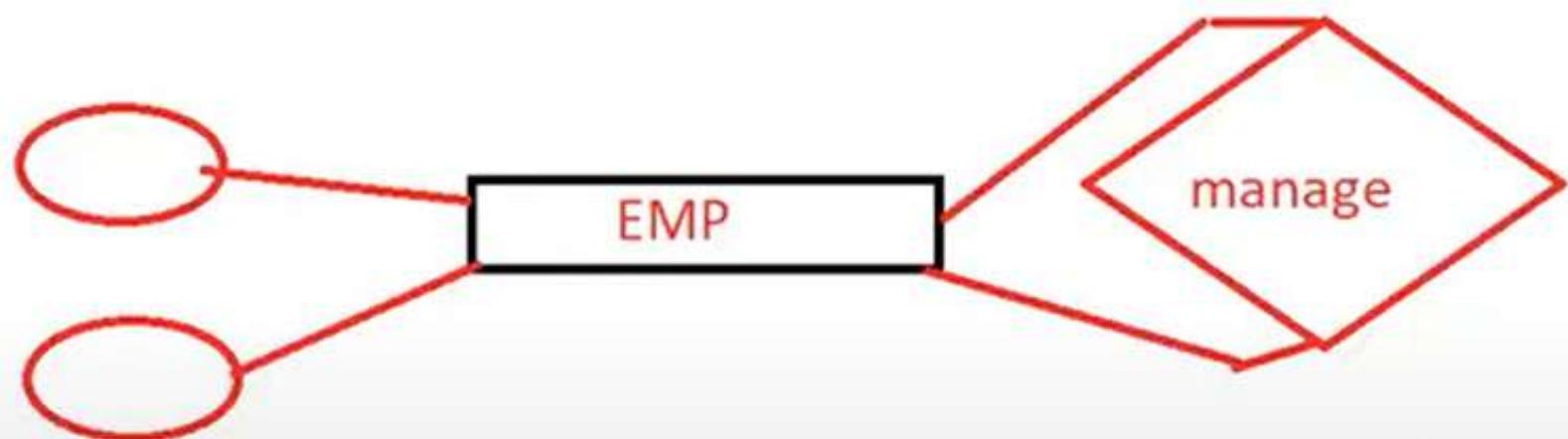
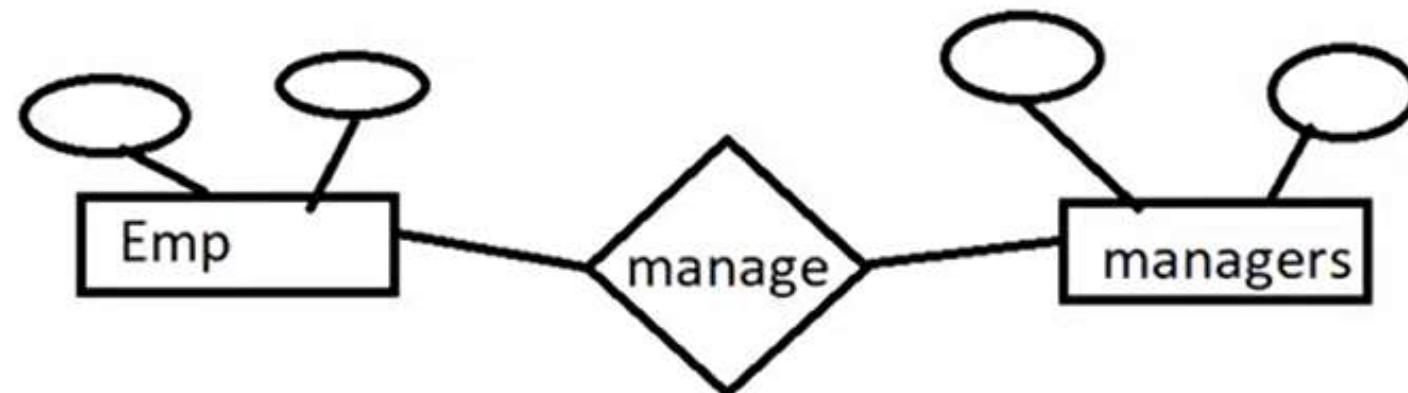


Ternary

# Recursive Relationship (Unary)

- **Recursive Relationships** - A relationship in which the same entity participates more than once.

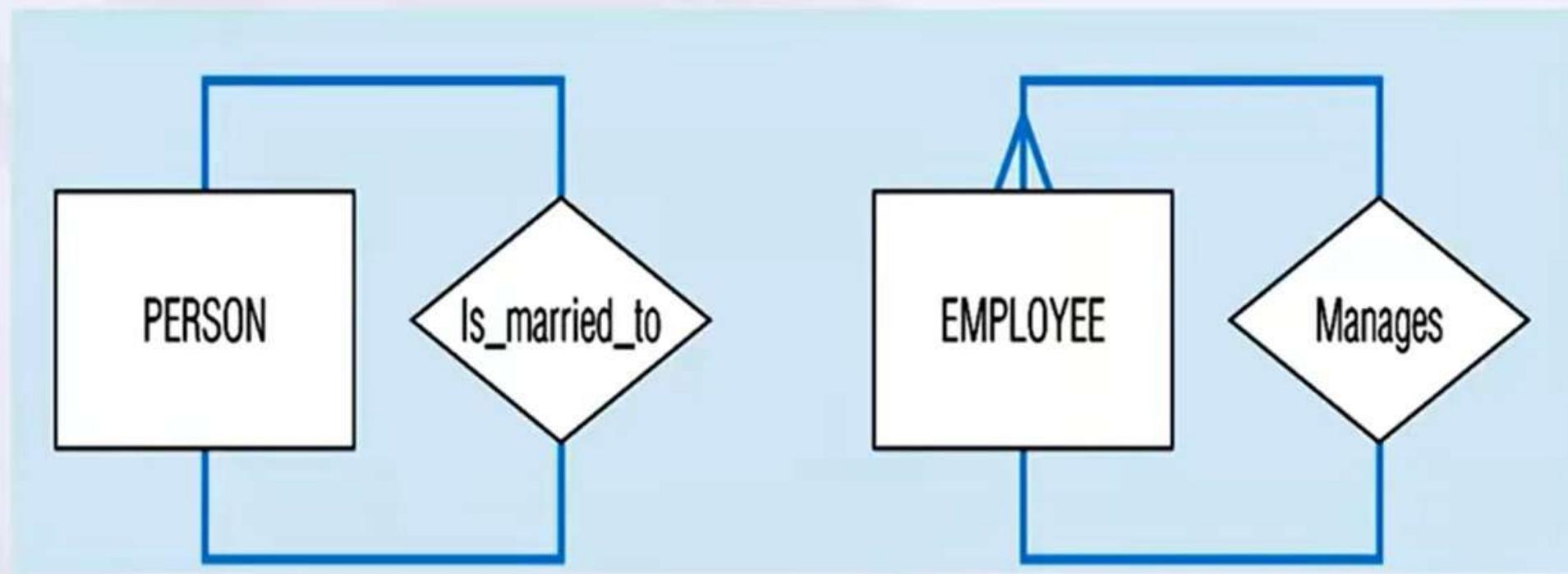




I

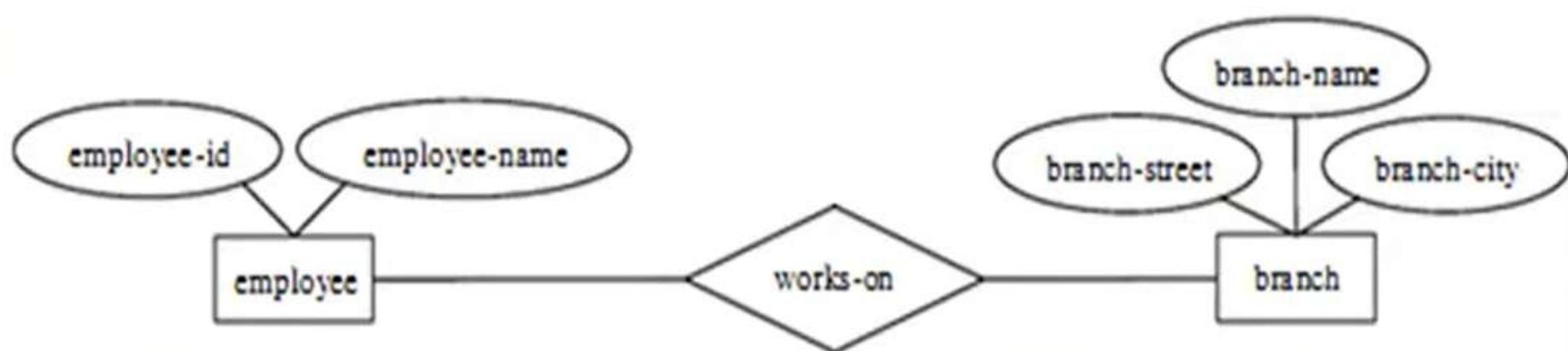
# Recursive Relationship (Unary)

- **Recursive Relationships** - A relationship in which the same entity participates more than once.



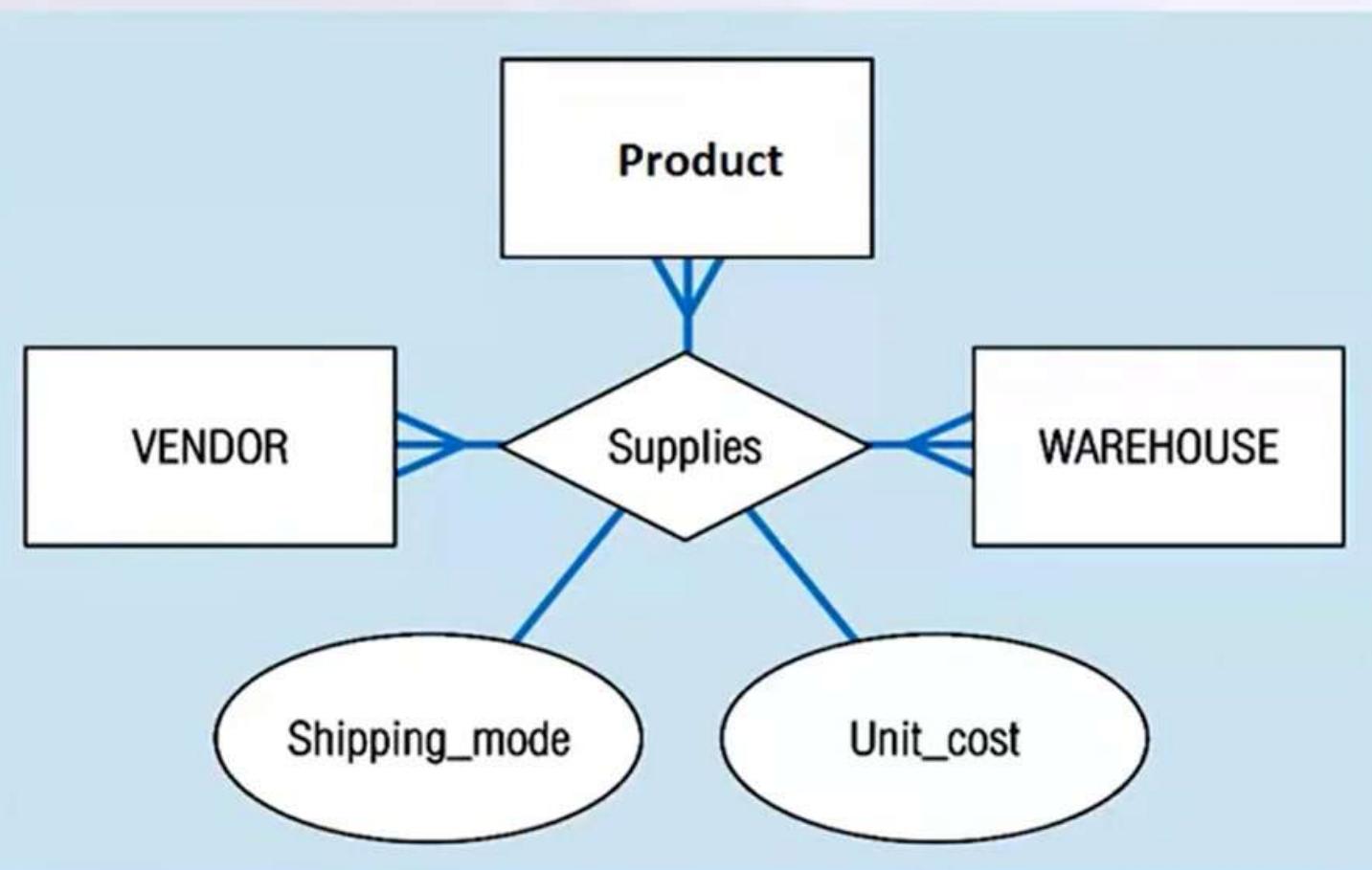
# *Binary Relationship*

- A binary relationship set is of degree 2.



# Ternary Relationship

- ▶ ternary relationship set is of degree 3.

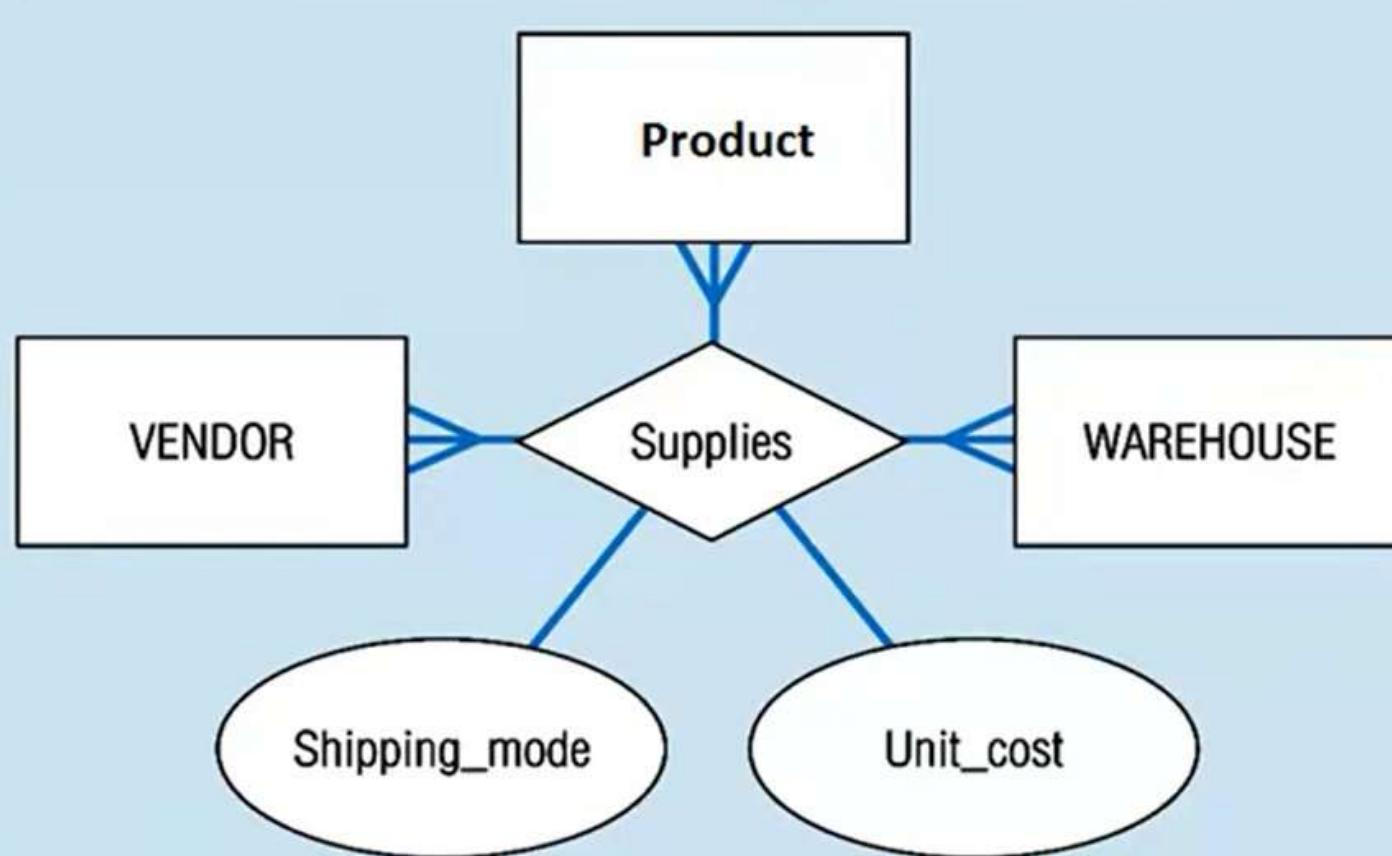


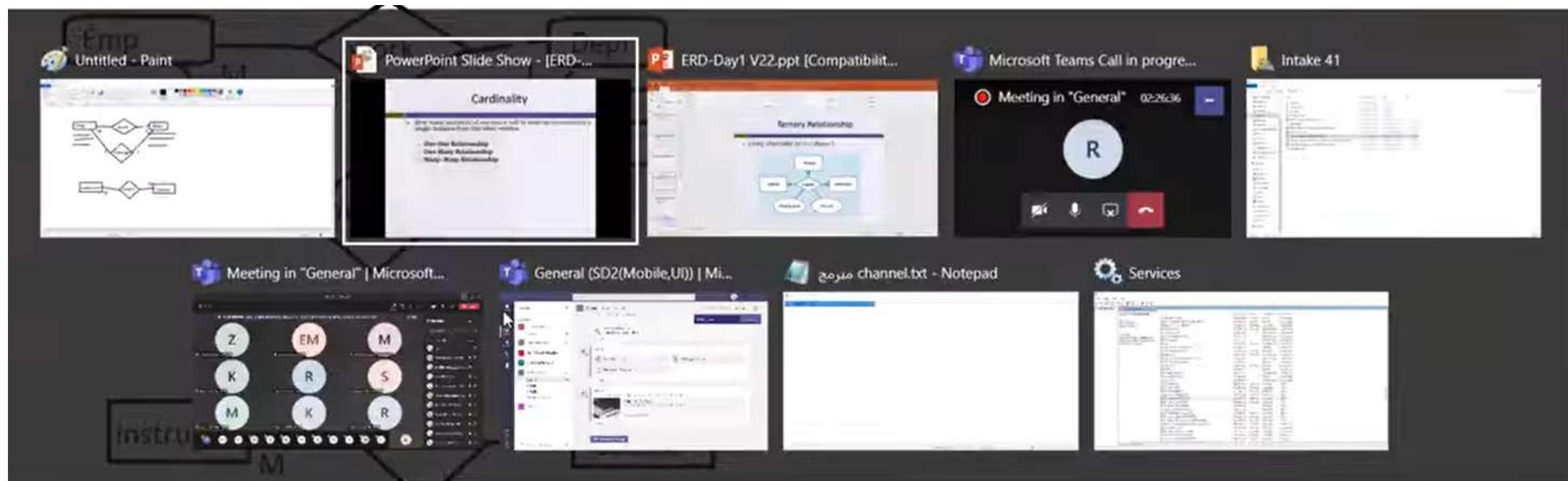
# Cardinality

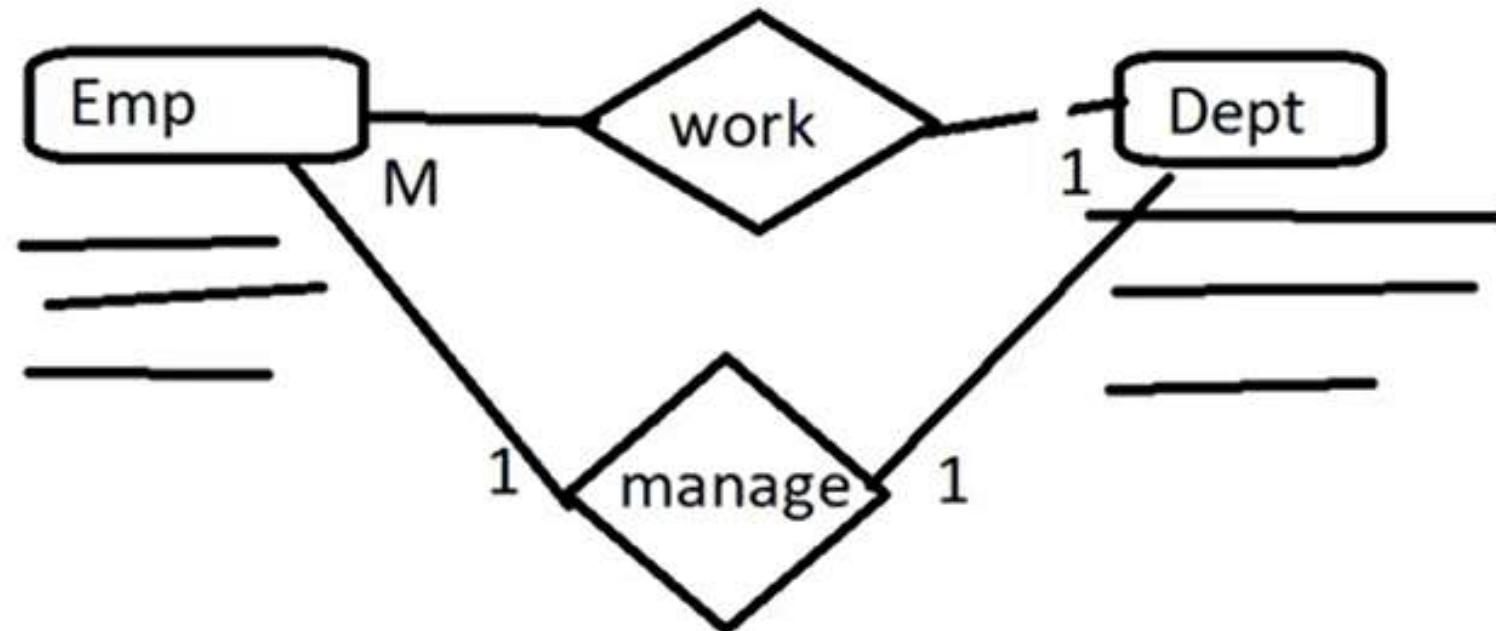
- How many instances of one entity will or must be connected to a single instance from the other entities.
  - **One-One Relationship**
  - **One-Many Relationship**
  - **Many- Many Relationship**

# Ternary Relationship

- ▶ ternary relationship set is of degree 3.





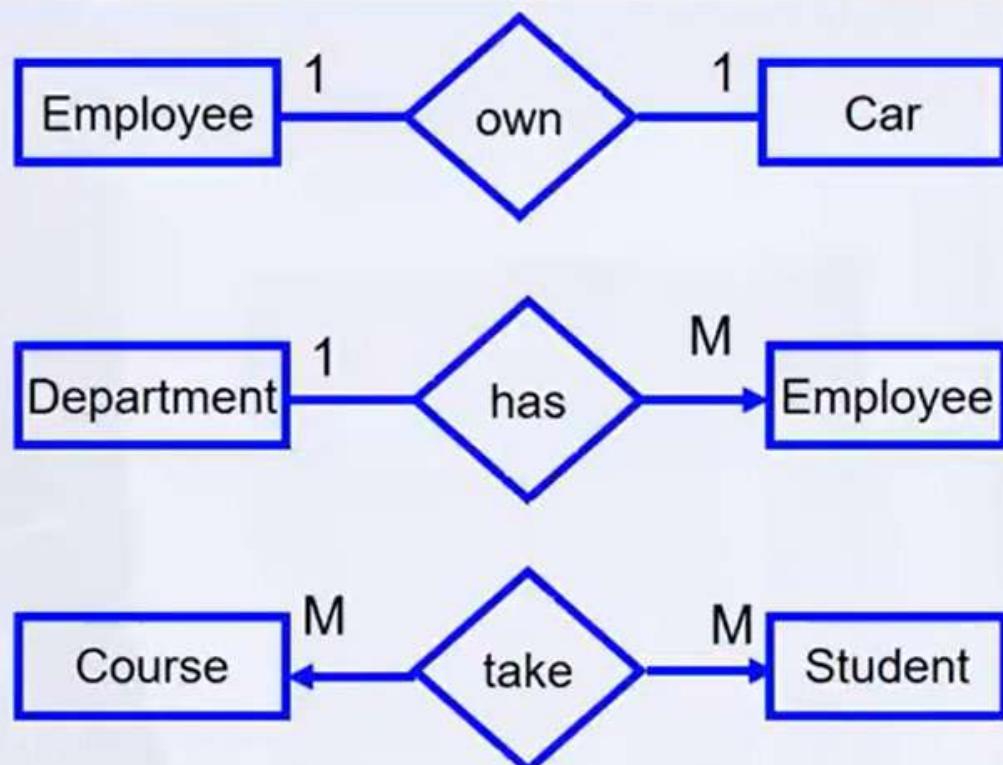


I



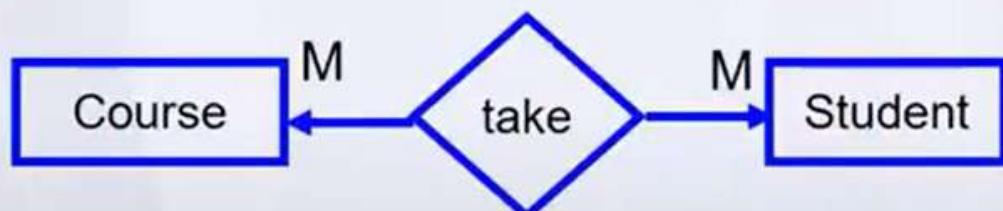
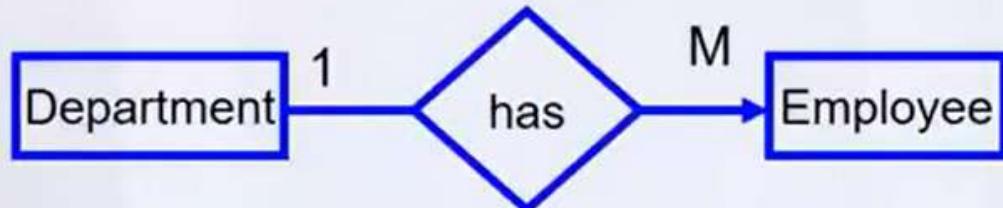
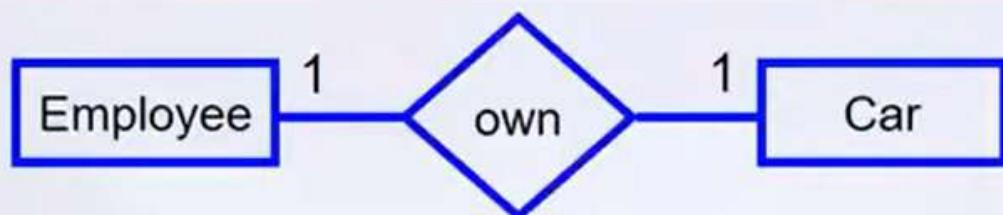
# Mapping Cardinalities

- ▶ One-to-One
- ▶ One-to-Many
- ▶ Many-to-Many



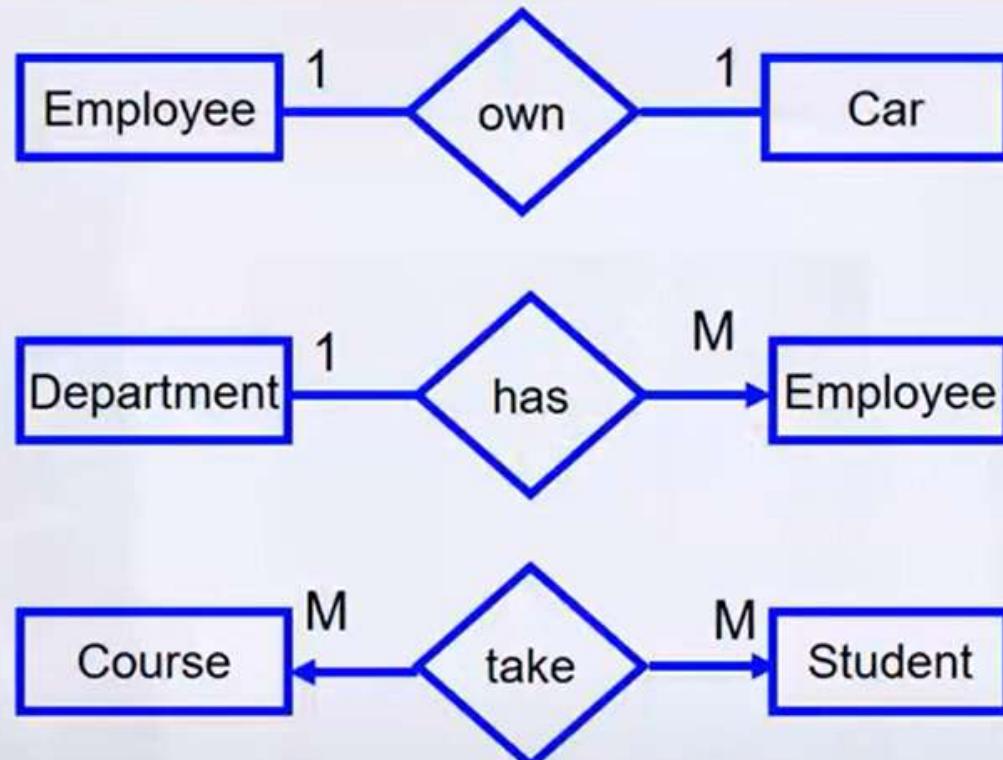
# Mapping Cardinalities

- ▶ One-to-One
- ▶ One-to-Many
- ▶ Many-to-Many



# Mapping Cardinalities

- ▶ One-to-One
- ▶ One-to-Many
- ▶ Many-to-Many

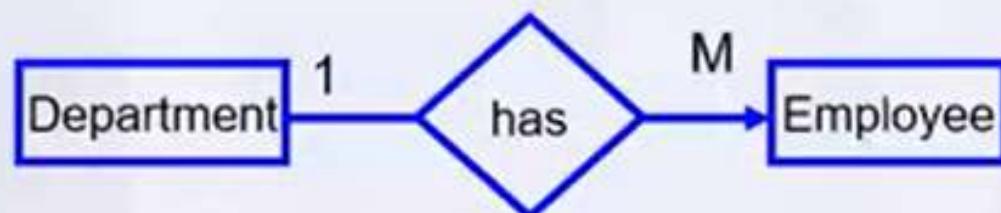


# Mapping Cardinalities

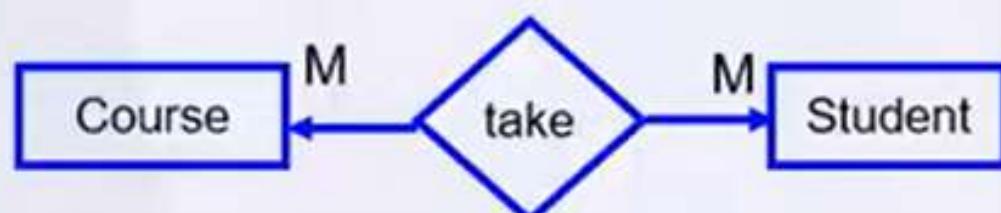
- One-to-One



- One-to-Many

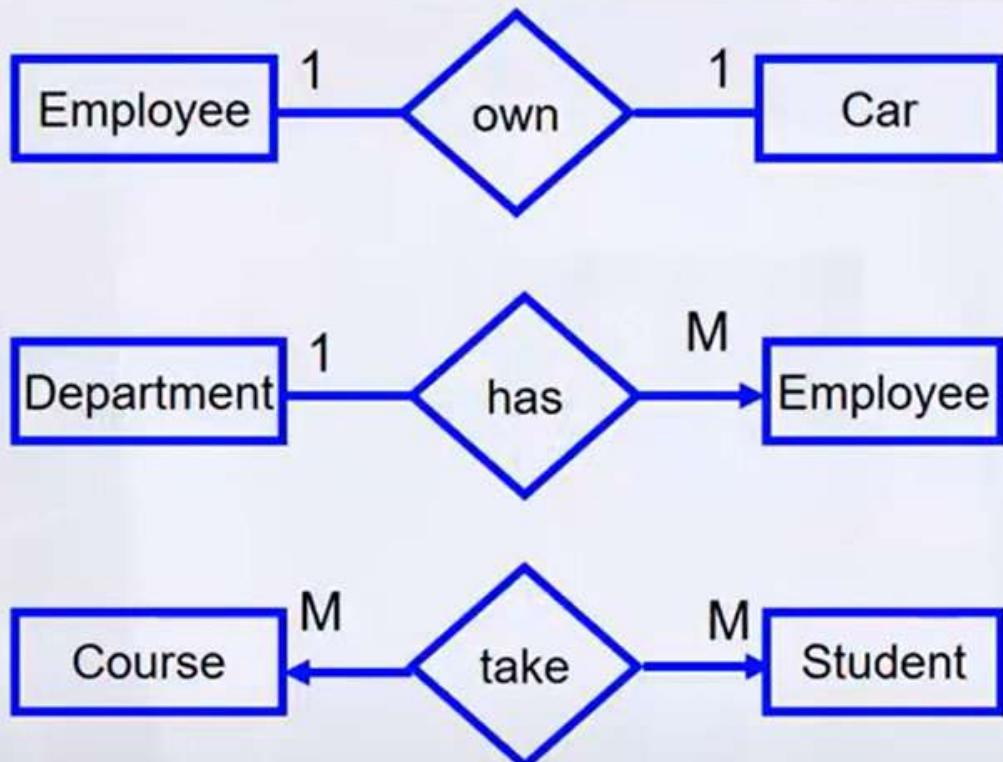


- Many-to-Many



# Mapping Cardinalities

- ▶ One-to-One
- ▶ One-to-Many
- ▶ Many-to-Many

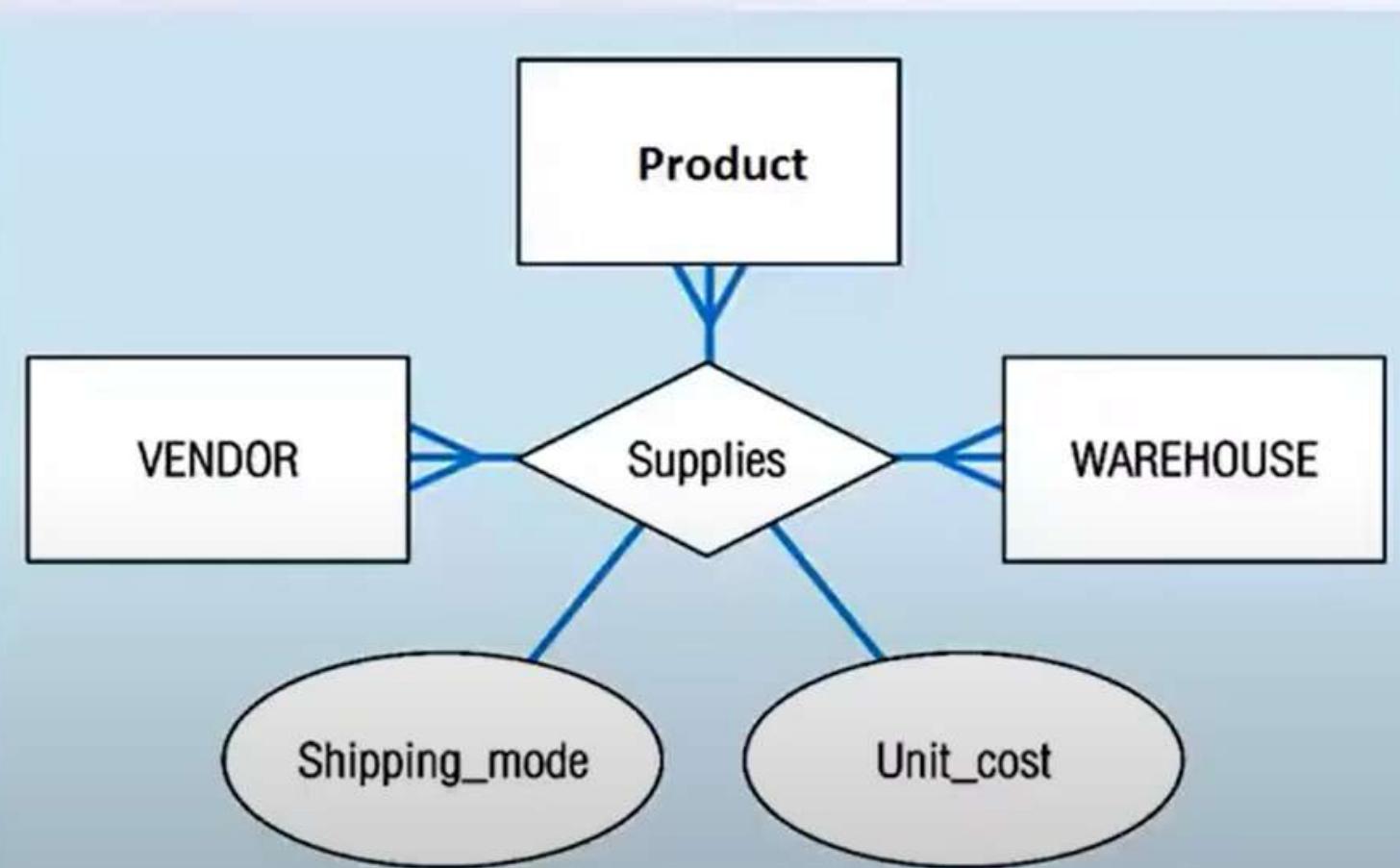


# Cardinality

- How many instances of one entity will or must be connected to a single instance from the other entities.
  - **One-One Relationship**
  - **One-Many Relationship**
  - **Many- Many Relationship**

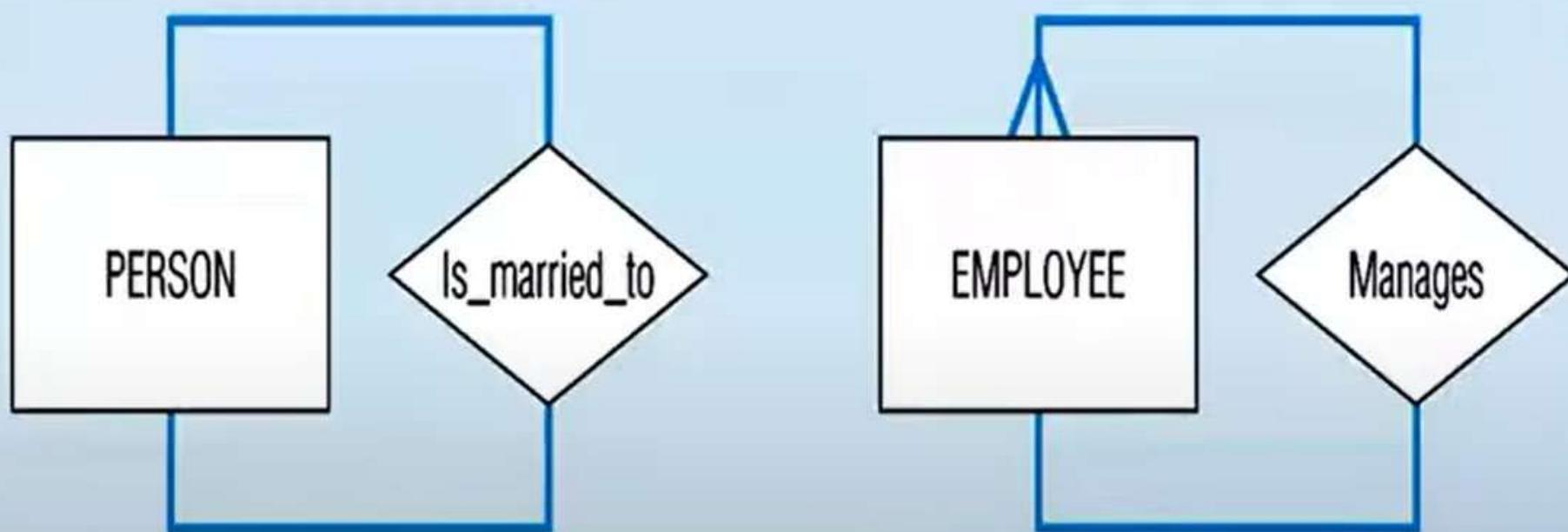
# Ternary Relationship

- ▶ ternary relationship set is of degree 3.



# Recursive Relationship (Unary)

- **Recursive Relationships** - A relationship in which the same entity participates more than once.



# Degree of Relationships

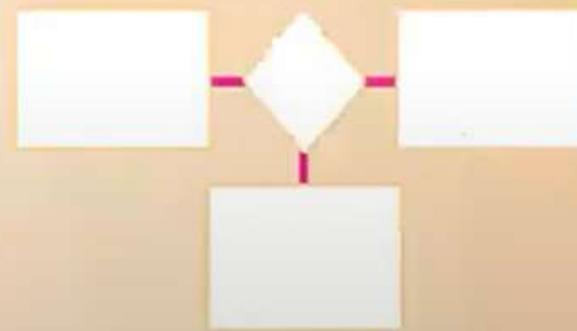
- Degree: number of entity types that participate in a relationship
- Three cases
  - **Unary:** between two instances of one entity type
  - **Binary:** between the instances of two entity types
  - **Ternary:** among the instances of three entity types



Unary



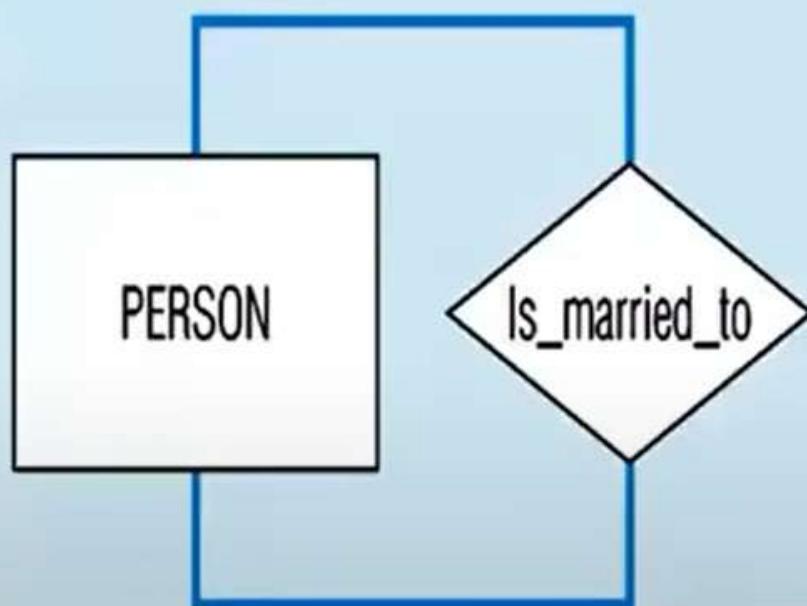
Binary



Ternary

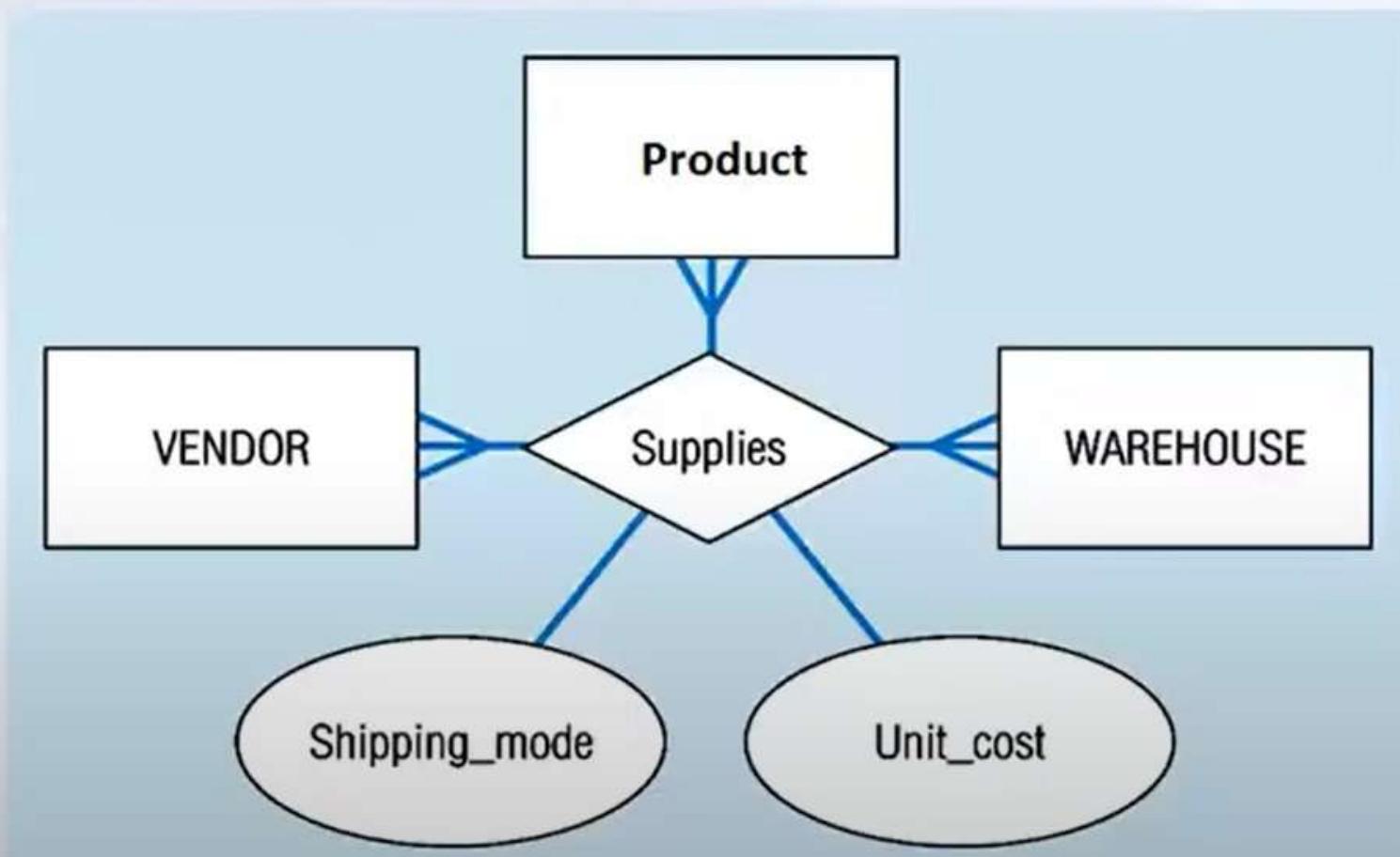
# Recursive Relationship (Unary)

- **Recursive Relationships** - A relationship in which the same entity participates more than once.

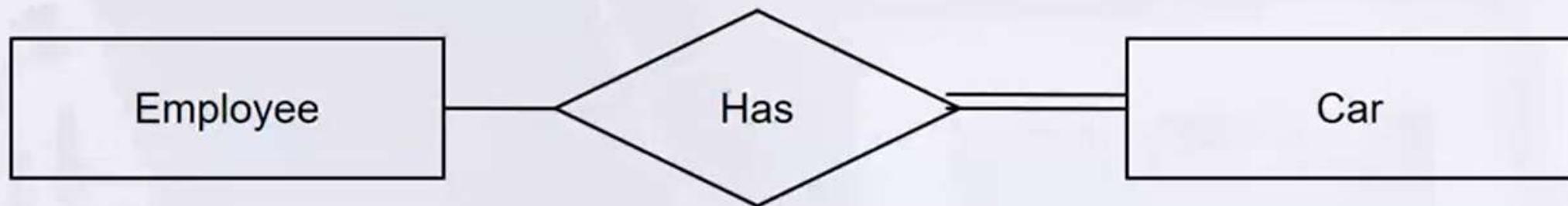


# Ternary Relationship

- ▶ ternary relationship set is of degree 3.



# PARTICIPATION CONSTRAINT



- An Employee **may** have a car.
- A Car **must** be assigned to particular employee

## 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

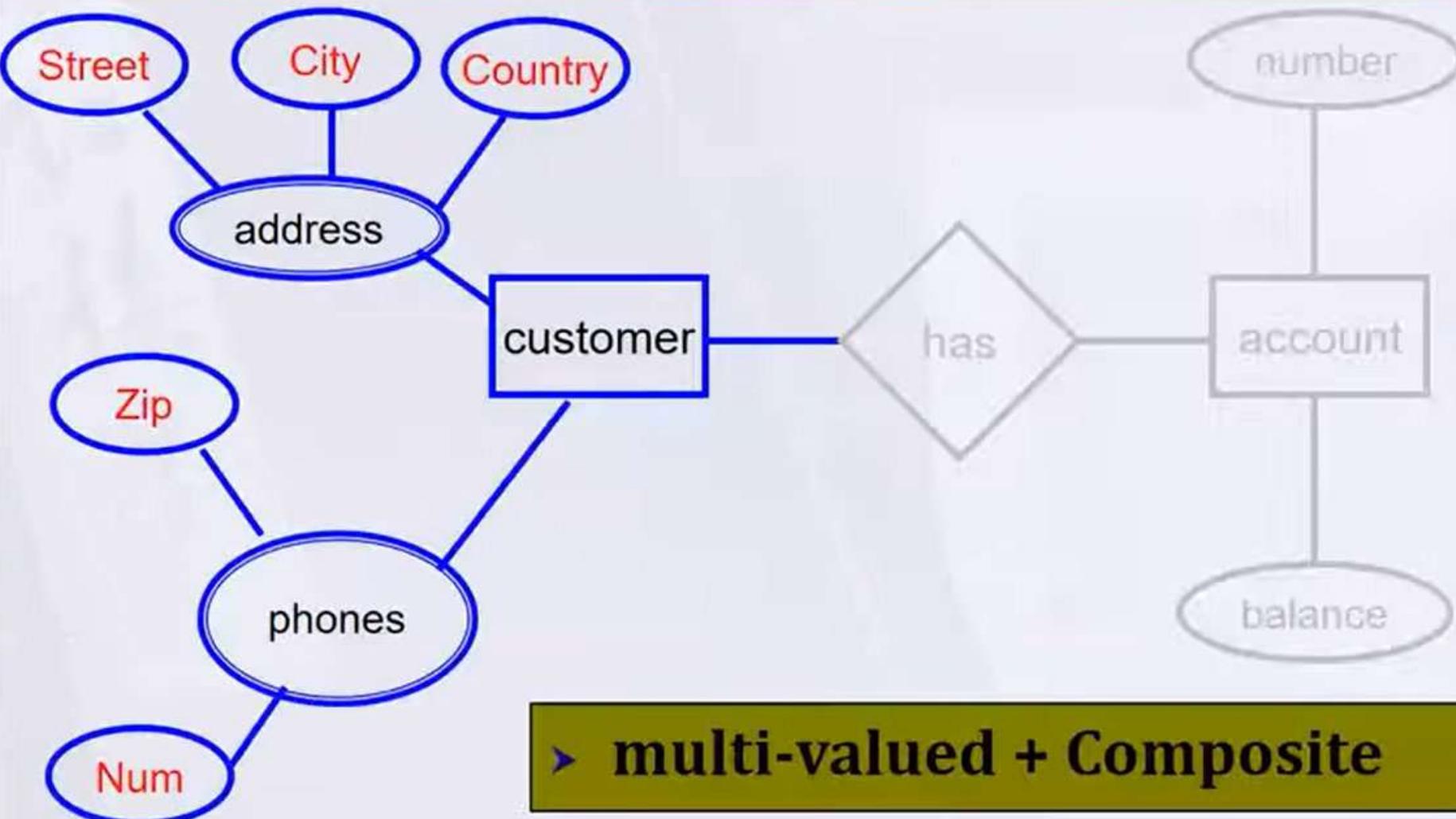
Only some employees manage departments  
The participation is PARTIAL

# PARTICIPATION CONSTRAINT



- A department **may** hire many employees (**Zero or more**)
    - An employee **must** be employed by a department
- (Department membership is **Optional**, Employee membership is **Mandatory**)

# Complex Attribute



# PARTICIPATION CONSTRAINT



- A department **may** hire many employees ( **Zero or more**)
  - An employee **must** be employed by a department
- (Department membership is **Optional**, Employee membership is **Mandatory**)

# PARTICIPATION CONSTRAINT



- A department **may** hire many employees ( **Zero or more**)
  - An employee **must** be employed by a department
- (Department membership is **Optional**, Employee membership is **Mandatory**)

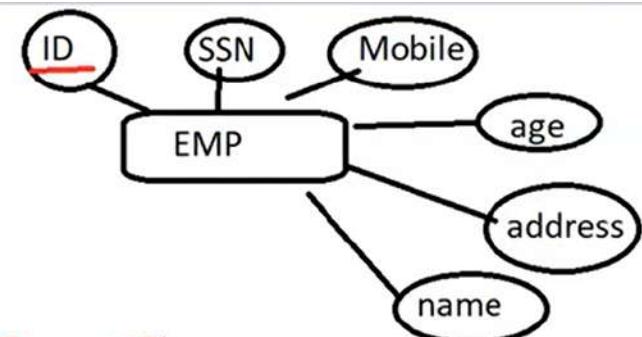
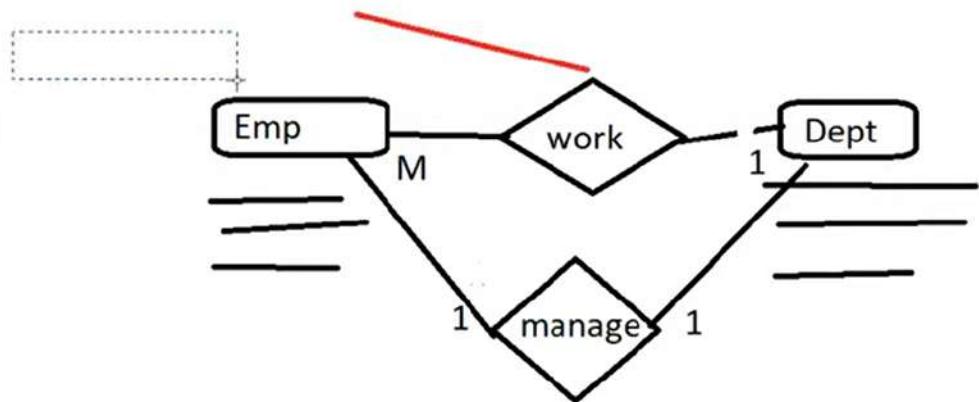
# PARTICIPATION CONSTRAINT



- A department **may** hire many employees (**Zero or more**)
  - An employee **must** be employed by a department
- (Department membership is **Optional**, Employee membership is **Mandatory**)

# Keys

- Different Types of Keys:
  1. Candidate Key
  2. Primary Key
  3. Foreign Key
  4. Composite Key
  5. Partial Key
  6. Alternate key
  7. Super Key



ID -----> Ok

SSN -----> OK

Mobile --> OK

# Candidate Key

**Candidate key:** is a set of one or more attributes whose value can uniquely identify an entity in the entity set

- Any attribute in the candidate key cannot be omitted without destroying the uniqueness property of the candidate key.

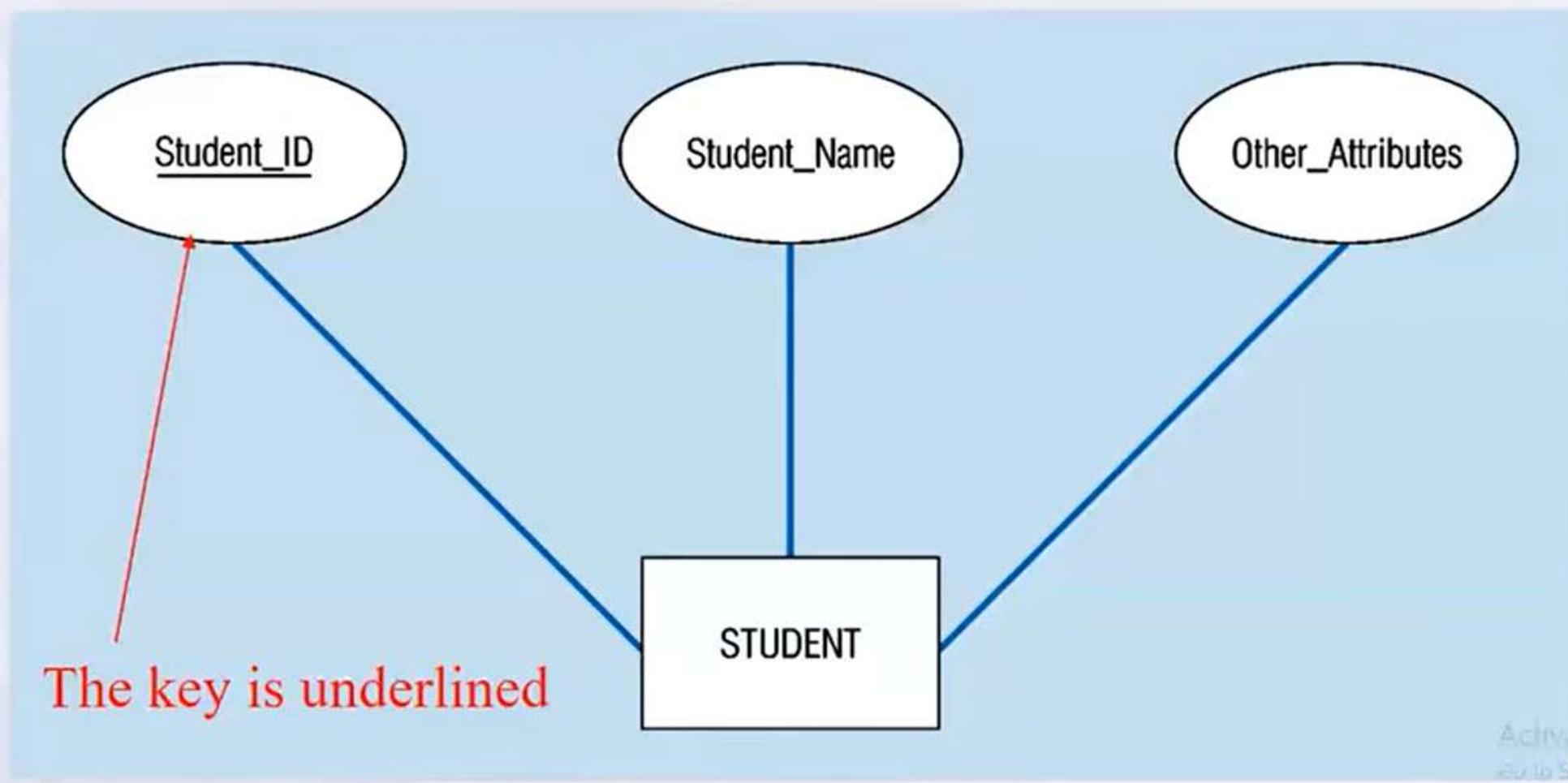
Example:

- $(SSN, Name)$  is NOT a candidate key .
- “ $SSN$ ” is a candidate key of *customer*.
- Candidate key could have more than one attributes.

# Primary Key

- **Example:** Both “SSN” and “License #” are candidate keys of *Driver* entity set.
- **Primary Key:** is the candidate key that is chosen by the database designer as the unique identifier of an entity.  
**[Unique & Not Null]**
- **Primary key May be Composite**

# Primary Key



## Case 1

**A big company has decided to store information about its projects and employees in a database. The company has wisely chosen to hire you as a database designer. Prepare an E-R diagram for this Company according to The following Description:**

- The company has a number of employees each employee has SSN, Birth Date, Gender and Name which represented as Fname and Lname.
- The company has a set of departments each department has a set of attributes DName, DNUM (unique) and locations.
- Employees work in several projects each project has Pname, PNumber as an identifier, Location and City.
- Each employee may have a set of dependents; each dependent has Dependent Name (unique), Gender, and Birth Date.  
Note: if the employee left the company no needs to store his dependents info
- For each Department, there is always one employee assigned to manage that Department and each manager has a hiring Date
- Department may have employees but employee must work on Only One department
- Each department may have a set of projects and each project must be assigned to one department
- Employees work in several projects and each project has several employees and each employee has a number of working hours in each project

**database designer. Prepare an E-R diagram for this Company according to  
The following Description:**

- The company has a number of employees each employee has SSN, Birth Date, Gender and Name which represented as Fname and Lname.
- The company has a set of departments each department has a set of attributes DName, DNUM (unique) and locations.
- Employees work in several projects each project has Pname, PNumber as an identifier, Location and City.
- Each employee may have a set of dependent; each dependent has Dependent Name (unique), Gender, and Birth Date.  
Note: if the employee left the company no needs to store his dependents info
- For each Department, there is always one employee assigned to manage that Department and each manager has a hiring Date
- Department may have employees but employee must work on Only One department
- Each department may have a set of projects and each project must assigned to one department
- Employees work in several projects and each project has several employees and each employee has a number of working hours in each project
- Each employee has a supervisor

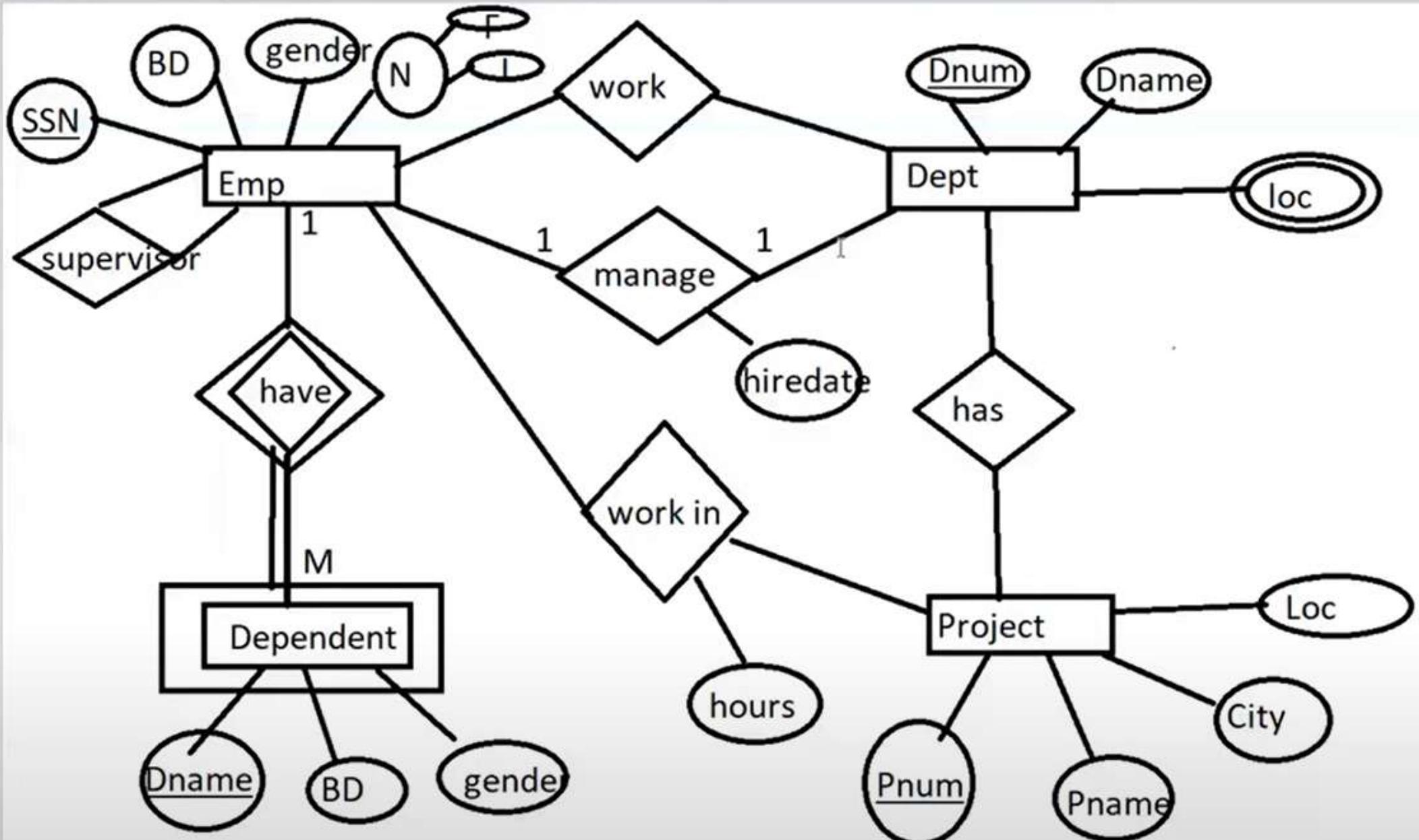
# Summary of notation for ER diagrams

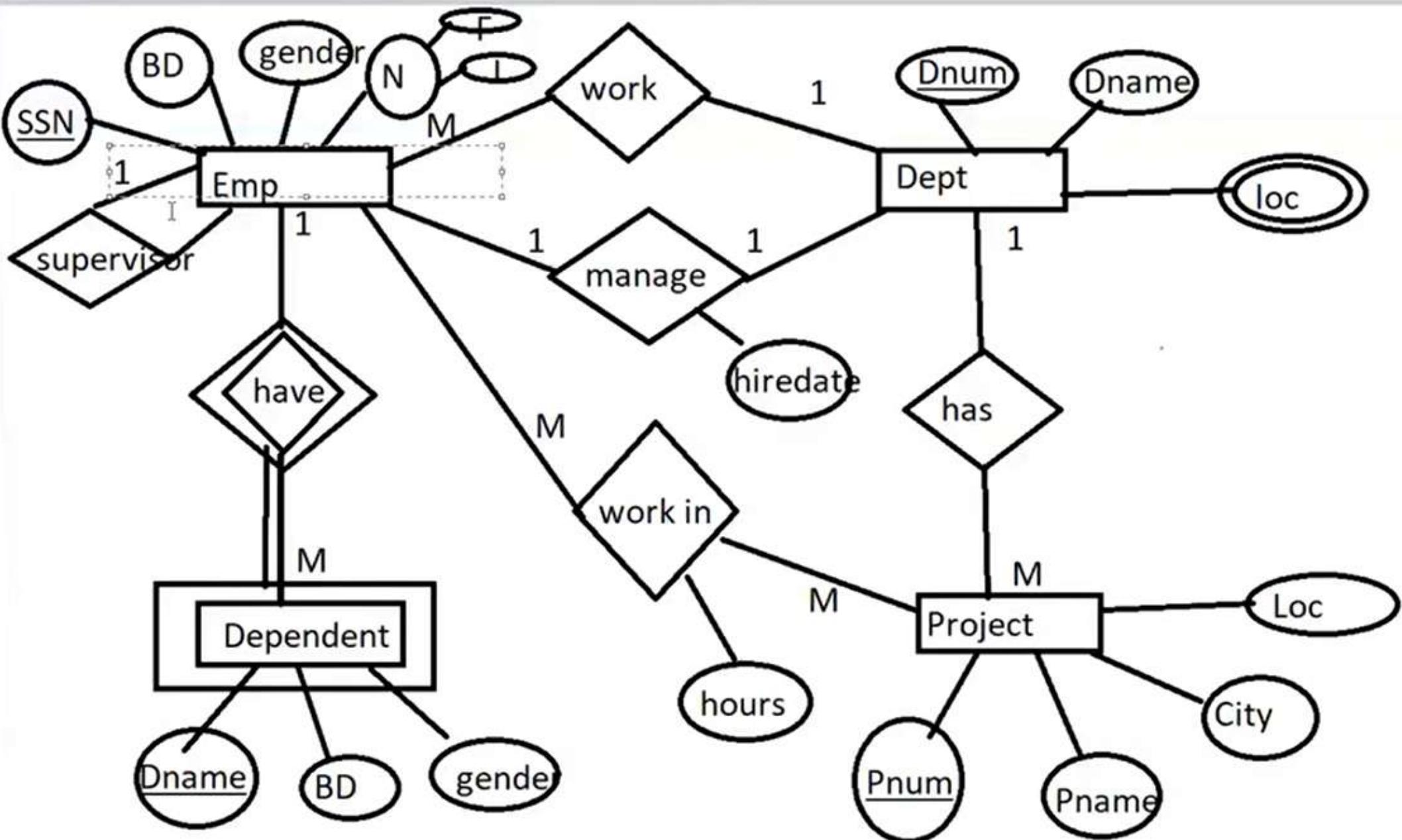
**Figure 3.14**  
Summary of the  
notation for ER  
diagrams.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1:N for $E_1:E_2$ in $R$
	Structural Constraint (min, max) on Participation of $E$ in $R$

**database designer. Prepare an E-R diagram for this Company according to  
The following Description:**

- The company has a number of employees each employee has SSN, Birth Date, Gender and Name which represented as Fname and Lname.
- The company has a set of departments each department has a set of attributes DName, DNUM (unique) and locations.
- Employees work in several projects each project has Pname, PNumber as an identifier, Location and City.
- Each employee may have a set of dependent; each dependent has Dependent Name (unique), Gender, and Birth Date.  
Note: if the employee left the company no needs to store his dependents info
- For each Department, there is always one employee assigned to manage that Department and each manager has a hiring Date
- Department may have employees but employee must work on Only One department
- Each department may have a set of projects and each project must assigned to one department
- Employees work in several projects and each project has several employees and each employee has a number of working hours in each project
- Each employee has a supervisor





**database designer. Prepare an E-R diagram for this Company according to  
The following Description:**

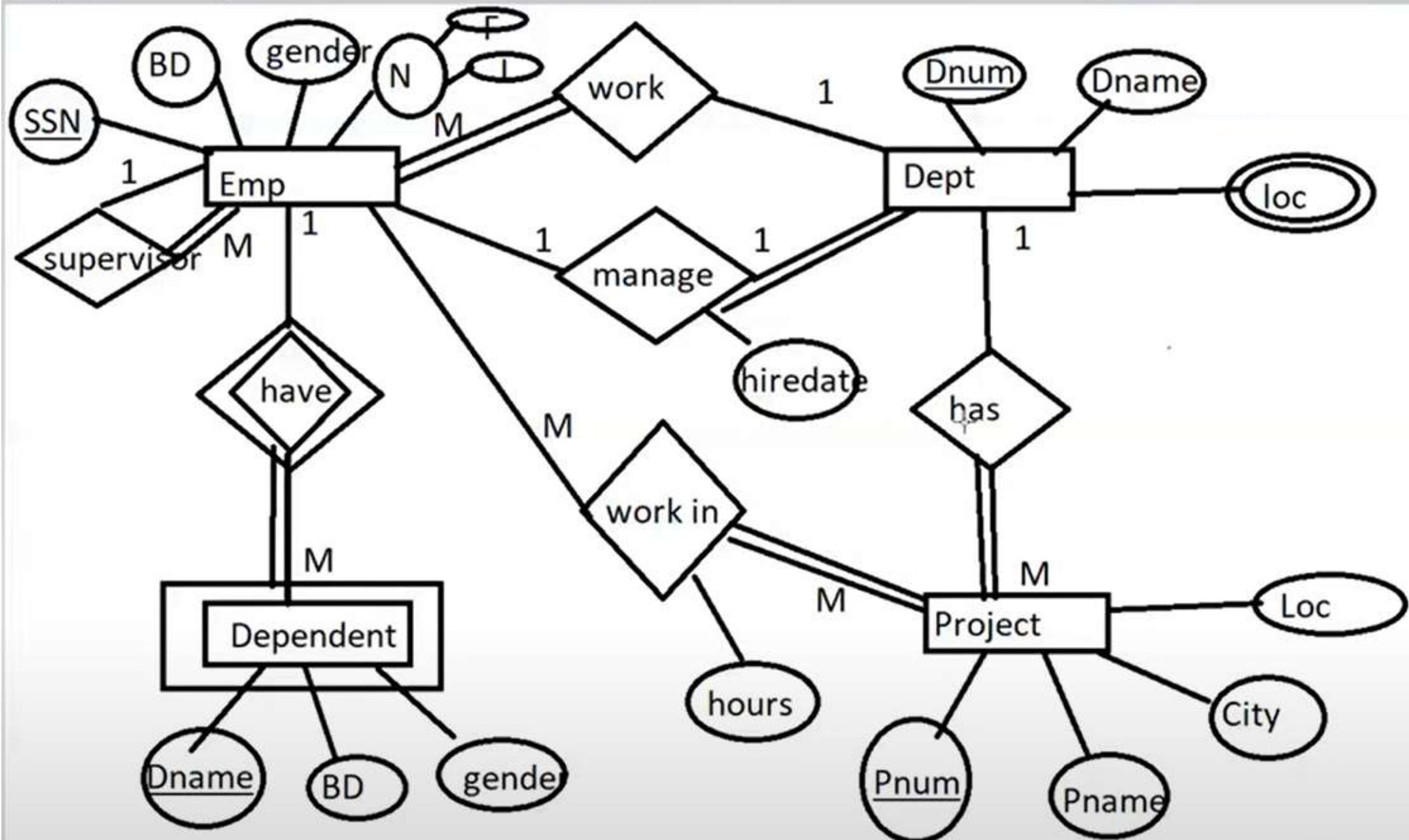
- The company has a number of employees each employee has SSN, Birth Date, Gender and Name which represented as Fname and Lname.
- The company has a set of departments each department has a set of attributes DName, DNUM (unique) and locations.
- Employees work in several projects each project has Pname, PNumber as an identifier, Location and City.
- Each employee may have a set of dependent; each dependent has Dependent Name (unique), Gender, and Birth Date.  
Note: if the employee left the company no needs to store his dependents info
- For each Department, there is always one employee assigned to manage that Department and each manager has a hiring Date
- Department may have employees but employee must work on Only One department
- Each department may have a set of projects and each project must assigned to one department
- Employees work in several projects and each project has several employees and each employee has a number of working hours in each project
- Each employee has a supervisor **and supervisor has many employees.**

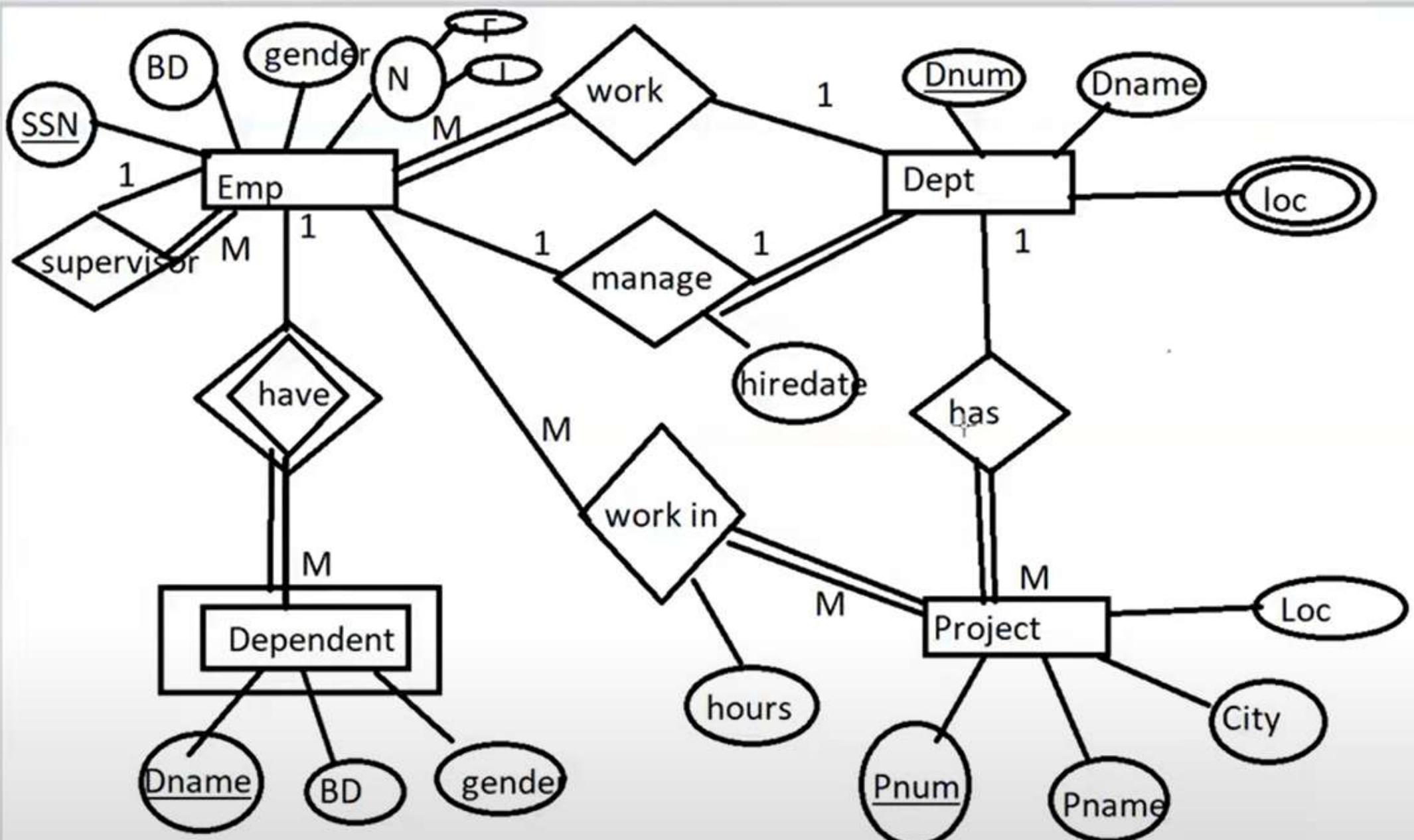
**database designer. Prepare an E-R diagram for this Company according to  
The following Description:**

- The company has a number of employees each employee has SSN, Birth Date, Gender and Name which represented as Fname and Lname.
- The company has a set of departments each department has a set of attributes DName, DNUM (unique) and locations.
- Employees work in several projects each project has Pname, PNumber as an identifier, Location and City.
- Each employee may have a set of dependent; each dependent has Dependent Name (unique), Gender, and Birth Date.  
Note: if the employee left the company no needs to store his dependents info
- For each Department, there is always one employee assigned to manage that Department and each manager has a hiring Date
- Department may have employees but employee must work on Only One department
- Each department may have a set of projects and each project must assigned to one department
- Employees work in several projects and each project has several employees and each employee has a number of working hours in each project
- Each employee has a supervisor **and supervisor has many employees.**

**database designer. Prepare an E-R diagram for this Company according to  
The following Description:**

- The company has a number of employees each employee has SSN, Birth Date, Gender and Name which represented as Fname and Lname.
- The company has a set of departments each department has a set of attributes DName, DNUM (unique) and locations.
- Employees work in several projects each project has Pname, PNumber as an identifier, Location and City.
- Each employee may have a set of dependent; each dependent has Dependent Name (unique), Gender, and Birth Date.  
Note: if the employee left the company no needs to store his dependents info
- For each Department, there is always one employee assigned to manage that Department and each manager has a hiring Date
- Department may have employees but employee must work on Only One department
- Each department may have a set of projects and each project must assigned to one department
- Employees work in several projects and each project has several employees and each employee has a number of working hours in each project
- Each employee has a supervisor





>><< على صوتك بالذاكرة >>

Schema is framework of tables.

Yes, a schema can be described as a framework of tables within a database.

---

## DB Life Cycle

1-Analysis --> System analyst  
Req Doc

2-DB Design ---> DB Designer  
ERD

3-DB Mapping --> DB Designer  
DB Schema (Tables& relationships)

4-DB Imp ---> DB Developer  
SQL (physical DB)

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

6-Client ---> EndUser

## ERD

## DB Day2

Entity ----- Strong Entity PK  
----- Weak Entity Partial Key

Attributes ----- Simple  
----- Composite  
----- Multivalued  
----- Computed  
-----Complex

## Relationship

Degree	Unary	Binary	Ternary
Cardinality	1	1	
	1	M	
	M	M	

Participation (total - Partial)

## Aggregate Functions

Count Min Max Sum Avg

Select **Sum(Salary)**  
from employee

62500

Select **Min(salary),Max(salary)**  
from employee

1000 9000

Count(eid) -----> 15

Count(\*) -----> 15

Count(ename) -----> 14

Select **Avg(salary)**  
from Employee

Select **Avg(isnull(salary,0))**  
from employee

Select **Sum(salary),did**  
from employee  
group by did

20000	10
28000	20
14500	30

Select **Count(eid),address**  
from employee  
group by address

6	cairo
5	alex
4	mansoura

Select **Sum(salary),did**  
from employee  
where address like '\_a%'  
group by did

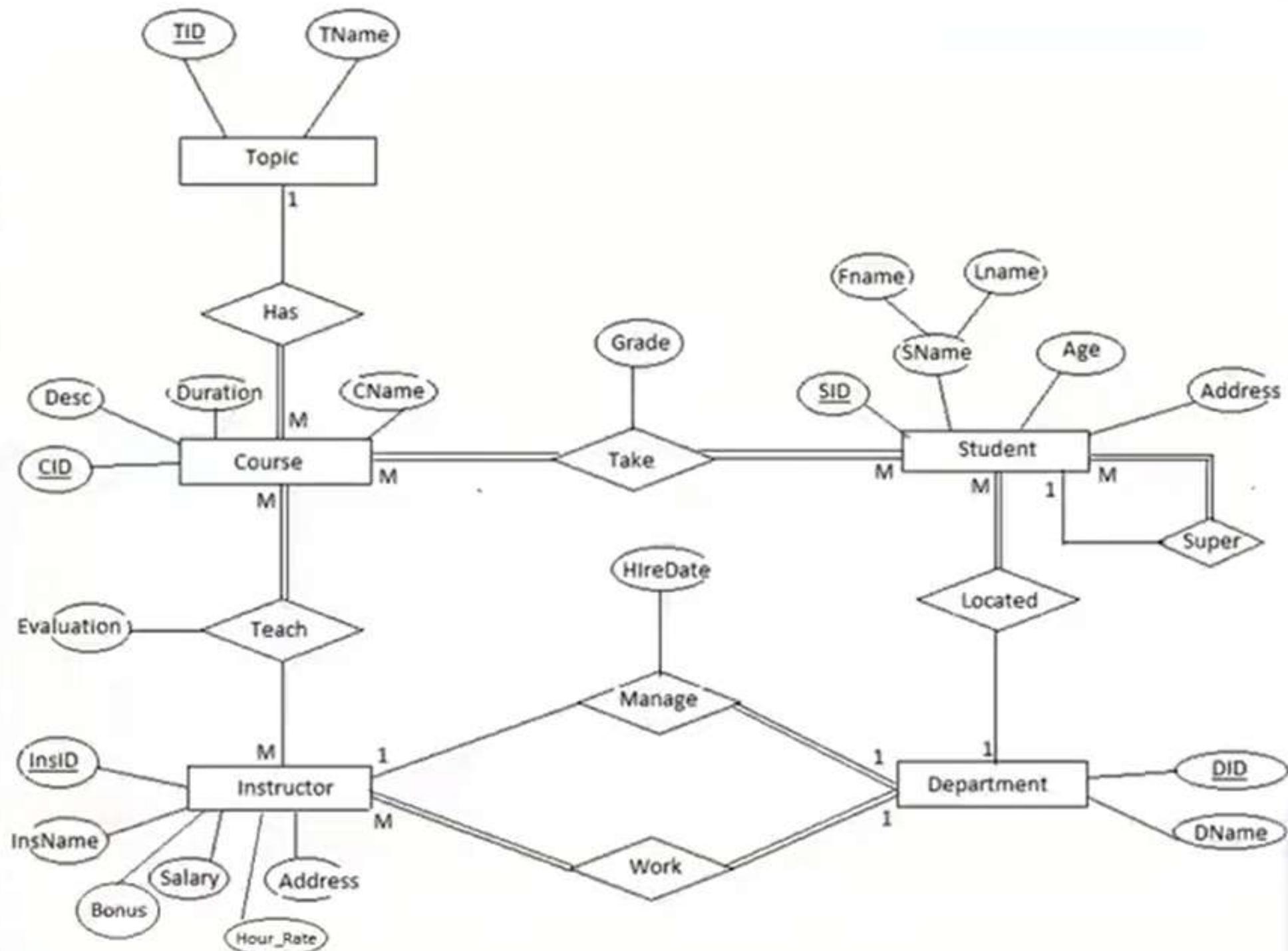
10000	10
15000	20
14500	30

Select **Count(eid),address**  
from employee  
where did in (10,30)  
group by address

3	cairo
3	alex
4	mansoura

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	<del>anser</del>	3000	mansoura	30

NULL



# Mapping Result

- Student(St\_id,st\_fname,st\_Lname,st\_age,st\_superDept\_ID)
- Course(Crs\_id.Crs\_Name,Crs\_Duration,Top\_id)
- Topic(Top\_ID,Top\_Name)
- Stud\_Course(St\_ID,Crs\_ID,grade)
- Instructor(Ins\_ID,ins\_Name,Address,Salary,Dept\_ID)
- Ins\_Course(Ins\_ID,Crs\_ID,Evalution)
- Department(Dept\_ID,Dept\_Name,Manager\_ID,HireDate)

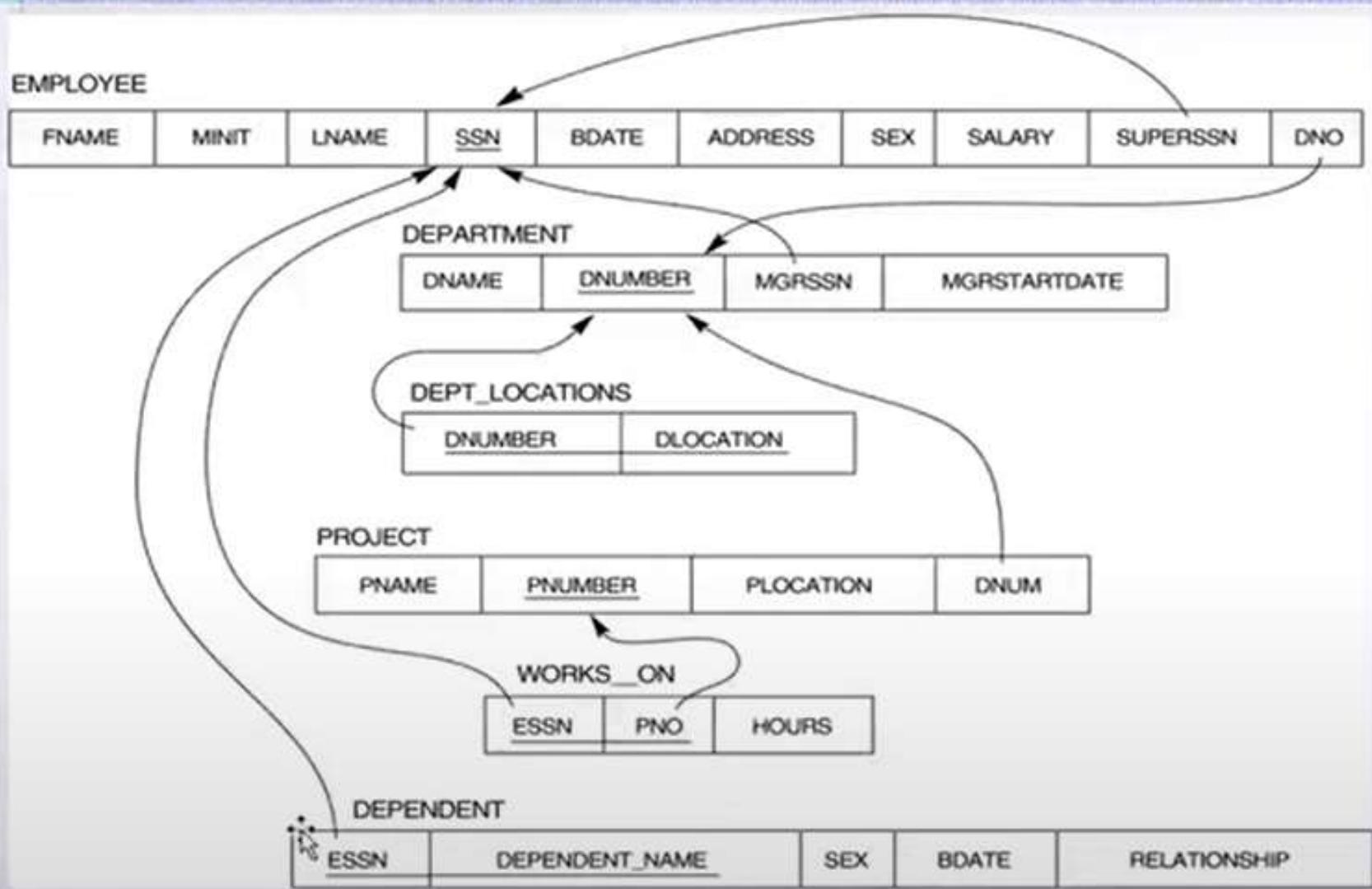
# Mapping Result

- Student(St\_id,st\_fname,st\_Lname,st\_age,st\_super,Dept\_ID)
- Course(Crs\_id,Crs\_Name,Crs\_Duration,Top\_id)
- Topic(Top\_ID,Top\_Name)
- Stud\_Course(St\_ID,Crs\_ID,grade)
- Instructor(Ins\_ID,ins\_Name,Address,Salary,Dept\_ID)
- Ins\_Course(Ins\_ID,Crs\_ID,Evalution)
- Department(Dept\_ID,Dept\_Name,Manager\_ID,HireDate)

# Mapping Result

- Student(St\_id,st\_fname,st\_Lname,st\_age,st\_super,Dept\_ID)
- Course(Crs\_id,Crs\_Name,Crs\_Duration,Top\_id)
- Topic(Top\_ID,Top\_Name)
- Stud\_Course(St\_ID,Crs\_ID,grade)
- Instructor(Ins\_ID,ins\_Name,Address,Salary,Dept\_ID)
- Ins\_Course(Ins\_ID,Crs\_ID,Evalution)
- Department(Dept\_ID,Dept\_Name,Manager\_ID,HireDate)

# Mapping Result



## DB Life Cycle

1-Analysis --> System analyst  
Req Doc

2-DB Design ---> DB Designer  
ERD

3-DB Mapping --> DB Designer  
DB Schema (Tables& relationships)

4-DB Imp ---> DB Developer  
SQL (physical DB)

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

6-Client ---> EndUser

## ERD

### DB Day2

Entity ----- Strong Entity PK  
----- Weak Entity Partial Key

Attributes ----- Simple  
----- Composite  
----- Multivalued  
----- Computed  
-----Complex

## Relationship

Degree	Unary
	Binary
	Ternary
Cardinality	1 1
	1 M
	M M

Participation (total - Partial)

## DB Mapping



Eid	Ename	Did
1	ahmed	10
2	ali	300
3	eman	10
		NULL

Did	Dname
10	SD
20	IS
30	CS

Activate Windows  
www.microsoft.com/activewindows

## DB Life Cycle

1-Analysis --> System analyst  
Req Doc

2-DB Design ---> DB Designer  
ERD

3-DB Mapping --> DB Designer  
DB Schema (Tables& relationships)

4-DB Imp ---> DB Developer  
SQL (physical DB)

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

6-Client ---> EndUser

## ERD

### DB Day2

#### Entity

----- Strong Entity PK  
----- Weak Entity Partial Key

#### Attributes

----- Simple  
----- Composite  
----- Multivalued  
----- Computed  
-----Complex

#### Relationship

Degree      Unary  
              Binary  
              Ternary  
Cardinality    1    1  
               1    M  
              M    M

#### Participation (total - Partial)

## DB Mapping



foreign key

Eid	Ename	Did
1	ahmed	10
2	ali	300
3	eman	10
		NULL

Did	Dname
10	SD Del XX
20	IS Del ok
30	CS

M  
Child

1  
Parent

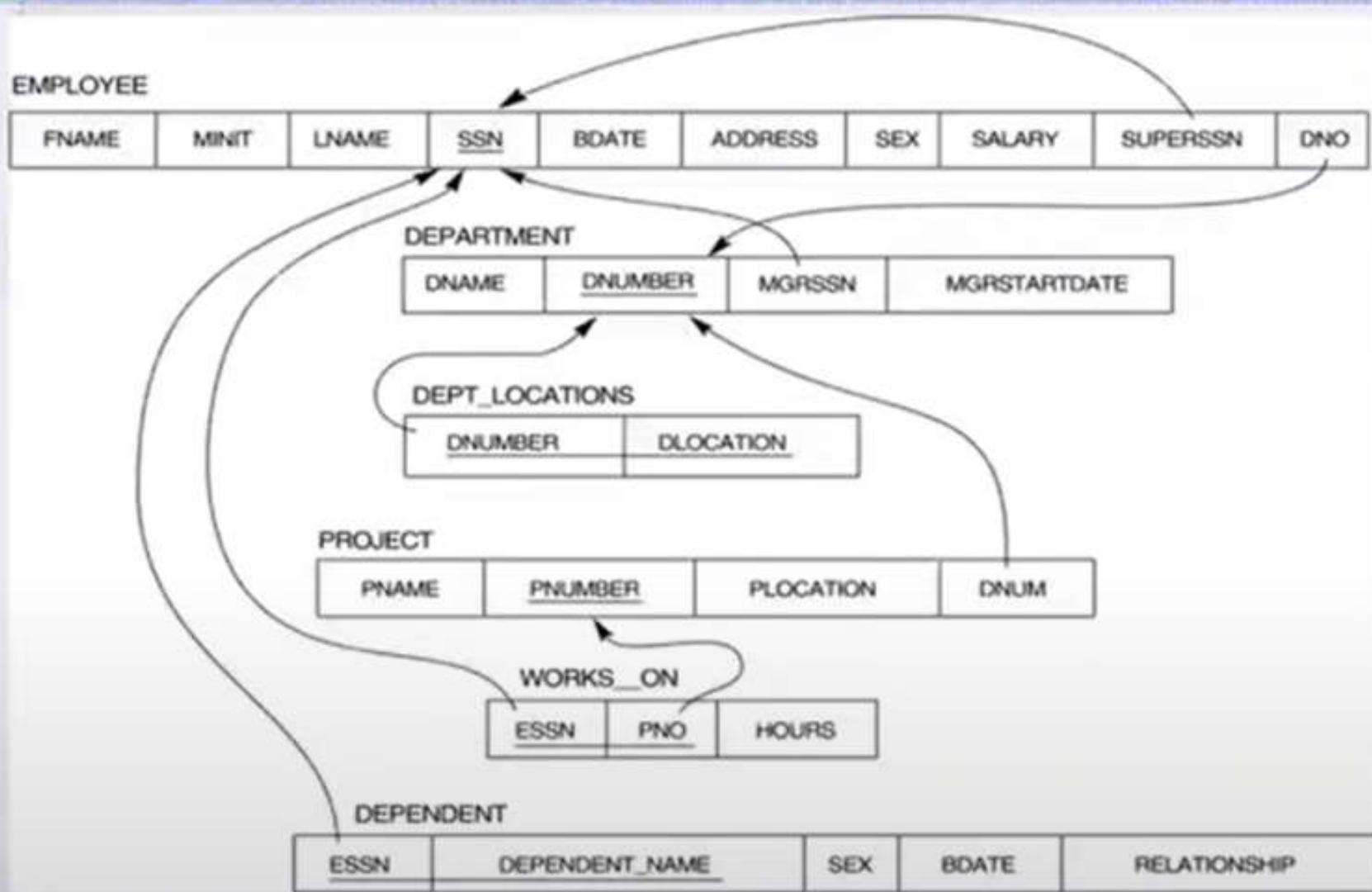
Activate Windows  
Goto: Settings > Activation > Windows

## Summary of notation for ER diagrams

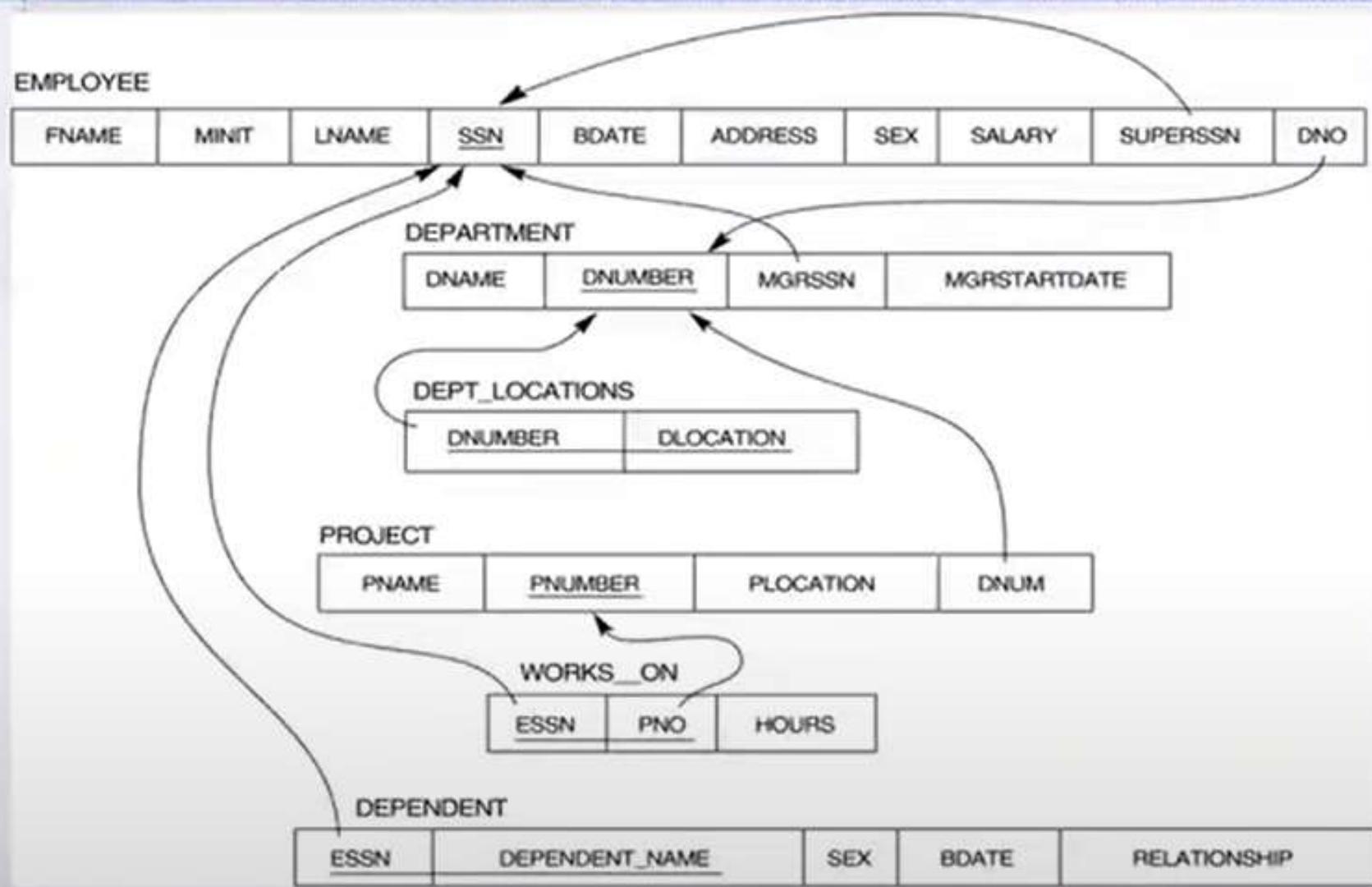
**Figure 3.14**  
Summary of the  
notation for ER  
diagrams.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1: N for $E_1/E_2$ in $R$
	Structural Constraint (min, max) on Participation of $E$ in $R$

# Mapping Result



# Mapping Result



# Summary of notation for ER diagrams

**Figure 3.14**  
Summary of the  
notation for ER  
diagrams.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1:N for $E_1:E_2$ in $R$
	Structural Constraint (min, max) on Participation of $E$ in $R$

# Summary of notation for ER diagrams

**Figure 3.14**  
Summary of the notation for ER diagrams.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1: N for $E_1:E_2$ in $R$
	Structural Constraint (min, max) on Participation of $E$ in $R$

# Relational Database Definitions

- Table or entity: a collection of records
- Attribute or Column or field: a characteristic of an entity
- Row or Record or tuple: the specific characteristics of one entity
- Database: a collection of tables

# Relational Database

The diagram illustrates a relational database table structure. The table has columns labeled "SSAN", "Name", and "Date of Birth". The first row contains the header labels. The second row contains the value "1/1/2012" under the "Date of Birth" column. The third row is highlighted with gray shading. The fourth row contains the value "31/12/2012" under the "Date of Birth" column. Annotations with arrows point to specific parts of the table:

- A horizontal arrow points to the left side of the table, labeled "Relation".
- A vertical arrow points down to the "Date of Birth" column, labeled "Column".
- A horizontal arrow points to the third row, labeled "Tuple".
- A vertical arrow points to the "SSAN" column, labeled "SSAN is a key".

SSAN	Name	Date of Birth			
		1/1/2012			
		31/12/2012			

## DB Life Cycle

1-Analysis --> System analyst  
Req Doc

2-DB Design --> DB Designer  
ERD

3-DB Mapping --> DB Designer  
DB Schema (Tables& relationships)

4-DB Imp --> DB Developer  
SQL (physical DB)

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

6-Client --> EndUser

## ERD

### DB Day2

Entity ----- Strong Entity PK  
----- Weak Entity Partial Key

Attributes ----- Simple  
----- Composite  
----- Multivalued  
----- Computed  
-----Complex

## Relationship

Degree Unary  
Binary  
Ternary  
Cardinality 1 1  
1 M  
M M

Participation (total - Partial)

## DB Mapping



foreign key

Y int

X int

Eid	Ename	Did	Did	Dname
1	ahmed	10	10	SD Del XK
2	ali	300	20	IS Del ok
3	eman	10	30	CS
		NULL		

M  
Child

1  
Parent

Activate Windows

Go to Settings Account Windows

Column

---- Domain

----- DataType

----- Quality

----- Size

----- tinyint

----- 1B

----- -128:127

----- Age

----- Constraints & Rules

# Mapping ->DB Tables

## CUSTOMER

<u>Customer_ID</u>	Customer_Name	Address	City	State	Zip
--------------------	---------------	---------	------	-------	-----

## ORDER

<u>Order_ID</u>	Order_Date	<u>Customer_ID</u>
-----------------	------------	--------------------

## ORDER LINE

<u>Order_ID</u>	<u>Product_ID</u>	Quantity
-----------------	-------------------	----------

## PRODUCT

<u>Product_ID</u>	Product_Description	Product_Finish	Standard_Price	On_Hand
-------------------	---------------------	----------------	----------------	---------

# Mapping ->DB Tables

CUSTOMER

Customer_ID	Customer_Name	Address	City	State	Zip
-------------	---------------	---------	------	-------	-----

ORDER

Order_ID	Order_Date	Customer_ID
----------	------------	-------------

Primary Key

Foreign Key

ORDER LINE

Order_ID	Product_ID	Quantity
----------	------------	----------

composite primary key

PRODUCT

Product_ID	Product_Description	Product_Finish	Standard_Price	On_Hand
------------	---------------------	----------------	----------------	---------

# ER-to-Relational Mapping

Step 1: Mapping of Regular 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.

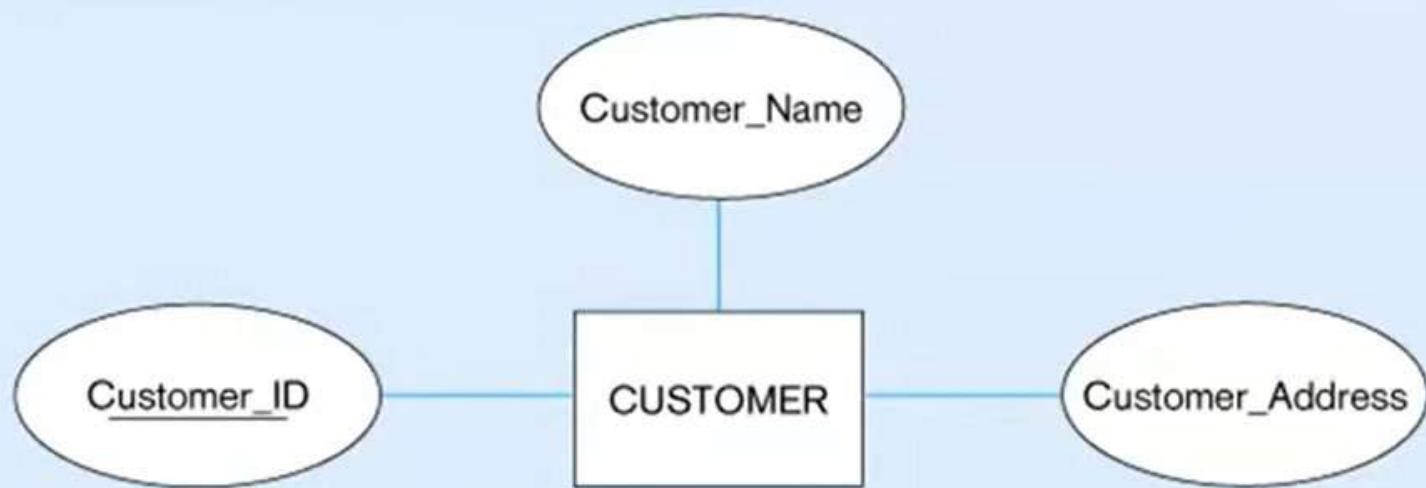
Step 7: Mapping of Unary Relationship.

## Step 1: Mapping of Regular Entity Types

- Create table for each entity type -> if there is no 1-1 relationship mandatory from 2 sides
- Choose one of key attributes to be the primary key

# Mapping Regular entity

(a) CUSTOMER  
entity type with  
simple  
attributes



(b) CUSTOMER relation

CUSTOMER

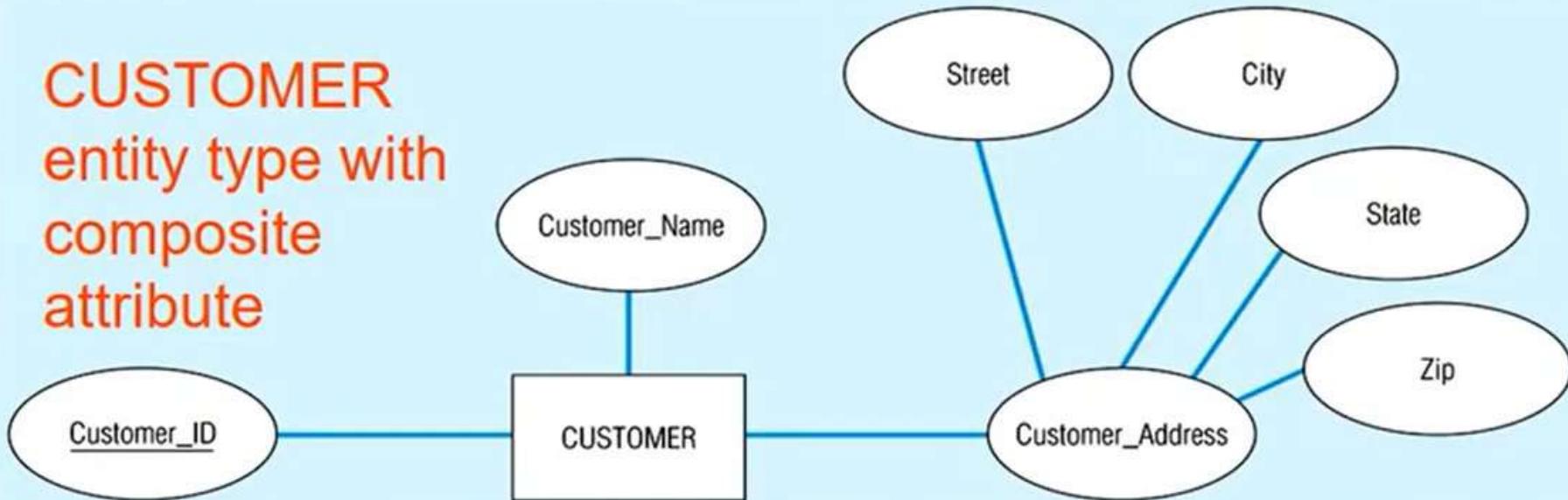
Customer\_ID

Customer\_Name

Customer\_Address

# Mapping Composite attribute

CUSTOMER  
entity type with  
composite  
attribute

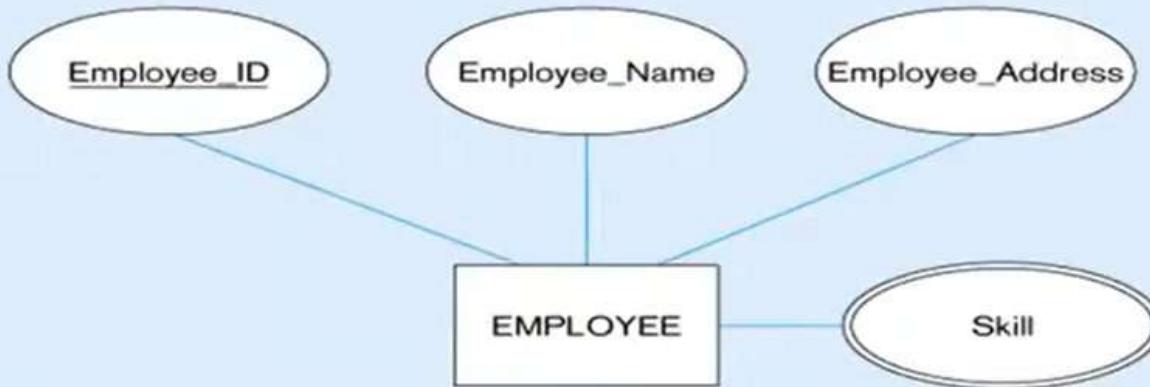


CUSTOMER relation with address detail

CUSTOMER

<u>Customer_ID</u>	Customer_Name	Street	City	State	Zip

# Mapping Multivalued Attribute



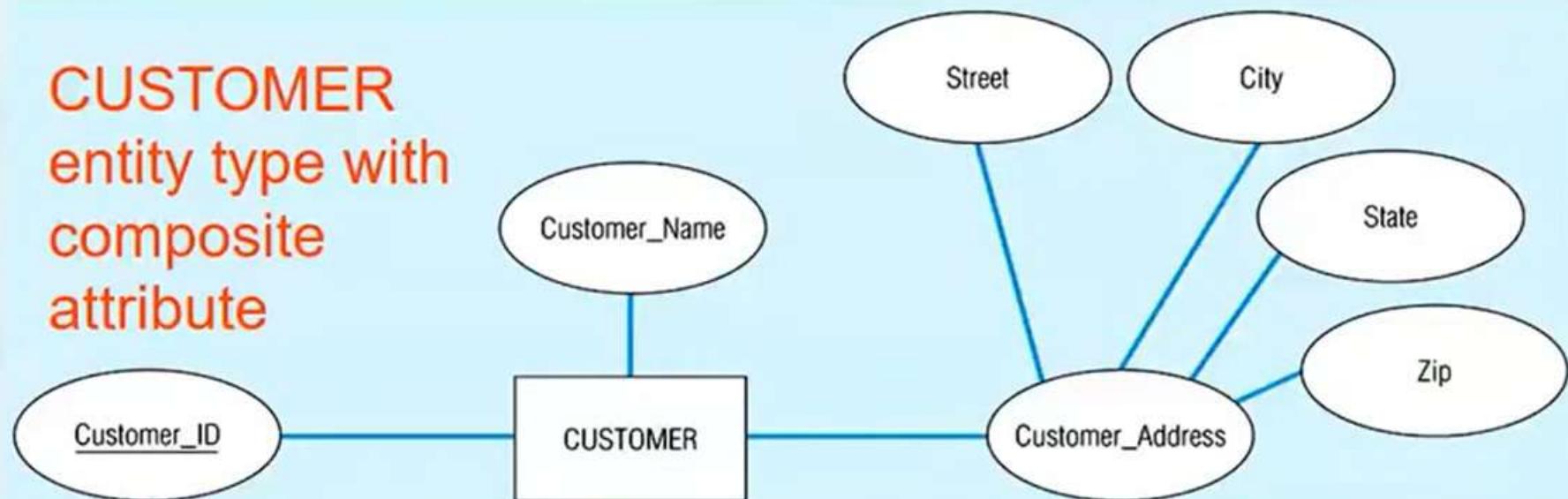
Multivalued attribute becomes a separate relation with foreign key



Activate Wi  
Go to Settings t

# Mapping Composite attribute

CUSTOMER  
entity type with  
composite  
attribute



CUSTOMER relation with address detail

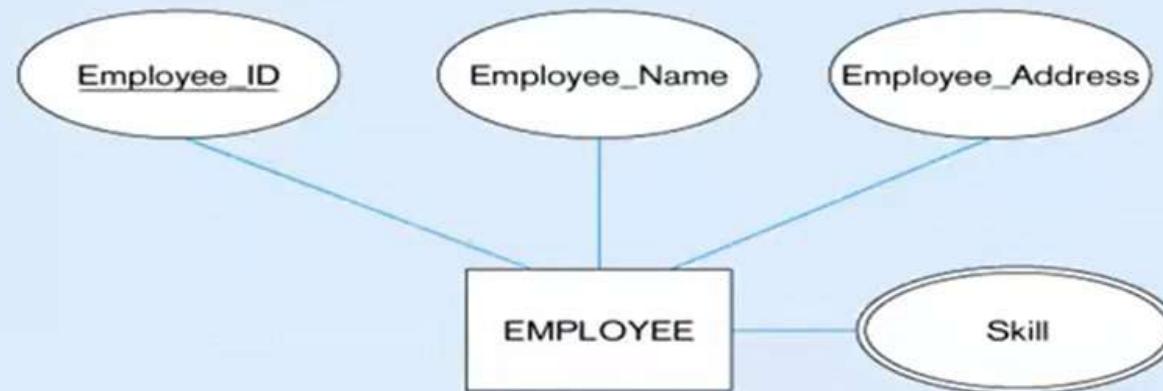
CUSTOMER

<u>Customer_ID</u>	Customer_Name	Street	City	State	Zip

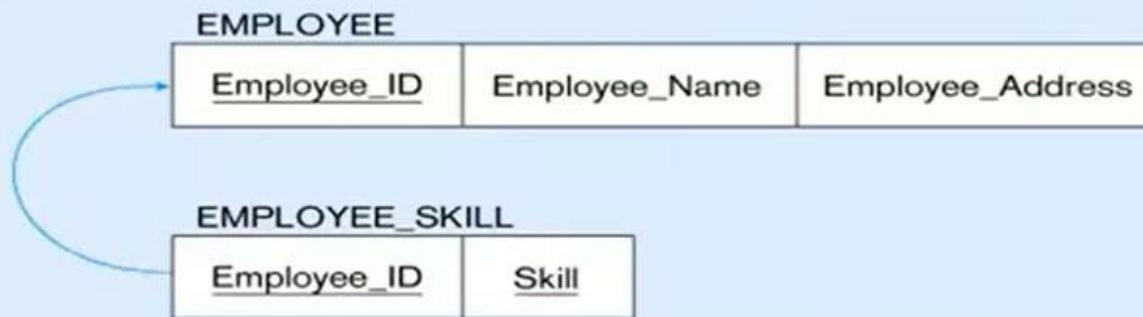
Activate Wi  
Go to Settings

Activ

# Mapping Multivalued Attribute



Multivalued attribute becomes a separate relation with foreign key



## 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

## 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

Column

---- Domain

----- DataType

----- Quality

----- Size

----- tinyint

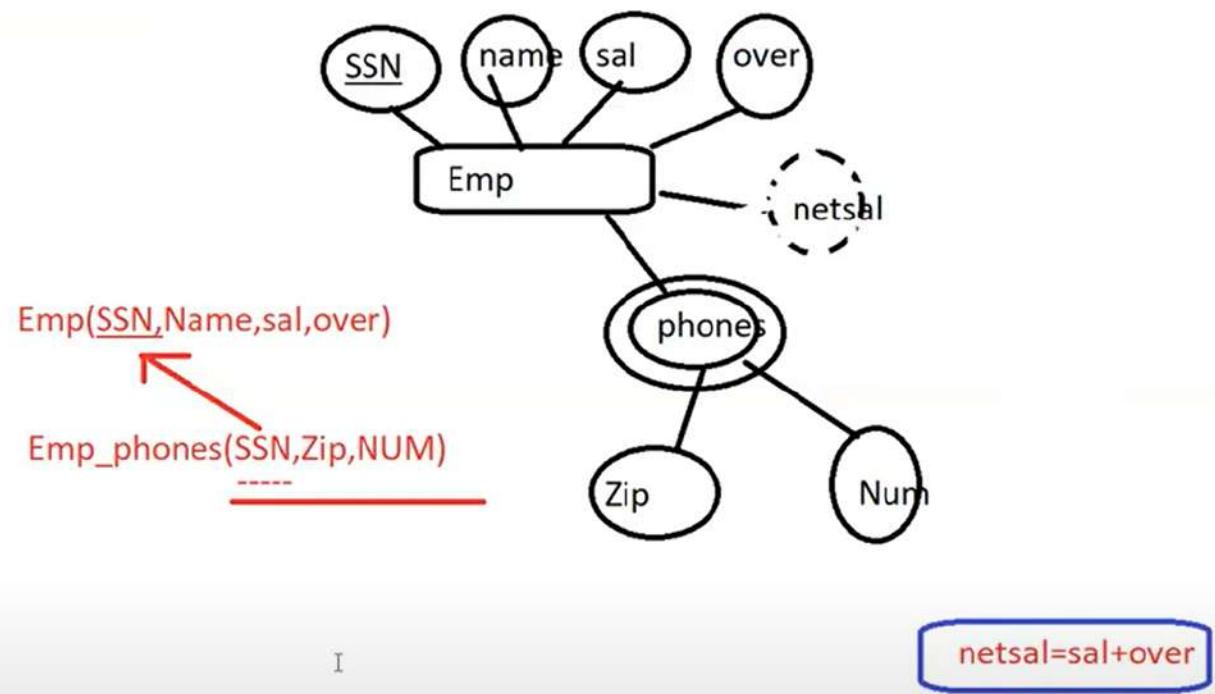
----- 1B

----- -128:127

----- Age

----- Constraints & Rules

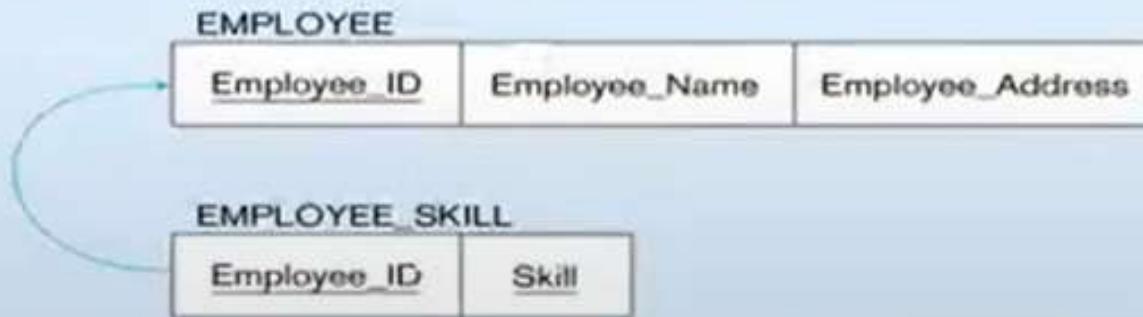
Column  
 ---- Domain  
 ----- DataType  
 ----- Quality  
 ----- Size  
 ----- tinyint  
 ----- 1B  
 ----- 128:127  
 ----- Age  
 ----- Constraints & Rules



# Mapping Multivalued Attribute

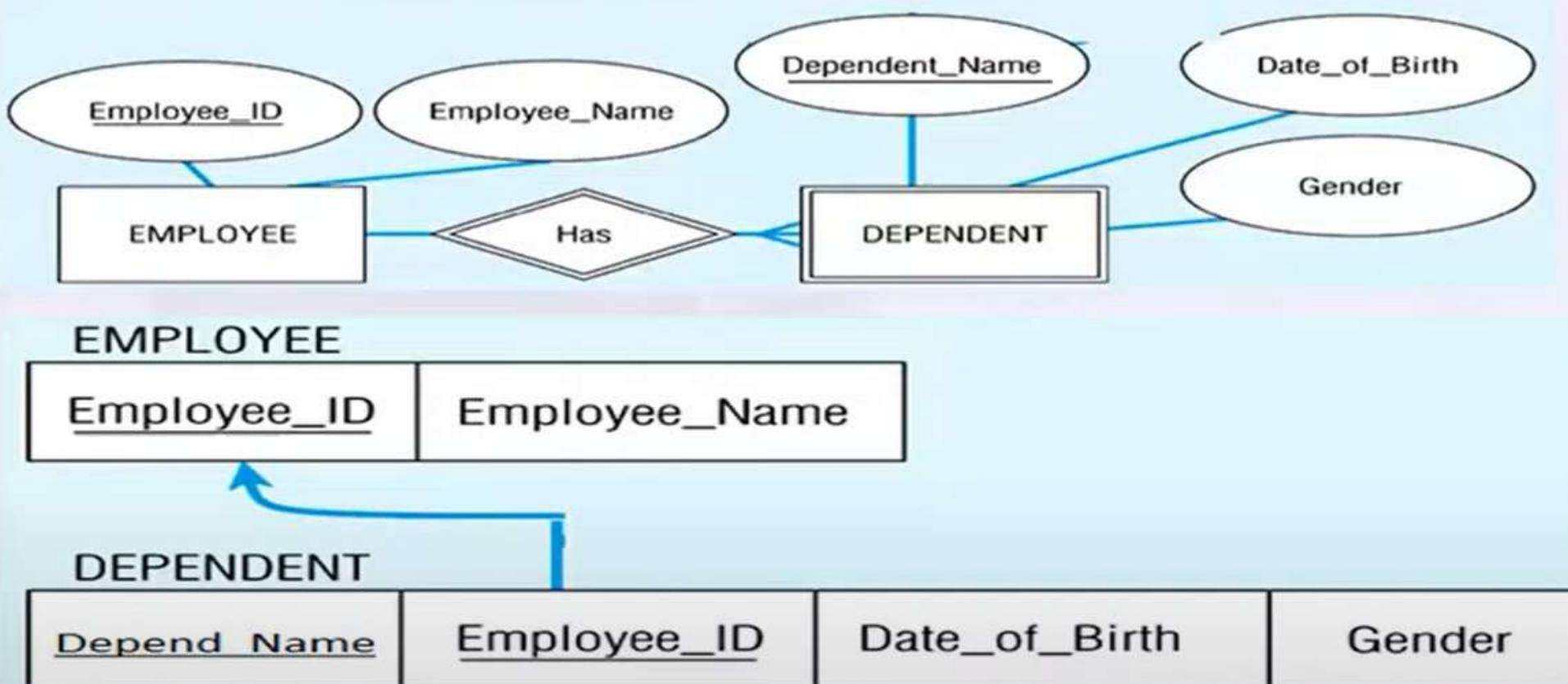


Multivalued attribute becomes a separate relation with foreign key



I  
1 – to – many relationship between original entity and new relation

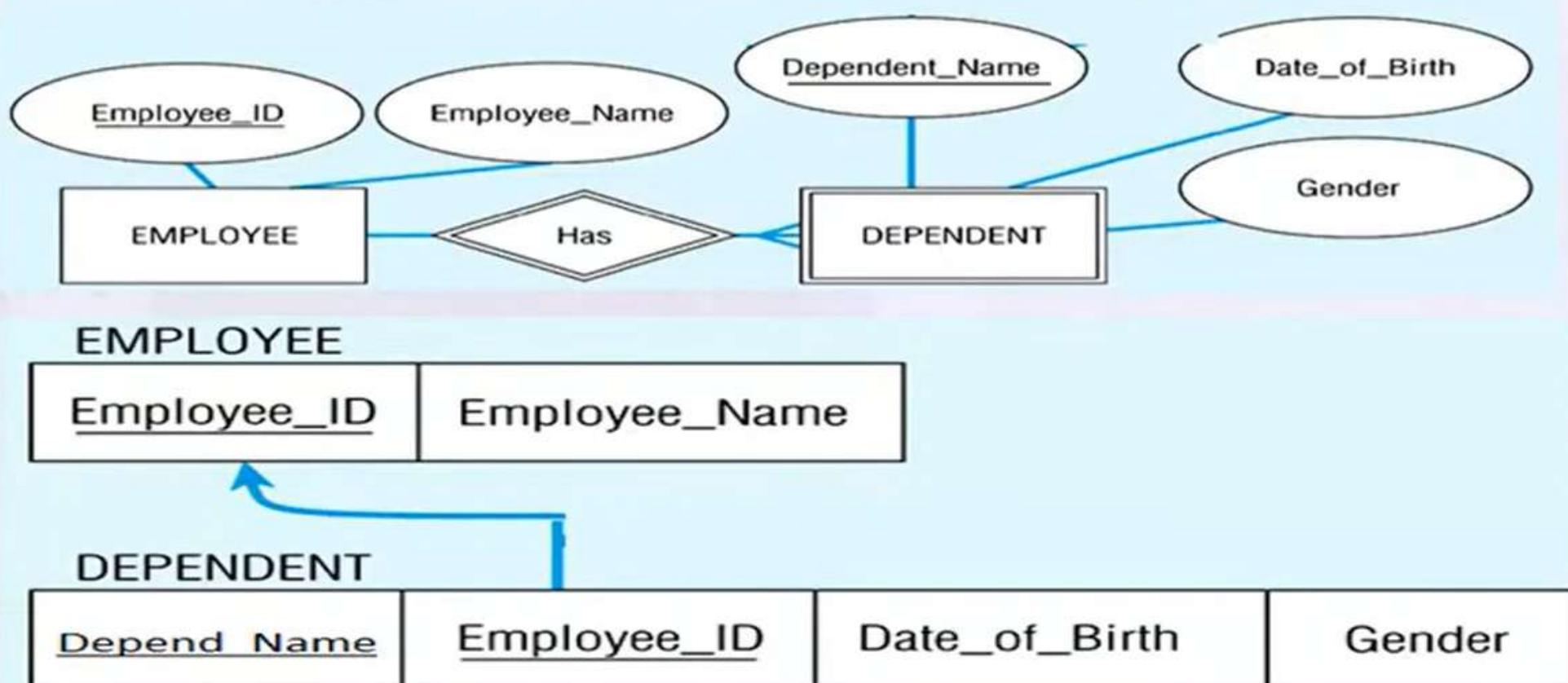
# Mapping Weak entity



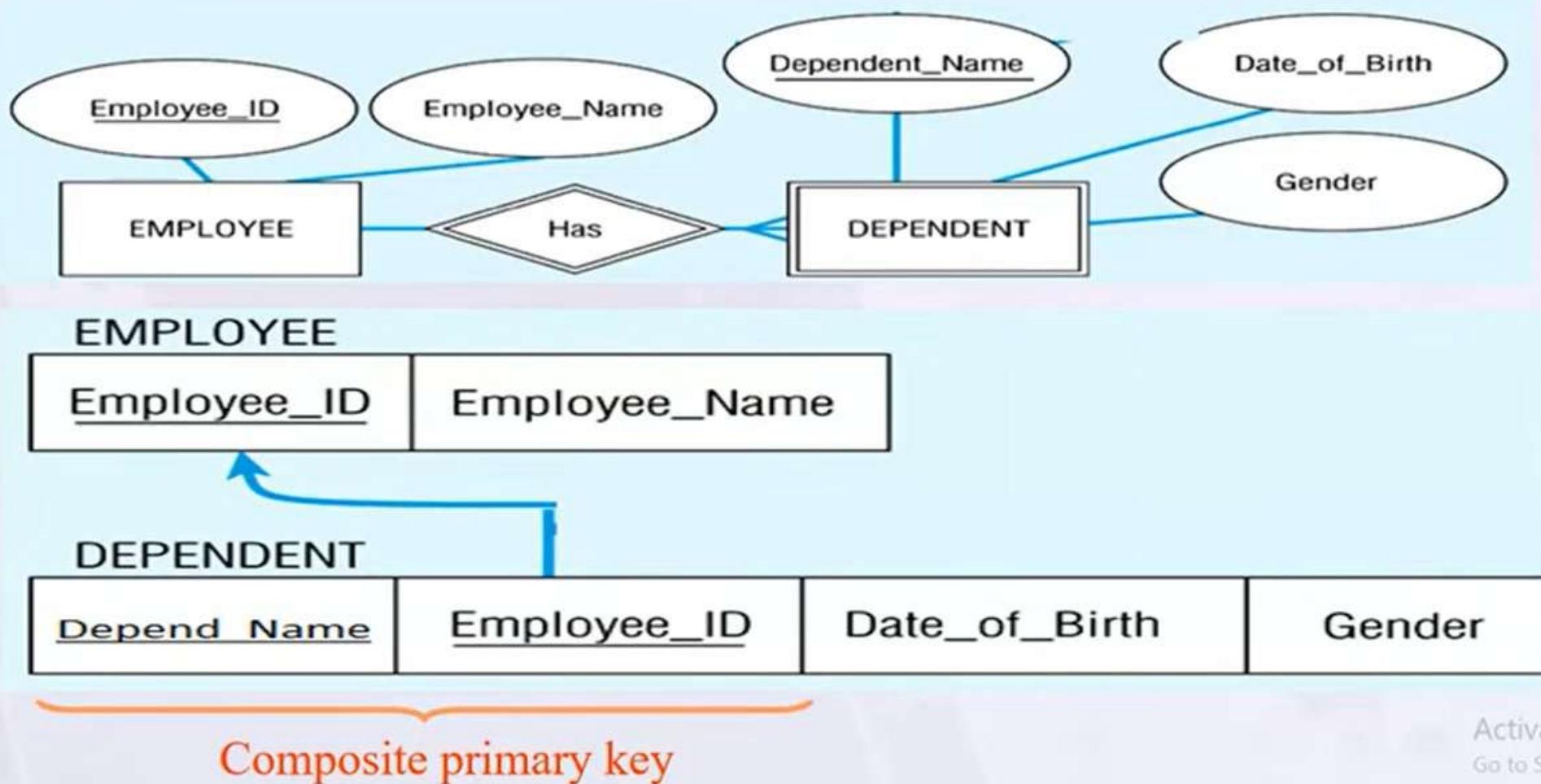
## Step 2: Mapping of Weak Entity Types

- Create table for each weak entity.
- Add foreign key that correspond to the owner entity type.
- Primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)

# Mapping Weak entity



# Mapping Weak entity



## Step 3: Mapping of Binary 1:1 Relation Types

- Merged two tables if both sides are Mandatory.
- Add FK into table with the total participation relationship to represent optional side.
- Create third table if both sides are optional.

## Step 3: Mapping of Binary 1:1 Relation Types

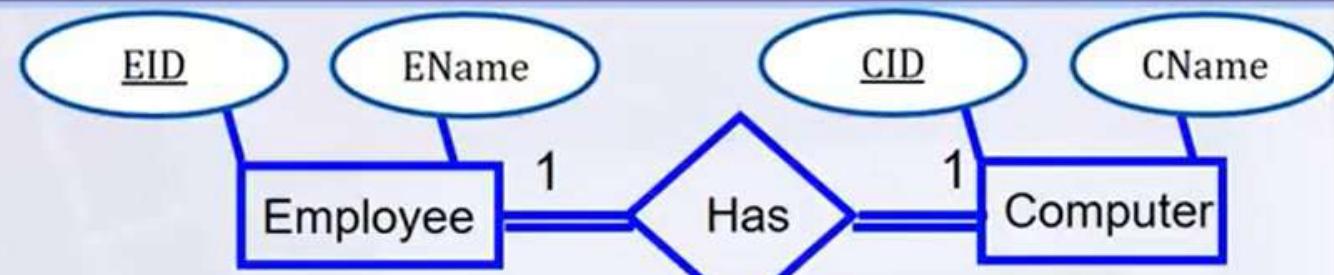
- Merged two tables if both sides are Mandatory.
- Add FK into table with the total participation relationship to represent optional side.
- Create third table if both sides are optional.

## 2 Mandatory

One-to-One  
2 Mandatory



**1 table**  
tbl\_xy (PK,.....,.....)  
PK = PKx or PKy

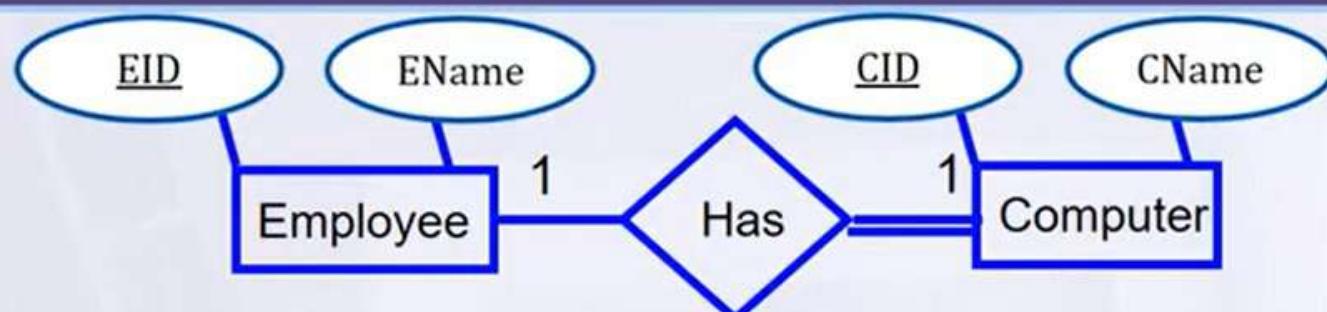


Emp(EID, Ename, Cname, **CID**)

# Optional-Mandatory

**One-to-One**

X optional – Y mandatory



**2 tables**

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....,PKx....)

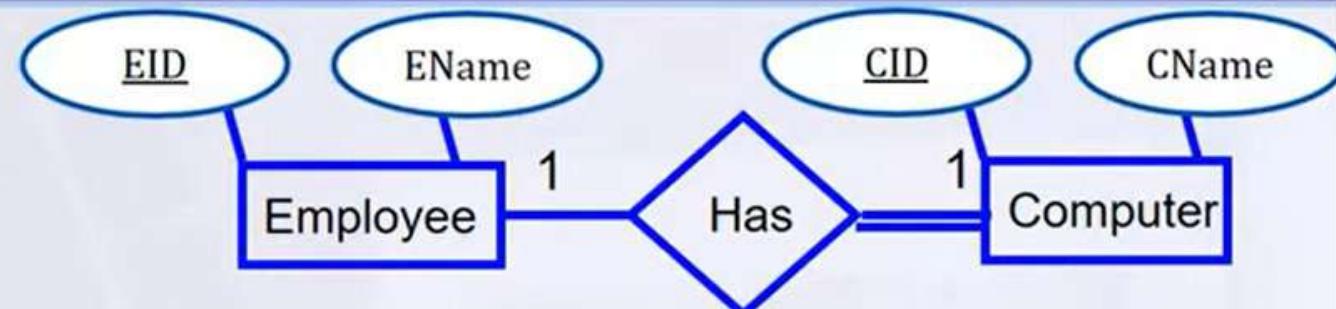
Employee(EID, Ename)

Computer(CID, Cname, EID\_FK)

# Optional-Mandatory

**One-to-One**

X optional – Y mandatory



**2 tables**

tbl\_x (PK<sub>x</sub>,.....,.....)

tbl\_y (PK<sub>y</sub>,.....,.....,PK<sub>x</sub>....)

Employee(EID, Ename)

Computer(CID, Cname, **EID\_FK**)

# 2 Optional

One-to-One

2 Optional



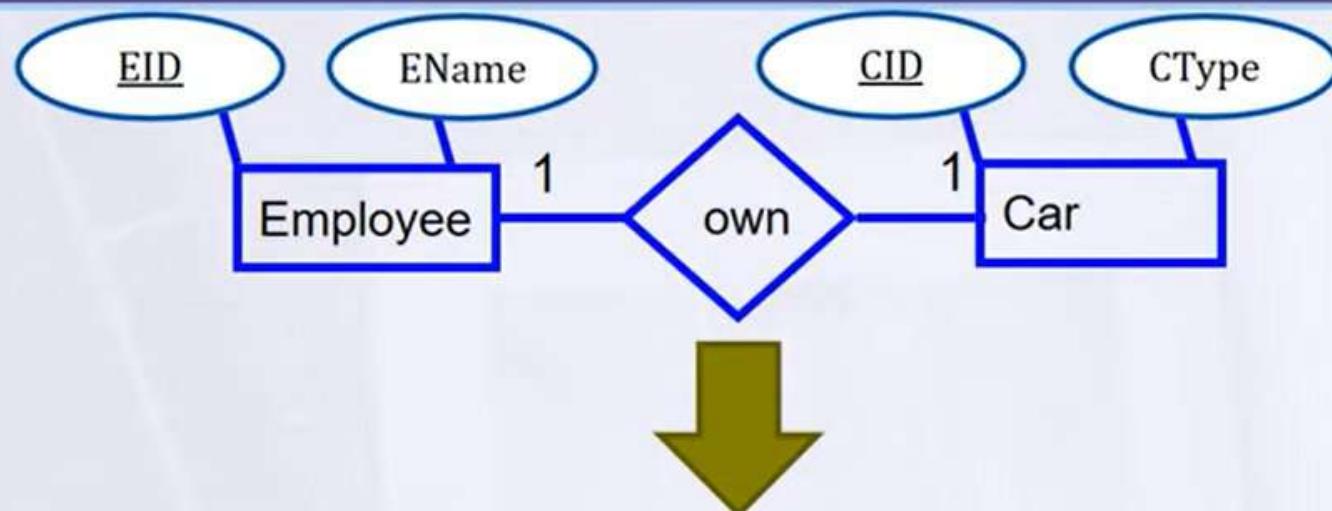
3 tables

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....)

tbl\_xy (PKxy,.....,...,FKxy,....)

PKxy = PKx or PKy



Employee(EID, Ename)

Car(CID, CType)

Emp\_Car(EID, CID\_FK)

## 2 Mandatory

One-to-One

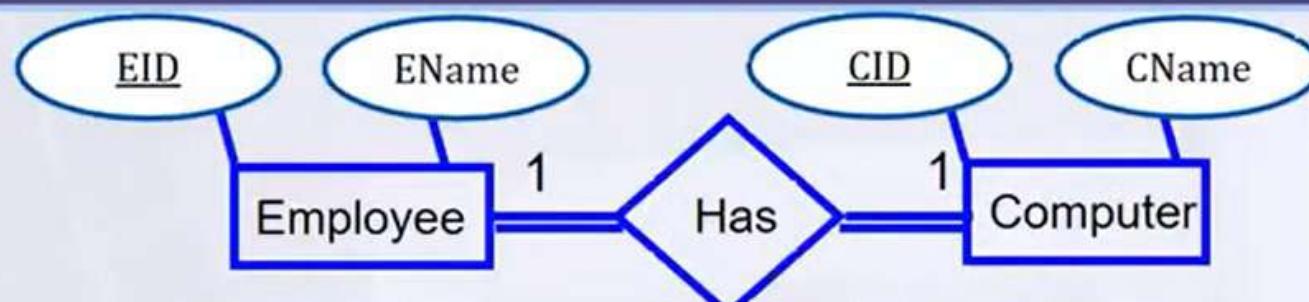
2 Mandatory



1 table

tbl\_xy (PK,.....,.....)

PK = PK<sub>x</sub> or PK<sub>y</sub>



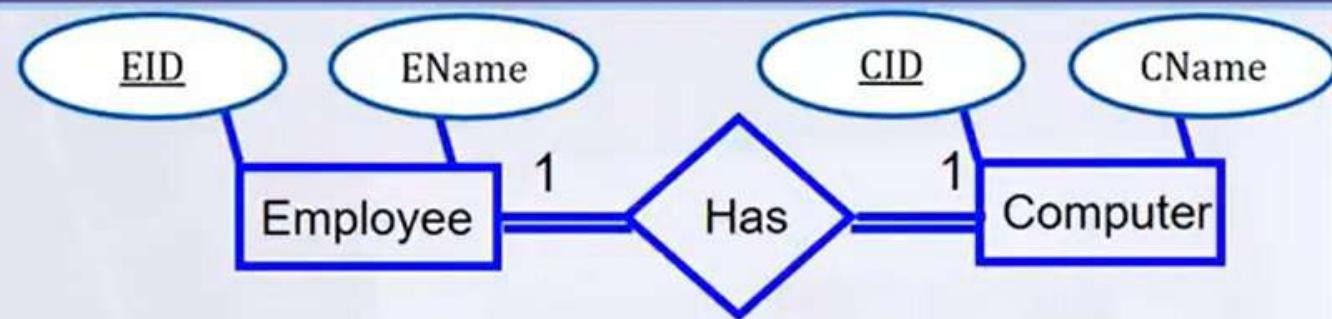
Emp(EID, Ename, Cname, **CID**)

# 2 Mandatory

**One-to-One**  
**2 Mandatory**



**1 table**  
tbl\_xy (PK,.....,.....)  
PK = PK<sub>x</sub> or PK<sub>y</sub>



Emp(EID, Ename, Cname, **CID**)

# 2 Optional

One-to-One

2 Optional



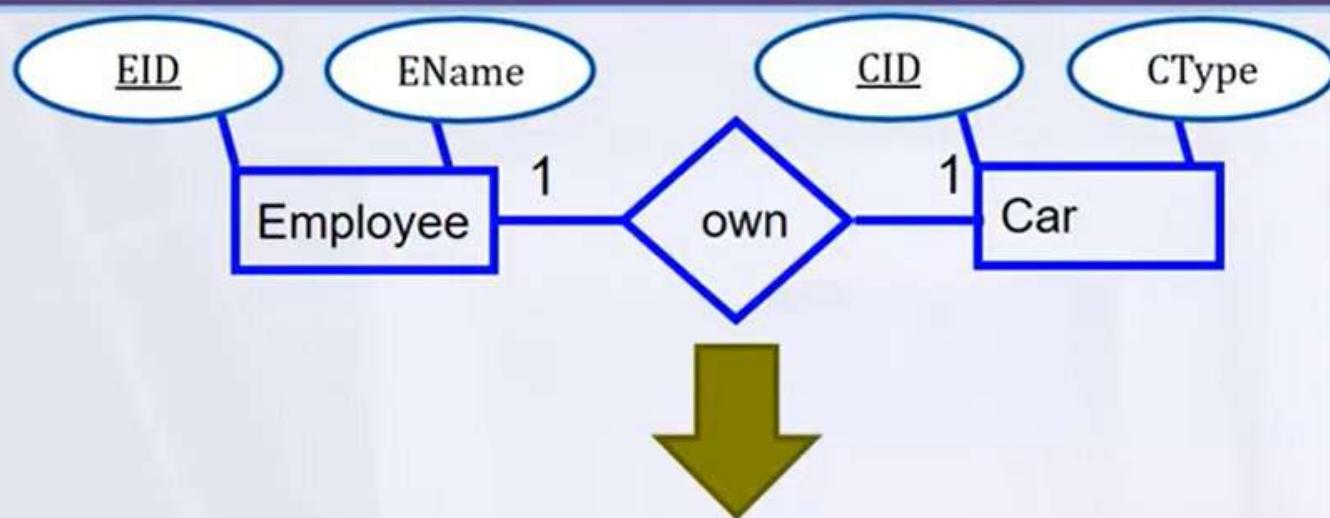
3 tables

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....)

tbl\_xy (PKxy,.....,...,FKxy,....)

PKxy = PKx or PKy



Employee(EID, Ename)

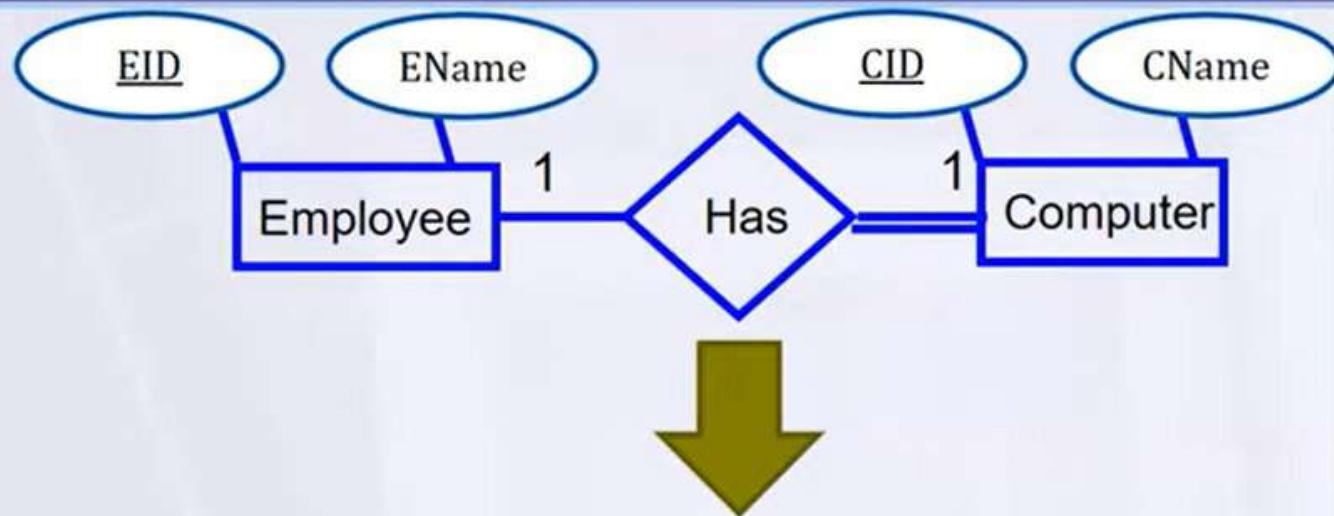
Car(CID, CType)

Emp\_Car(EID, CID\_FK)

# Optional-Mandatory

**One-to-One**

X optional – Y mandatory



**2 tables**

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....,PKx....)

Employee(EID, Ename)

Computer(CID, Cname, EID\_FK)

## Step 4: Mapping of Binary 1:N Relationship Types.

- Add FK to N-side table if N-Side mandatory
- Add any simple attributes of relationship as column to N-side table.

# Many is Optional

**One-to-Many**

X whatever- Y Optional



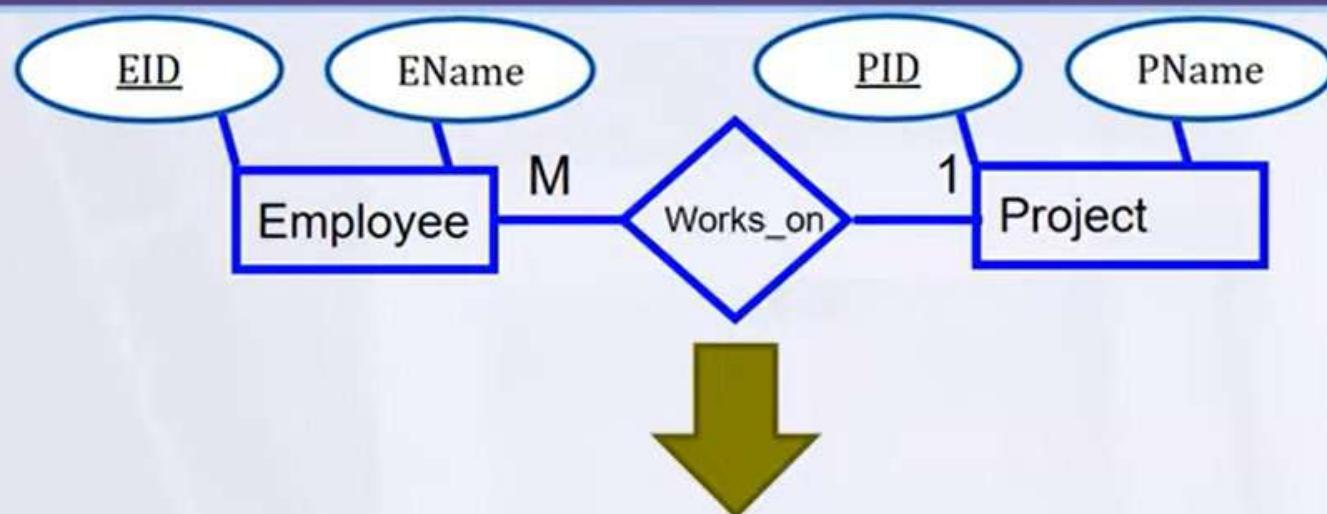
**3 tables**

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....)

tbl\_xy (PKxy,.....,.....)

$\text{PK}_{\text{xy}} = \text{PK}_y$



Project(PID, Pname)

Employee(EID, Ename)

Proj\_Emp(EID,PID\_FK)

# Many is Mandatory

**One-to-Many**

X whatever - Y mandatory

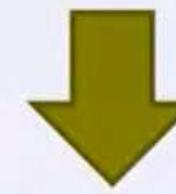
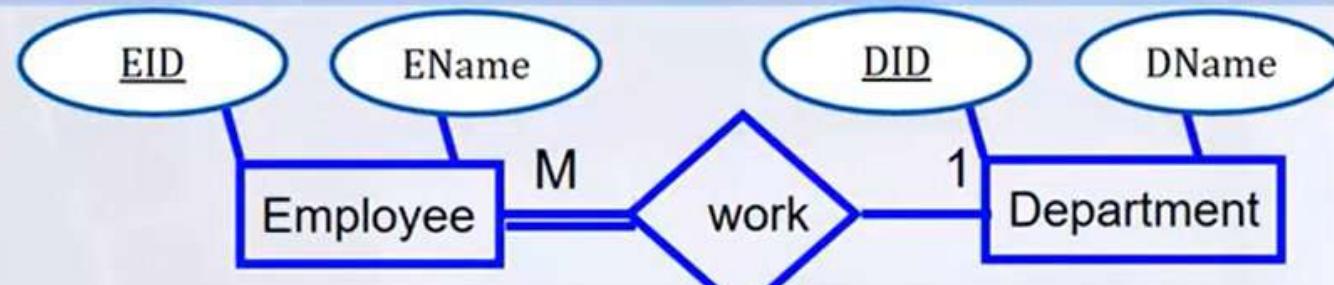


**2 tables**

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....,FKy....)

FKy= PKx



Department(DID, Dname)

Employee(EID, Ename,DID)

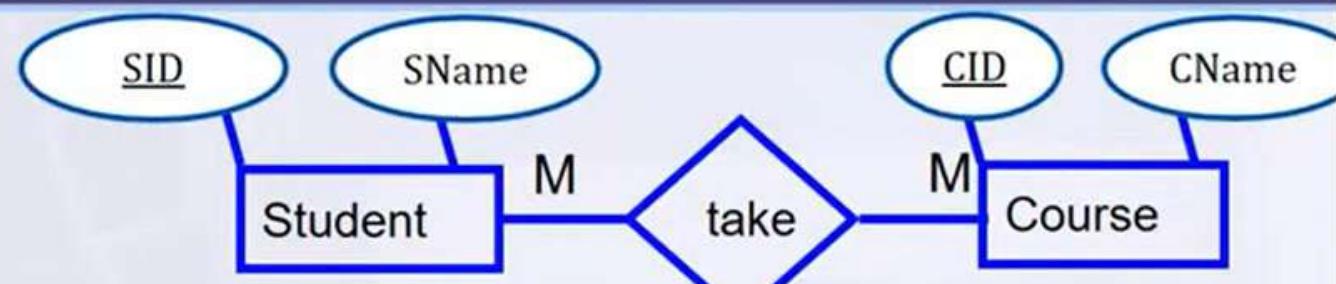
## Step 5: Mapping of Binary M:N Relationship Types.

- 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 .

# M:N

## Many-to-Many

X whatever - Y whatever



**3 tables**

tbl\_x (PK<sub>x</sub>,.....,.....)

tbl\_y (PK<sub>y</sub>,.....,.....)

tbl\_xy (PK<sub>x</sub>,PK<sub>y</sub>, .....,.....)

PK<sub>xy</sub>=PK<sub>x</sub>+PK<sub>y</sub>

Student(SID, Sname)

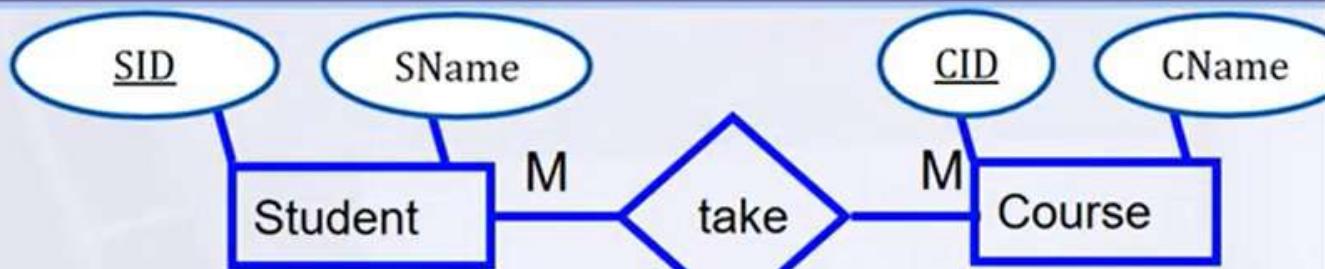
Course(CID, Cname)

Stud\_Course(SID, CID)

# M:N

## Many-to-Many

X whatever - Y whatever



3 tables

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....)

tbl\_xy (PKx,PKy,.....,.....)

PKxy = PKx + PKy

Student(SID, Sname)

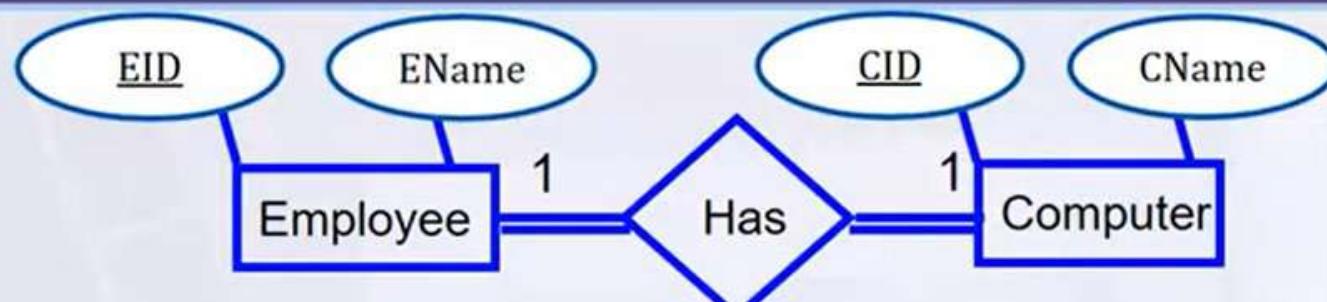
Course(CID, Cname)

Stud\_Course(SID, CID)

## 2 Mandatory

One-to-One

2 Mandatory



1 table

tbl\_xy (PK,.....,.....)

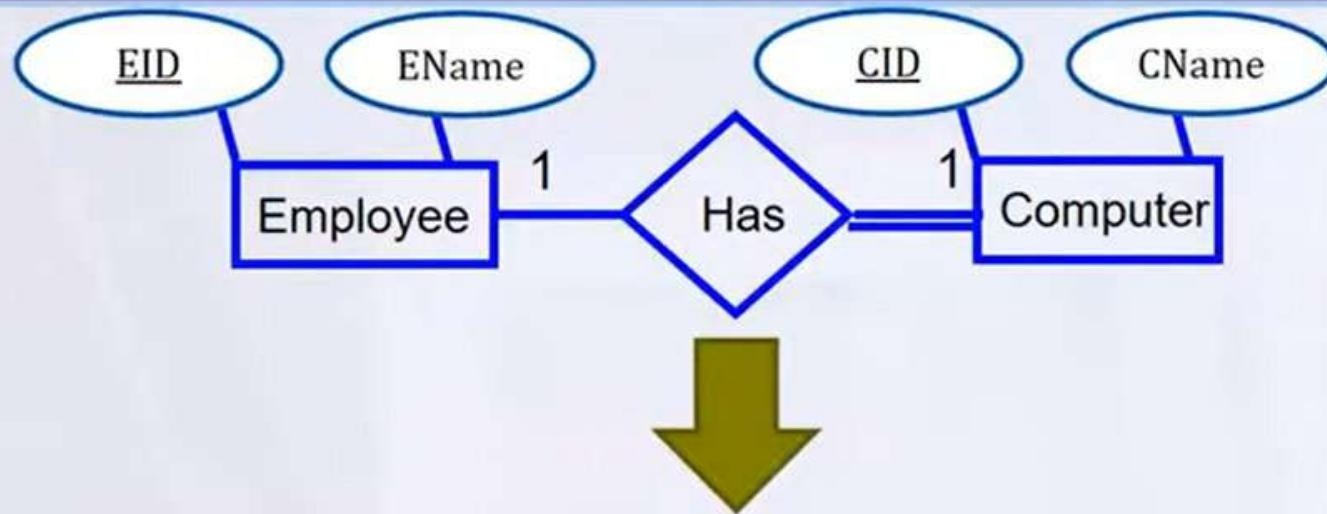
PK = PK<sub>x</sub> or PK<sub>y</sub>

Emp(EID, Ename, Cname, **CID**)

# Optional-Mandatory

**One-to-One**

X optional – Y mandatory



**2 tables**

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....,PKx....)

Employee(EID, Ename)

Computer(CID, Cname, **EID\_FK**)

# 2 Optional

One-to-One

2 Optional



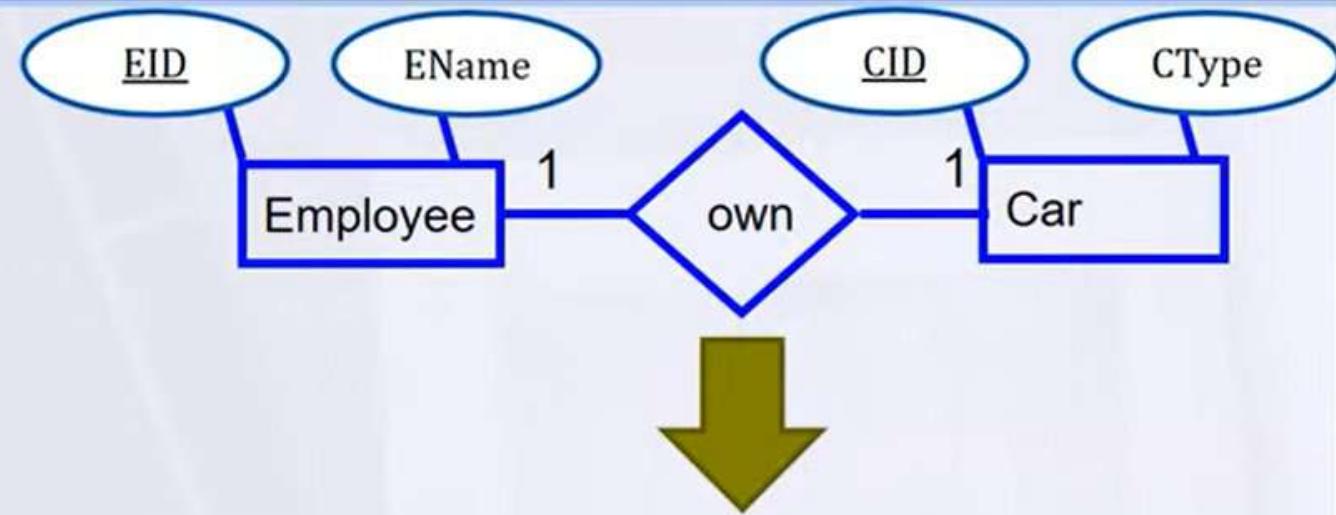
3 tables

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....)

tbl\_xy (PKxy,.....,...,FKxy,....)

PKxy = PKx or PKy



Employee(EID, Ename)

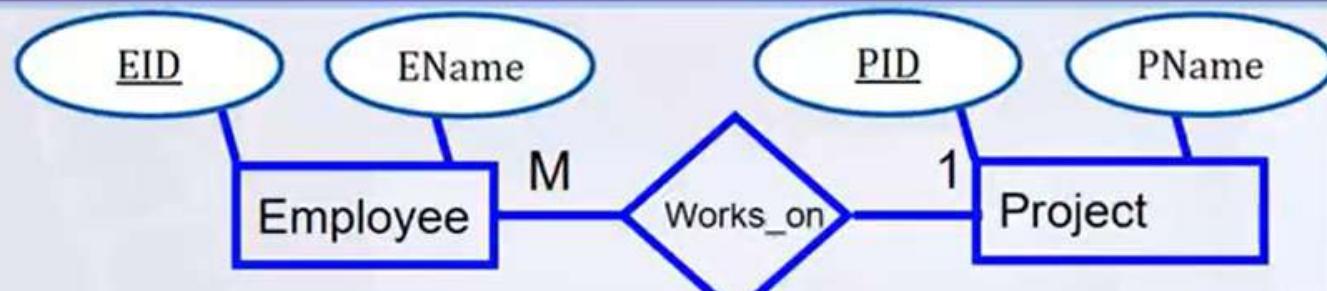
Car(CID, CType)

Emp\_Car(EID, CID\_FK)

# Many is Optional

**One-to-Many**

X whatever- Y Optional



**3 tables**

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....)

tbl\_xy (PKxy,.....,.....)

PKxy = PKy

Project(PID, Pname)

Employee(EID, Ename)

Proj\_Emp(EID,PID\_FK)

## Step 5: Mapping of Binary M:N 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 .

# M:N

## Many-to-Many

X whatever - Y whatever



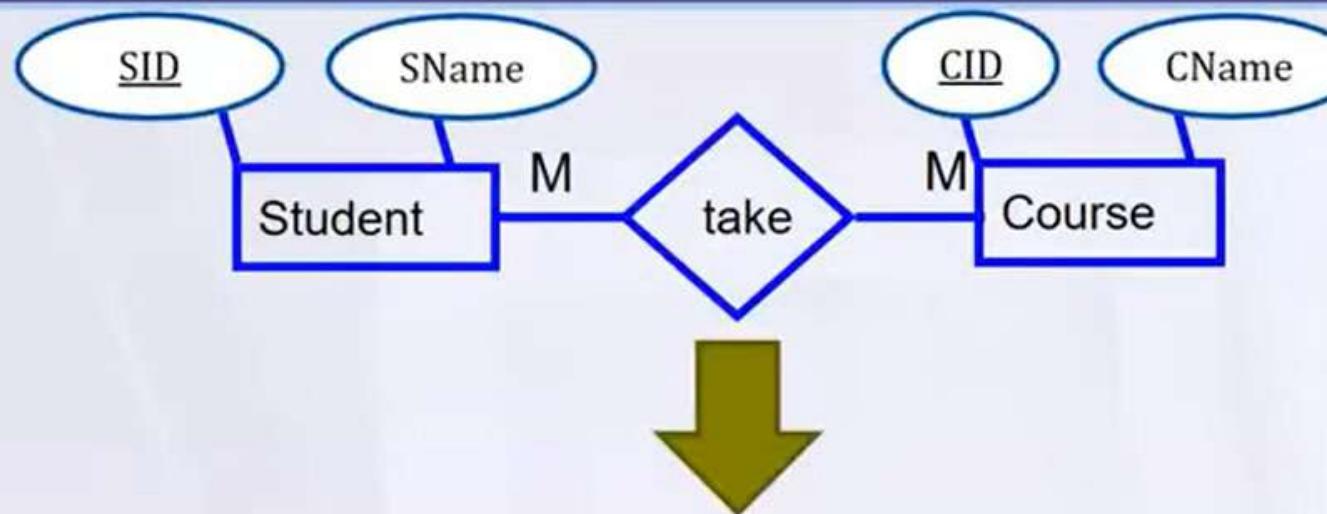
3 tables

tbl\_x (PKx,.....,.....)

tbl\_y (PKy,.....,.....)

tbl\_xy (PKx,PKy, .....,.....)

PKxy = PKx + PKy



Student(SID, Sname)

Course(CID, Cname)

Stud\_Course(SID, CID)

# M:N with attribute



The diagram provides a detailed view of the three entities as tables:

- RAW MATERIALS**:  
A table with columns: **Material\_ID**, **Standard\_Cost**, and **Unit\_of\_Measure**.
- QUOTE**:  
A table with columns: **Material\_ID**, **Vendor\_ID**, and **Unit\_Price**.
- VENDOR**:  
A table with columns: **Vendor\_ID**, **Vendor\_Name**, and **Vendor\_Address**.

Curved arrows from the ER diagram point to the **Material\_ID** column in the **RAW MATERIALS** table, the **Material\_ID** and **Vendor\_ID** columns in the **QUOTE** table, and the **Vendor\_ID** column in the **VENDOR** table, indicating that these columns are shared or related across the entities.

# M:N with attribute

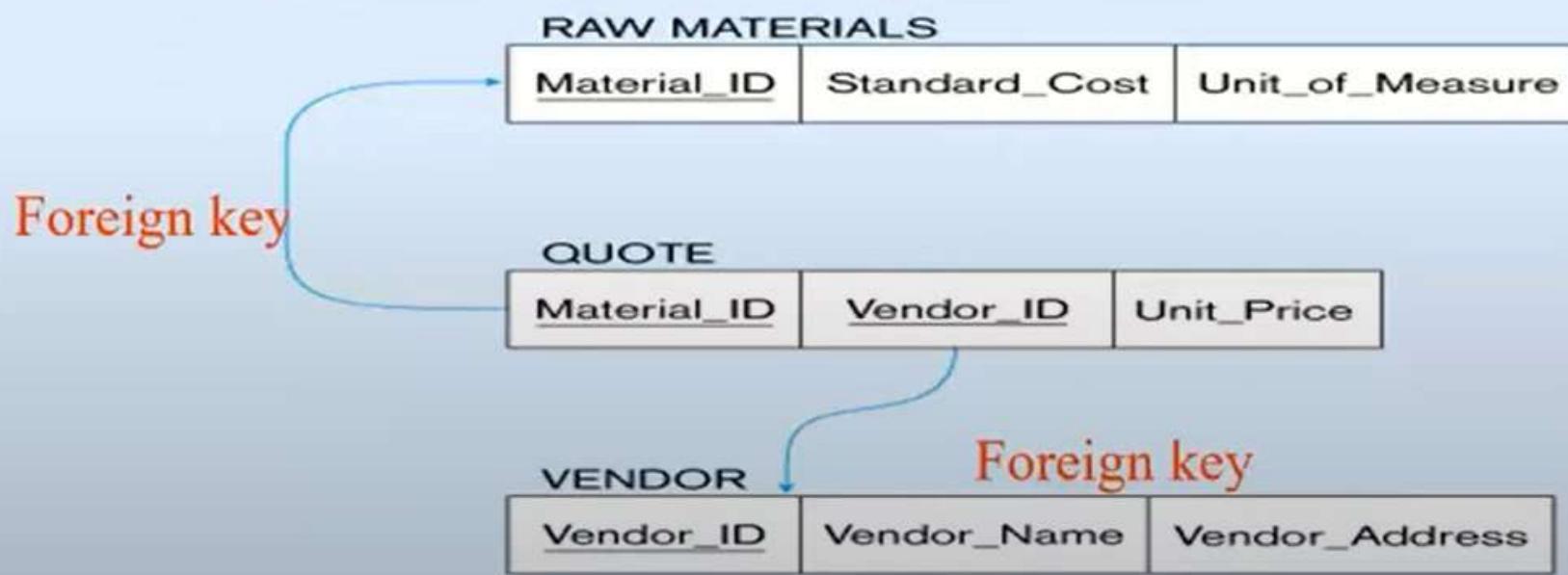


RAW MATERIALS		
Material_ID	Standard_Cost	Unit_of_Measure

QUOTE		
Material_ID	Vendor_ID	Unit_Price

VENDOR		
Vendor_ID	Vendor_Name	Vendor_Address

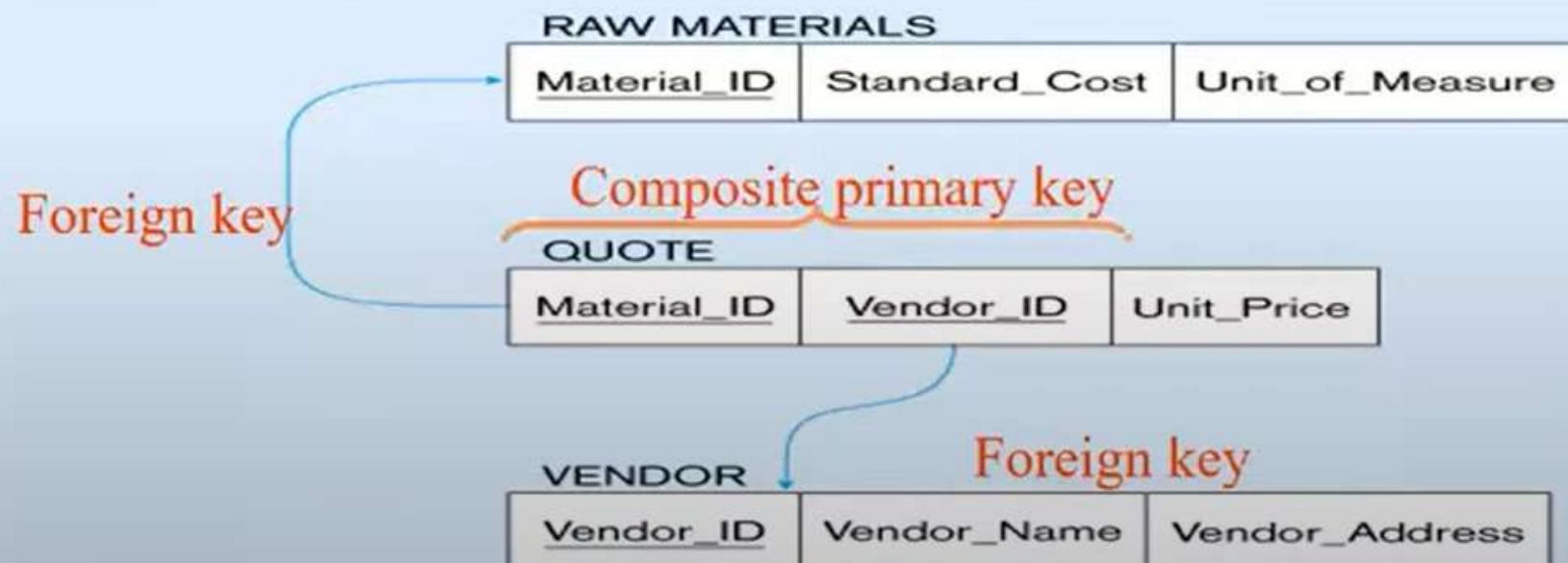
# M:N with attribute



# M:N with attribute



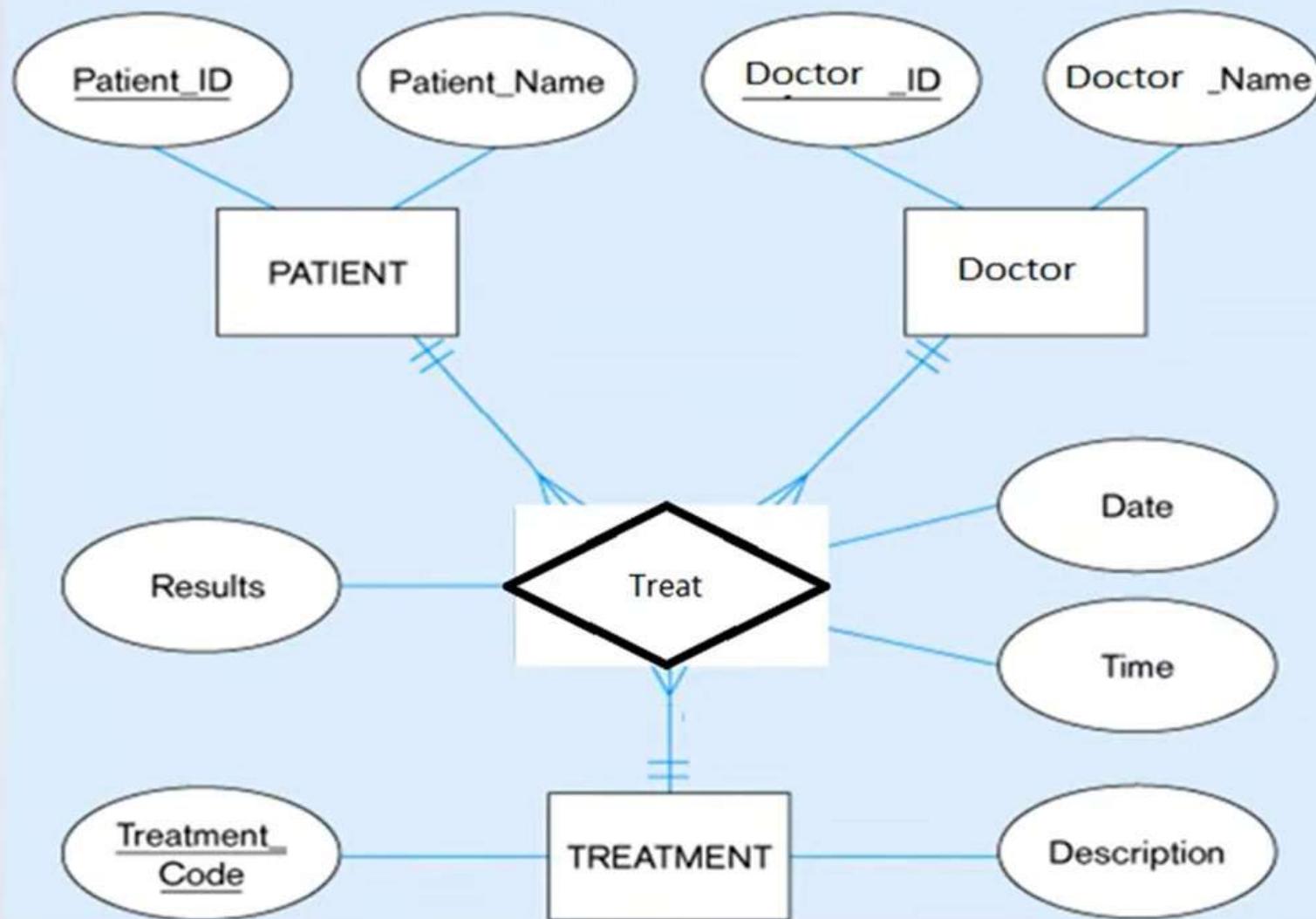
The *Supplies* relationship will need to become a separate relation



## Step 6: Mapping of N-ary Relationship Types.

- If  $n > 2$  then :
- Create a new third table
- Add FKs to the new table for all parent tables

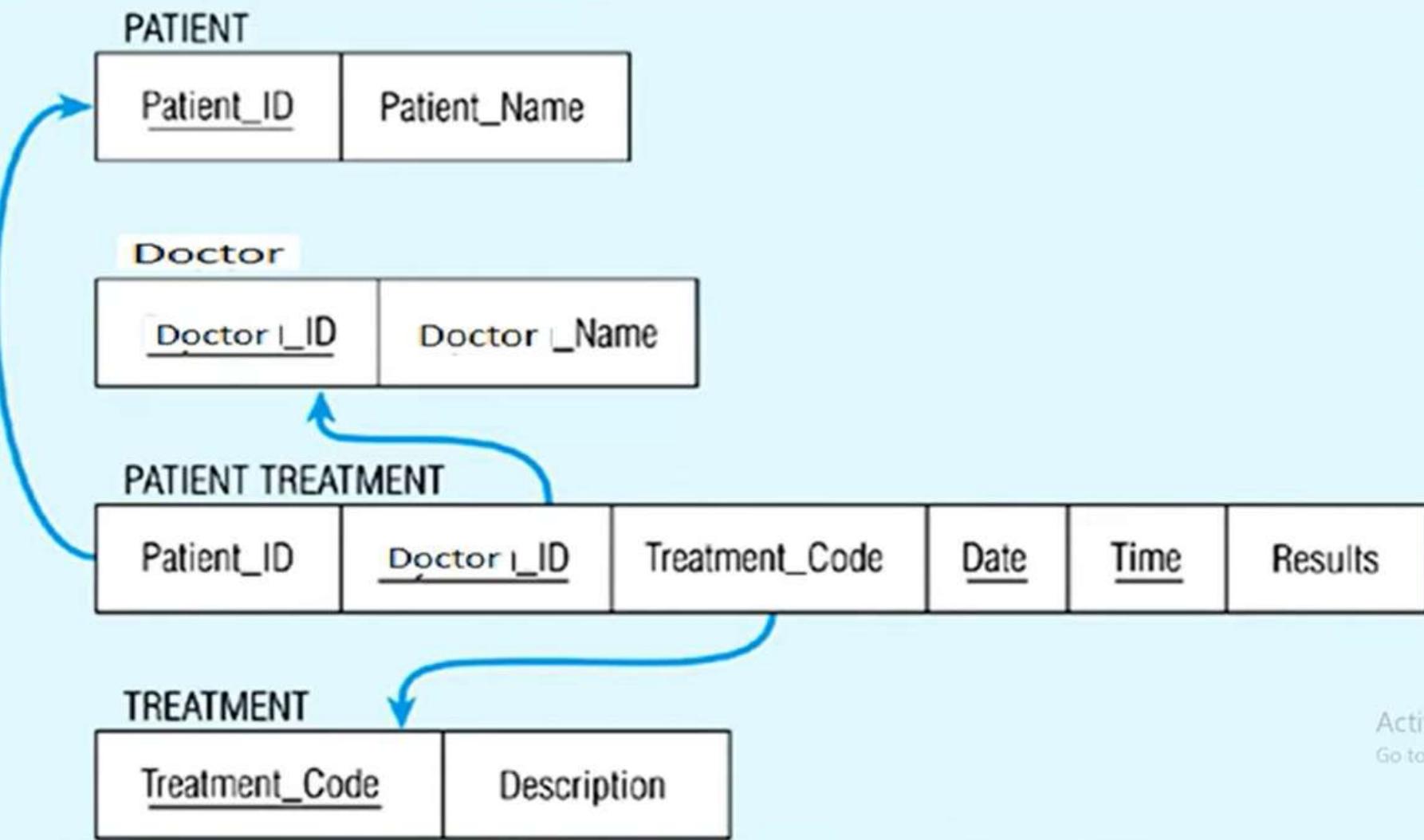
# Step 6: Mapping of N-ary Relationship Types.



Activate Wi  
Go to Settings

Activ  
Go to

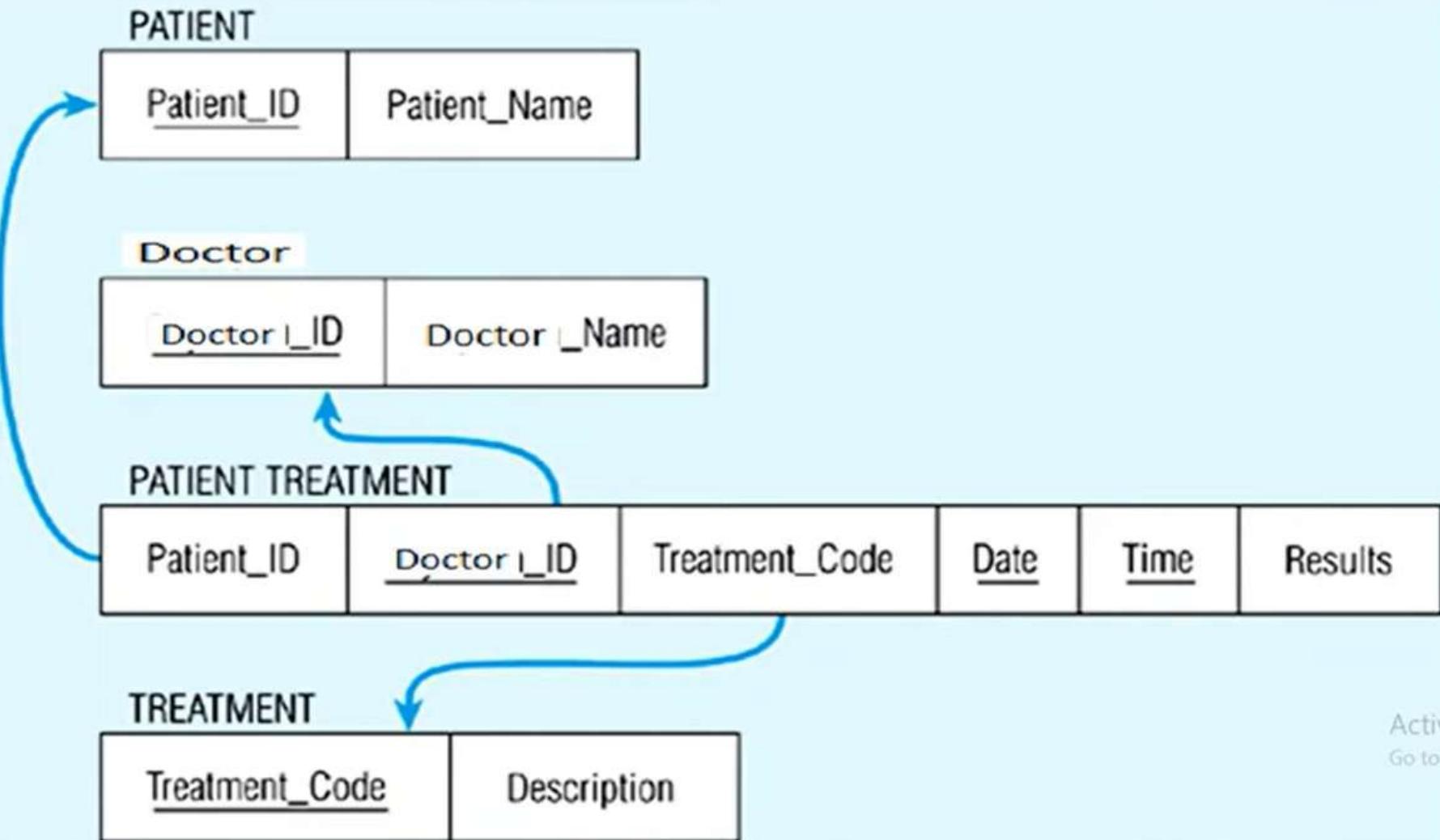
# Step 6: Mapping of N-ary Relationship Types.



Activate Wi  
Go to Settings t

Activ

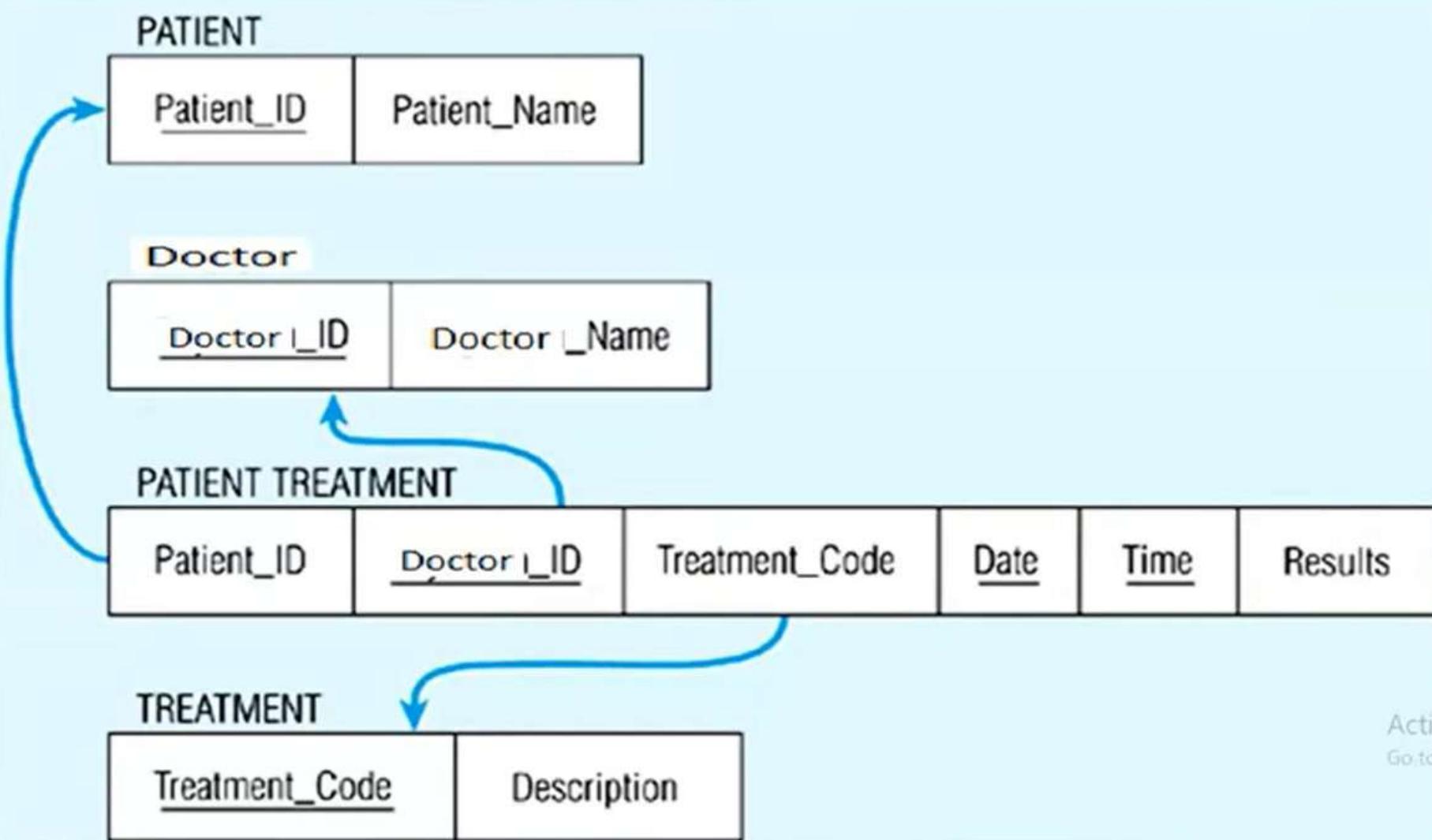
# Step 6: Mapping of N-ary Relationship Types.



Activate Wi  
Go to Settings &

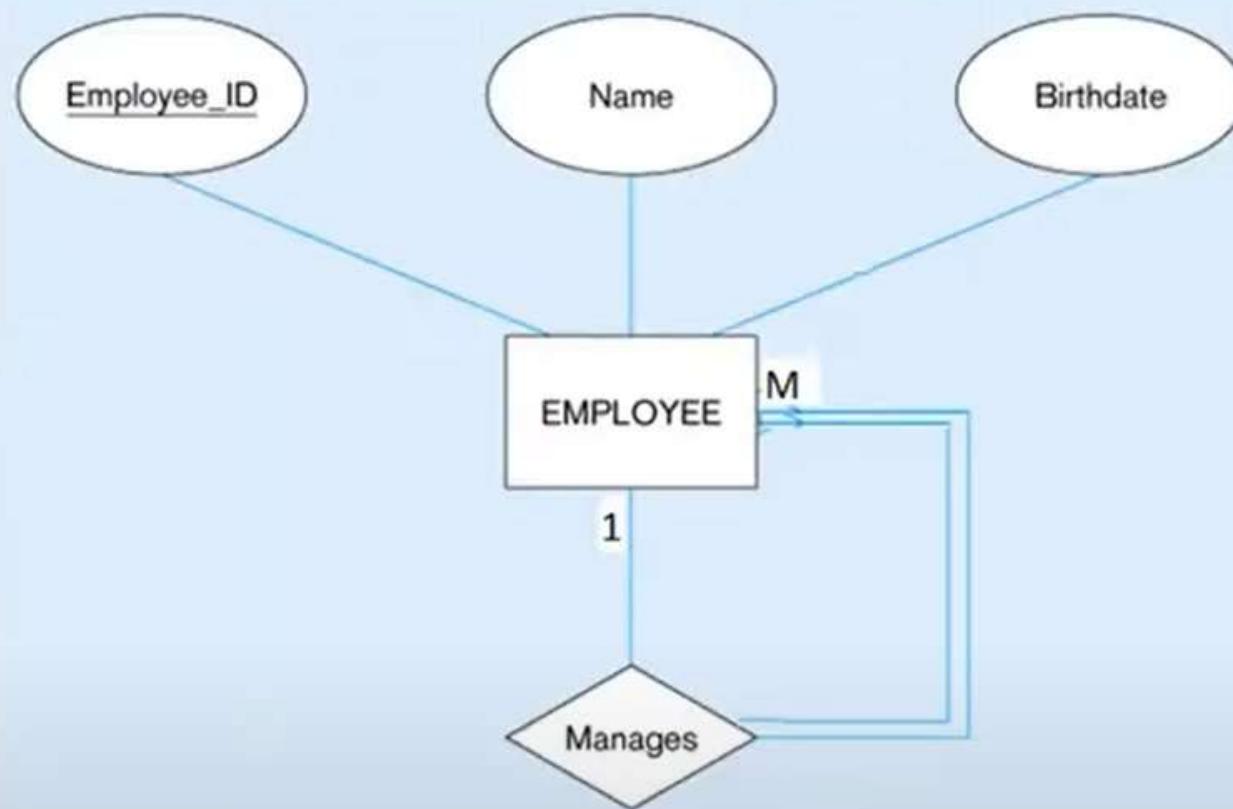
Activ  
Go to &

# Step 6: Mapping of N-ary Relationship Types.



# Step 7: Mapping Unary Relationship

(a) EMPLOYEE entity with  
Manages relationship



(b) EMPLOYEE  
relation with  
recursive foreign  
key

EMPLOYEE			
<u>Employee_ID</u>	Name	Birthdate	<u>Manager_ID</u>

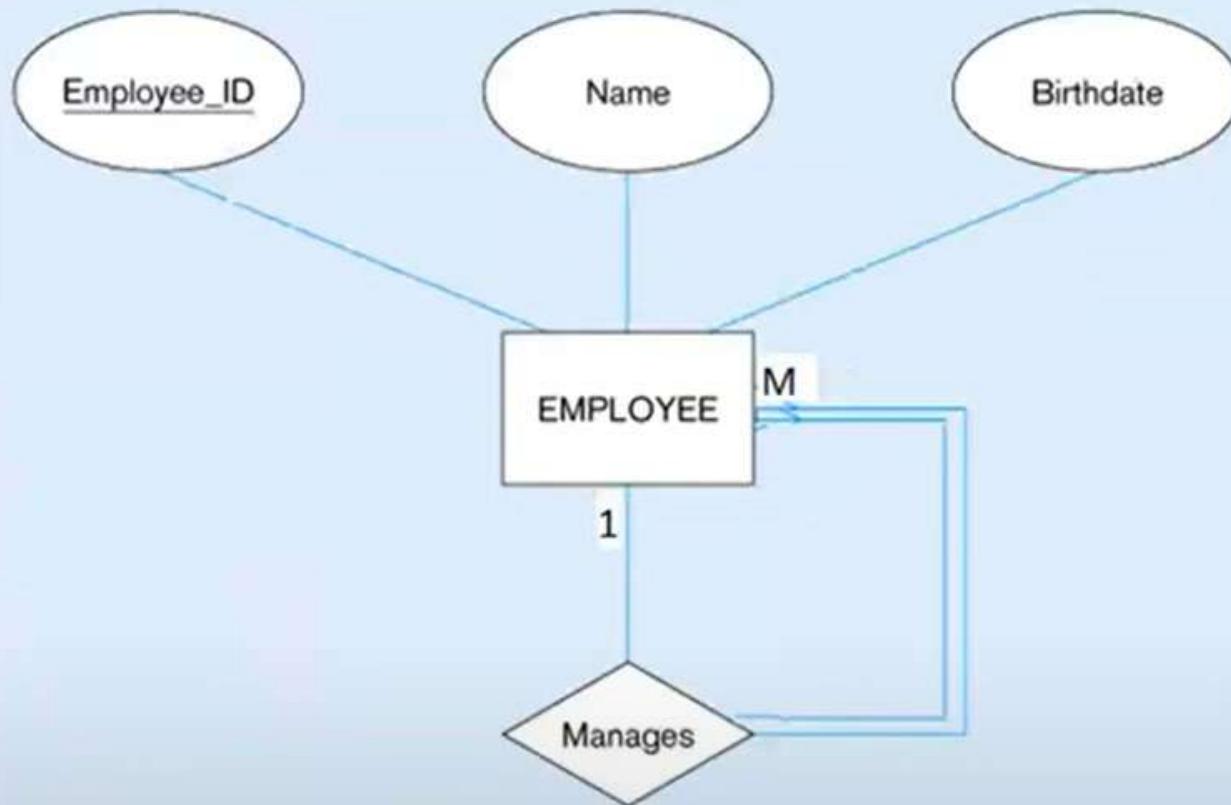


Emp(Eid,Ename,age ,Mid)

1,ahmed,22 ,	2
2,eman,23 ,	3
3,ali,24 ,	2
4,omar,	1

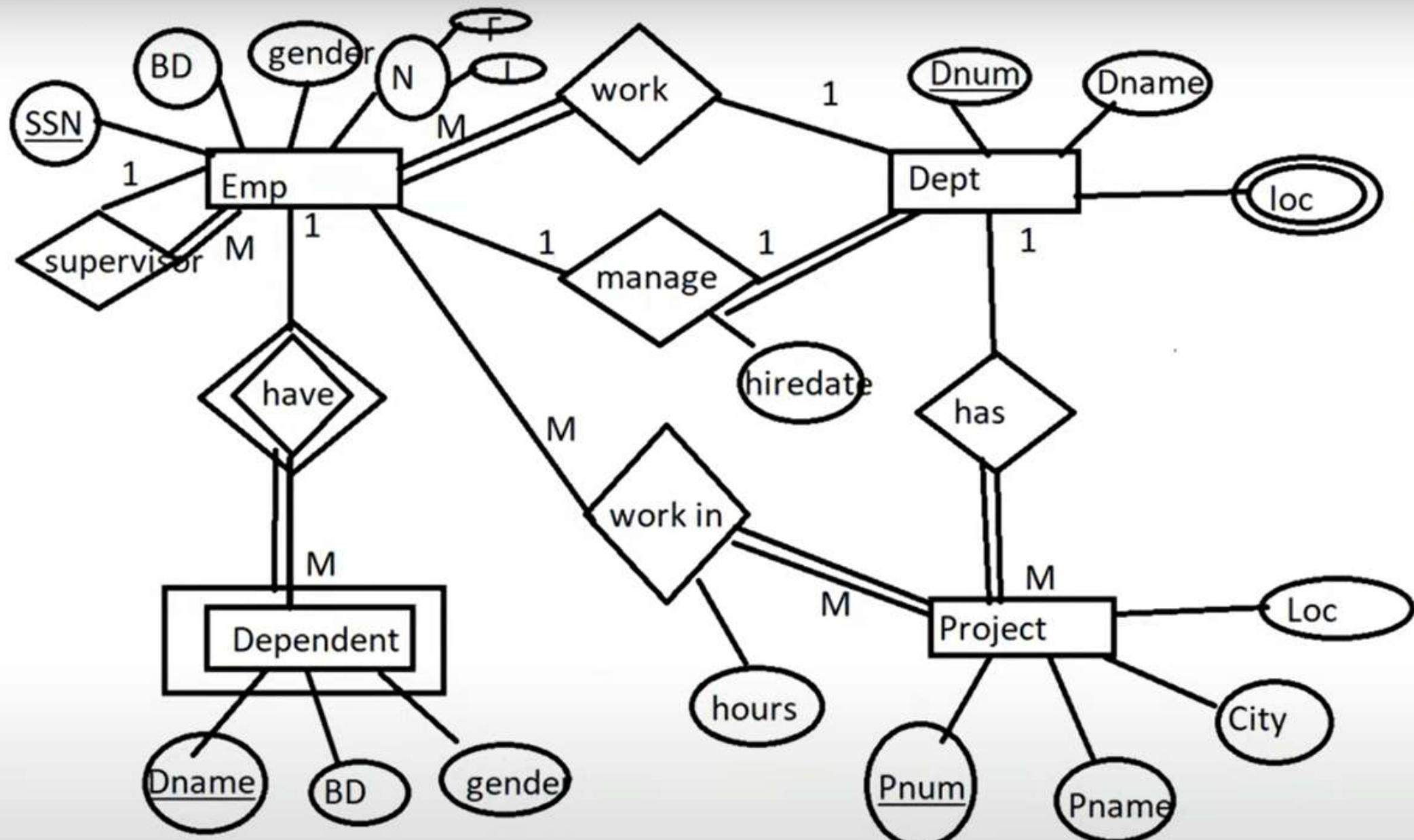
# Step 7: Mapping Unary Relationship

(a) EMPLOYEE entity with  
Manages relationship



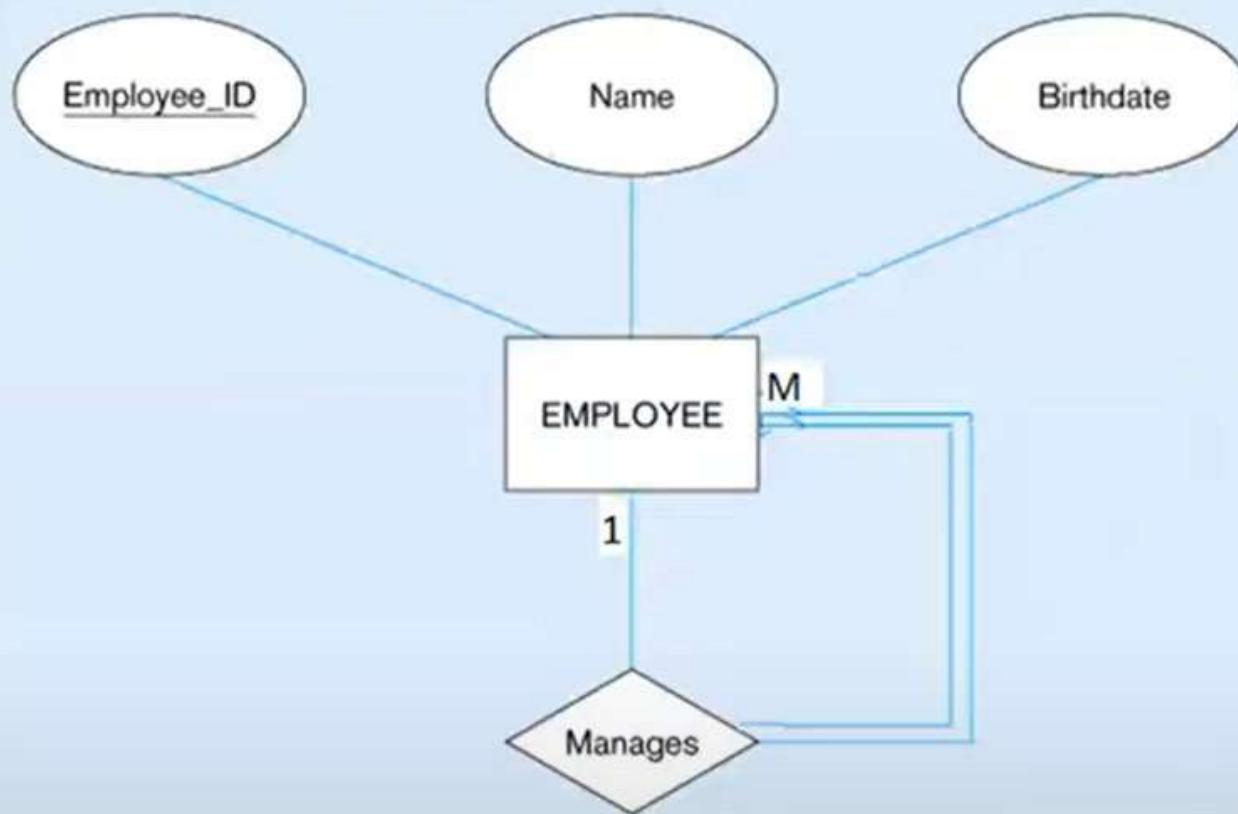
(b) EMPLOYEE  
relation with  
recursive foreign  
key

EMPLOYEE			
<u>Employee_ID</u>	Name	Birthdate	<u>Manager_ID</u>



# Step 7: Mapping Unary Relationship

(a) EMPLOYEE entity with  
Manages relationship

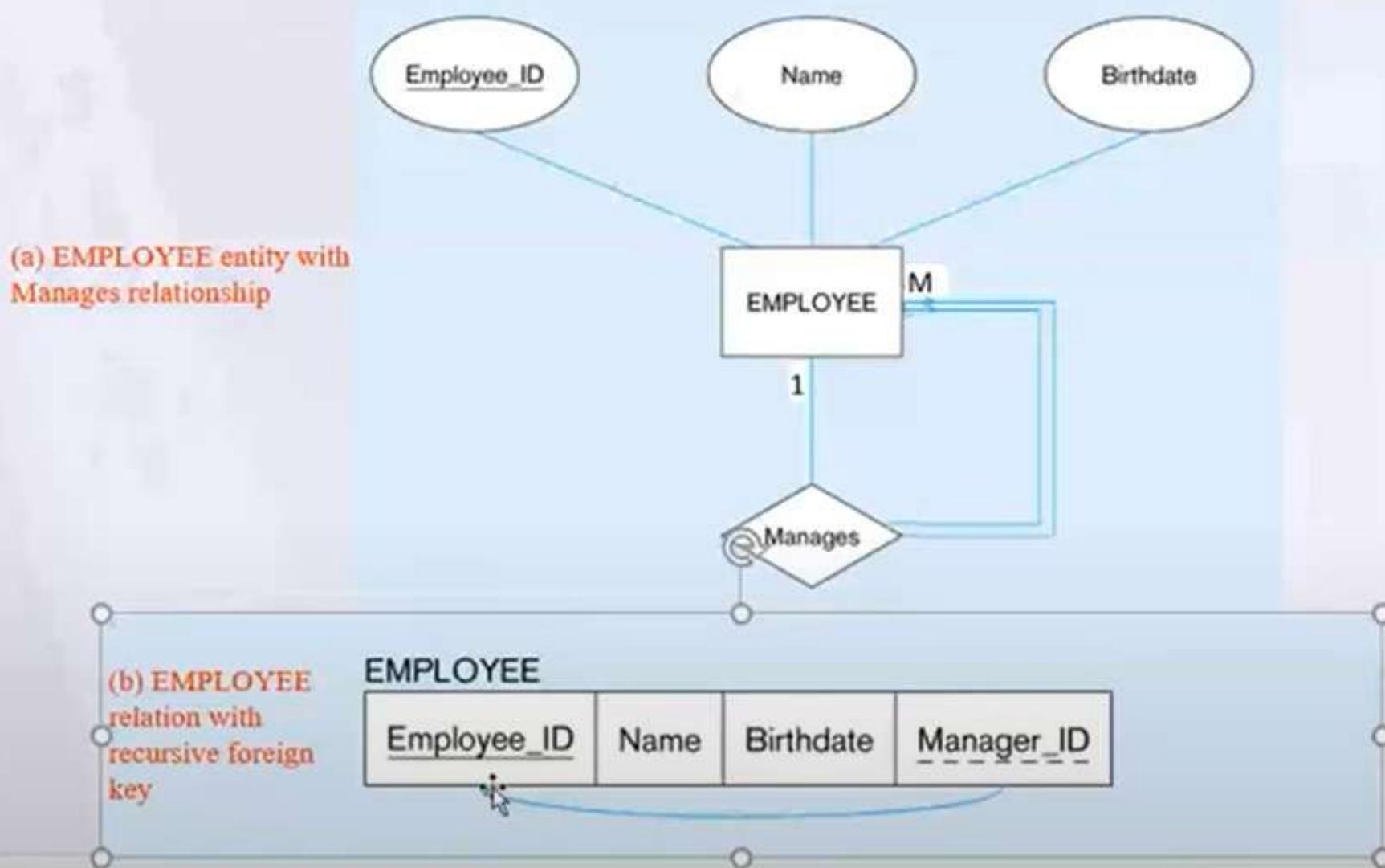


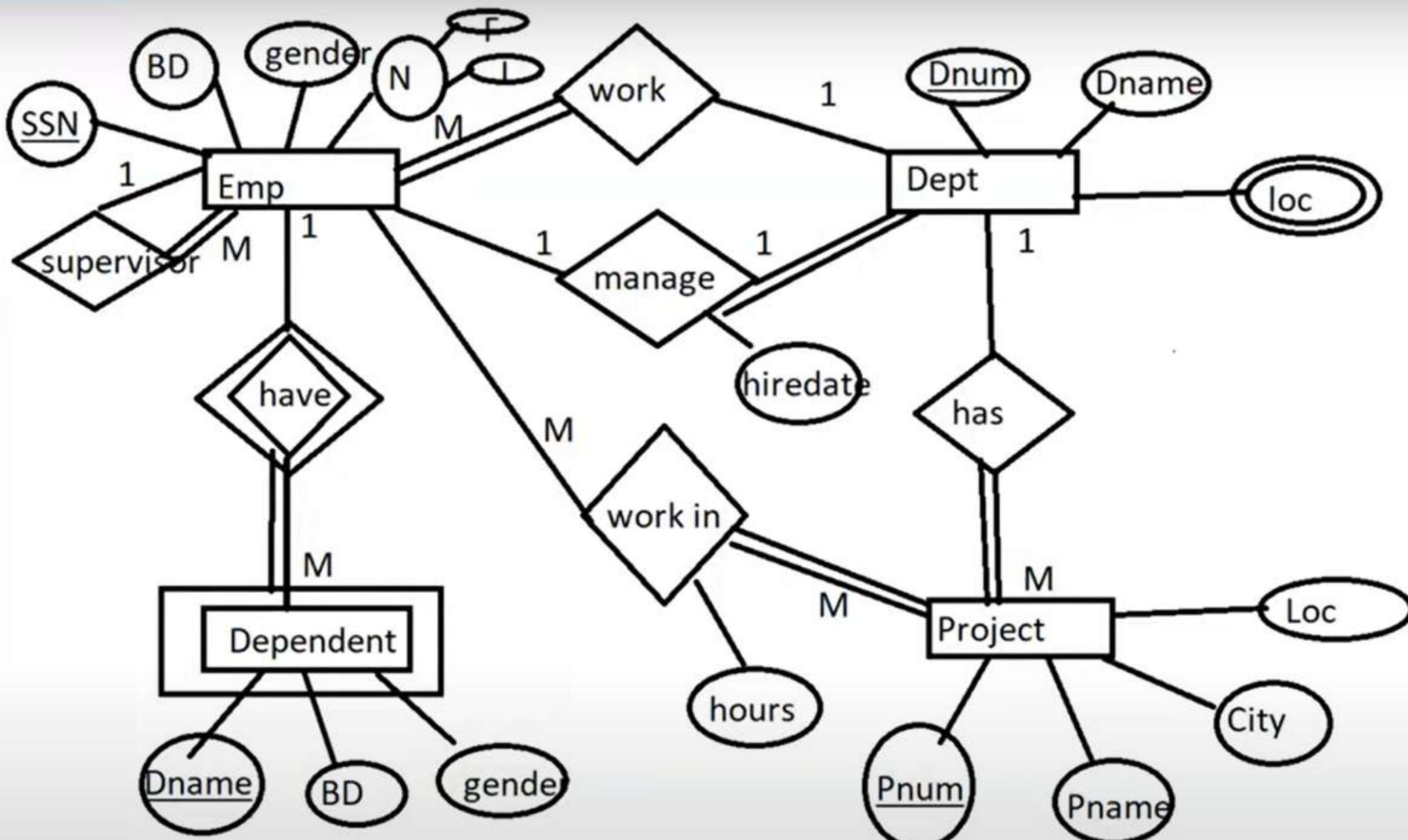
(b) EMPLOYEE  
relation with  
recursive foreign  
key

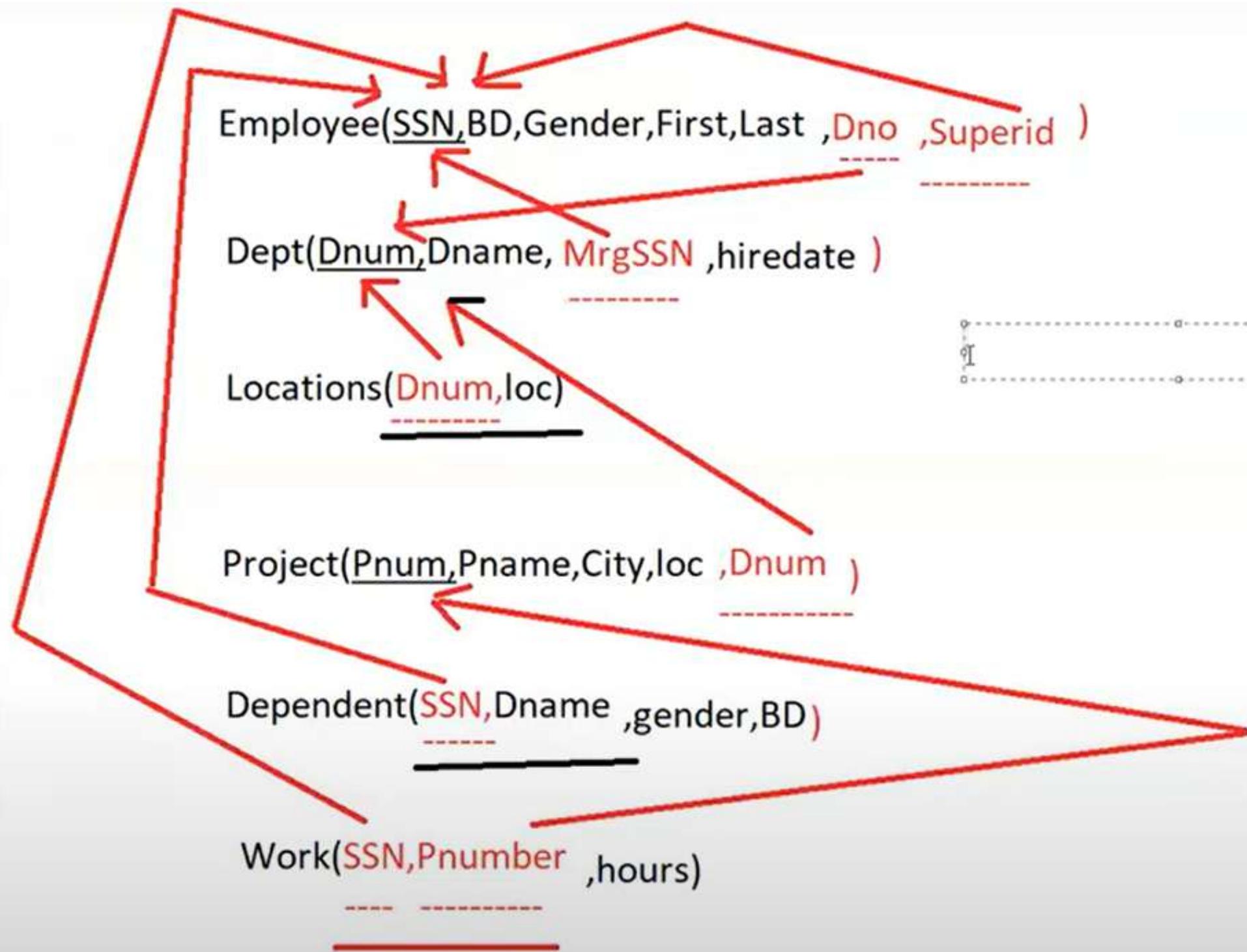
EMPLOYEE

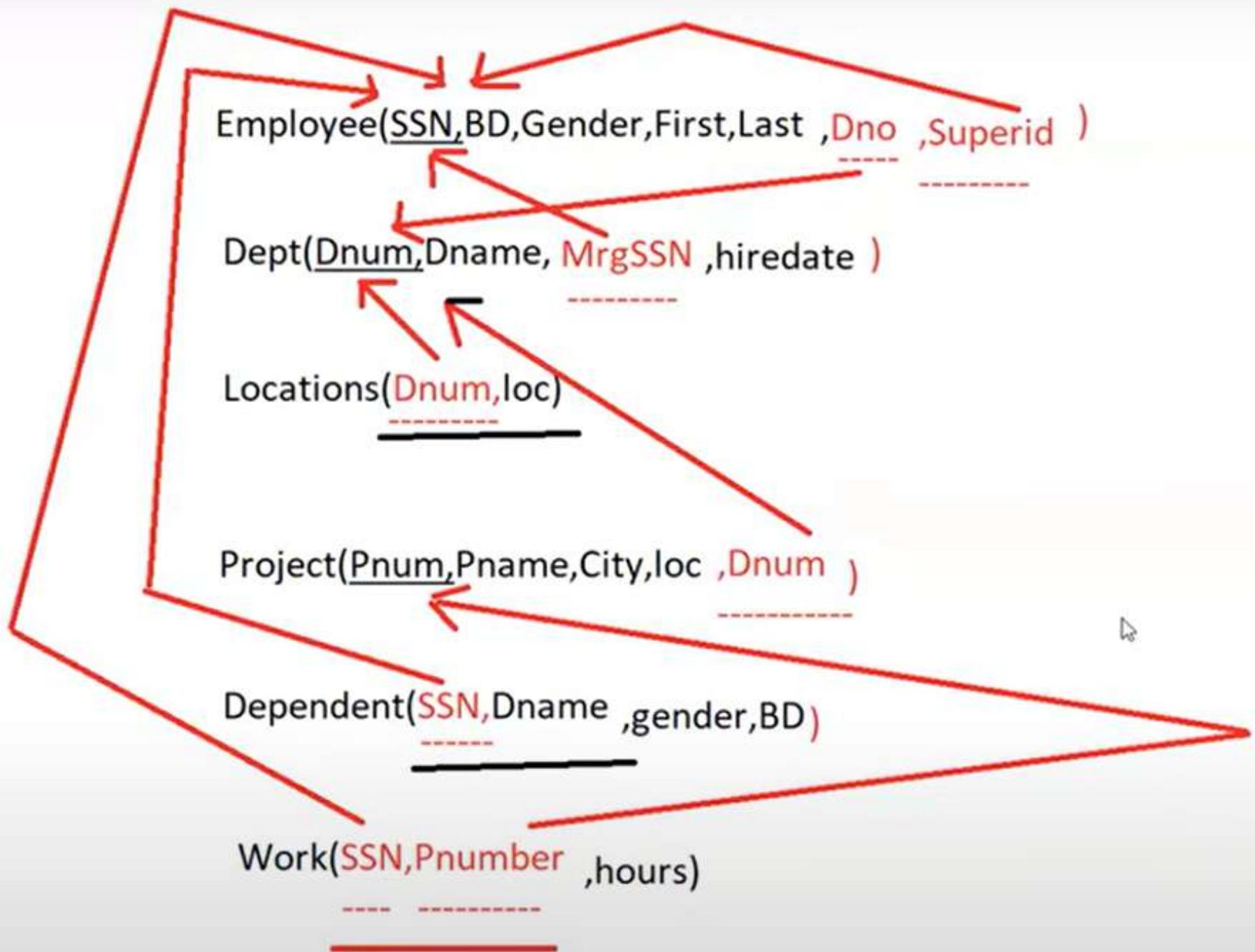
Employee_ID	Name	Birthdate	Manager_ID
-------------	------	-----------	------------

## Step 7: Mapping Unary Relationship









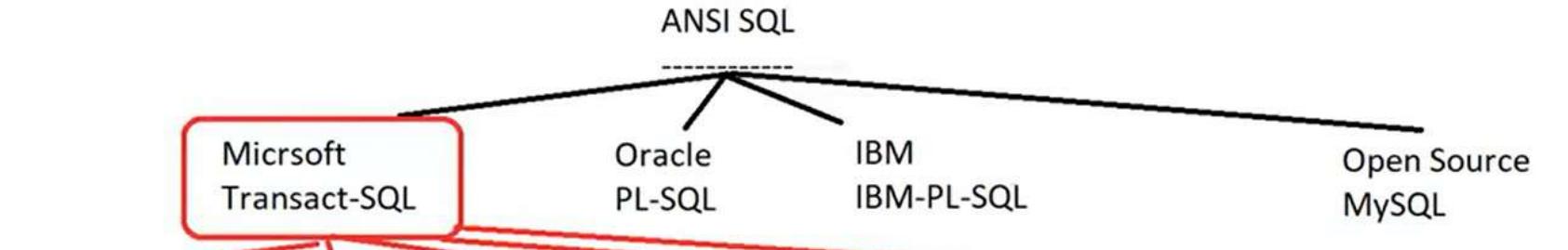


...  
...



# Microsoft SQL Server Management Studio

App

ANSI SQL				
				
DDL	DML	DCL	DQL	TCL
Data Definition Lang Metadata & Structure	Data Manipulation lang Data	Data Control Lang Security & Permissions	Data Query lang Display	Transational Control Lang Execution
Create table Create view Create Function Alter Drop Select into	Insert Update Delete Merge	Grant Deny Revoke	Select + Agg Fun Grouping union joins Subqueries	begin transaction commit rollback

Activate Windows

ANSI SQL				
Microsoft Transact-SQL	Oracle PL-SQL	IBM IBM-PL-SQL	Open Source MySQL	
DDL	DML	DCL	DQL	TCL
Data Definition Lang Metadata & Structure	Data Manipulation lang Data	Data Control Lang Security& Permissions	Data Query lang Display	Transational Control Lang Execution
Create table Create view Create Function Alter Drop Select into	Insert Update Delete Merge	Grant Deny Revoke	Select + Agg Fun Grouping union joins Subqueries	begin transaction commit rollback

Activate Windows

Get the full version of Microsoft Windows

 SD.mdf	02/11/2020 14:43	SQL Server Database	8,192 KB
 SD_Log.ldf	02/11/2020 14:43	SQL Server Database	8,192 KB

Object Explorer

Connect ▾

Databases

- System Databases
- Database Snapshots
- AdventureWorks2012
- AdventureWorksDW2012
- Company\_SD
- ITI
- ReportServer
- ReportServerTempDB
- sales
- SD

Tasks

- New Database...
- New Query
- Script Database as
- Tasks
- Policies
- Facets
- Start PowerShell
- Reports
- Rename
- Delete
- Refresh
- Properties

Detach...

Take Offline

Bring Online

Stretch ▾

Encrypt Columns...

Shrink ▾

Back Up... **▶**

Restore ▾

Mirror...

Launch Database Mirroring Monitor...

Ship Transaction Logs...

Generate Scripts...

Generate In-Memory OLTP Migration Checklists

Extract Data-tier Application...

Deploy Database to Microsoft Azure SQL Database...

Deploy Database to a Microsoft Azure VM...

Export Data-tier Application...

Register as Data-tier Application...

Upgrade Data-tier Application...

Delete Data-tier Application...

Import Data...

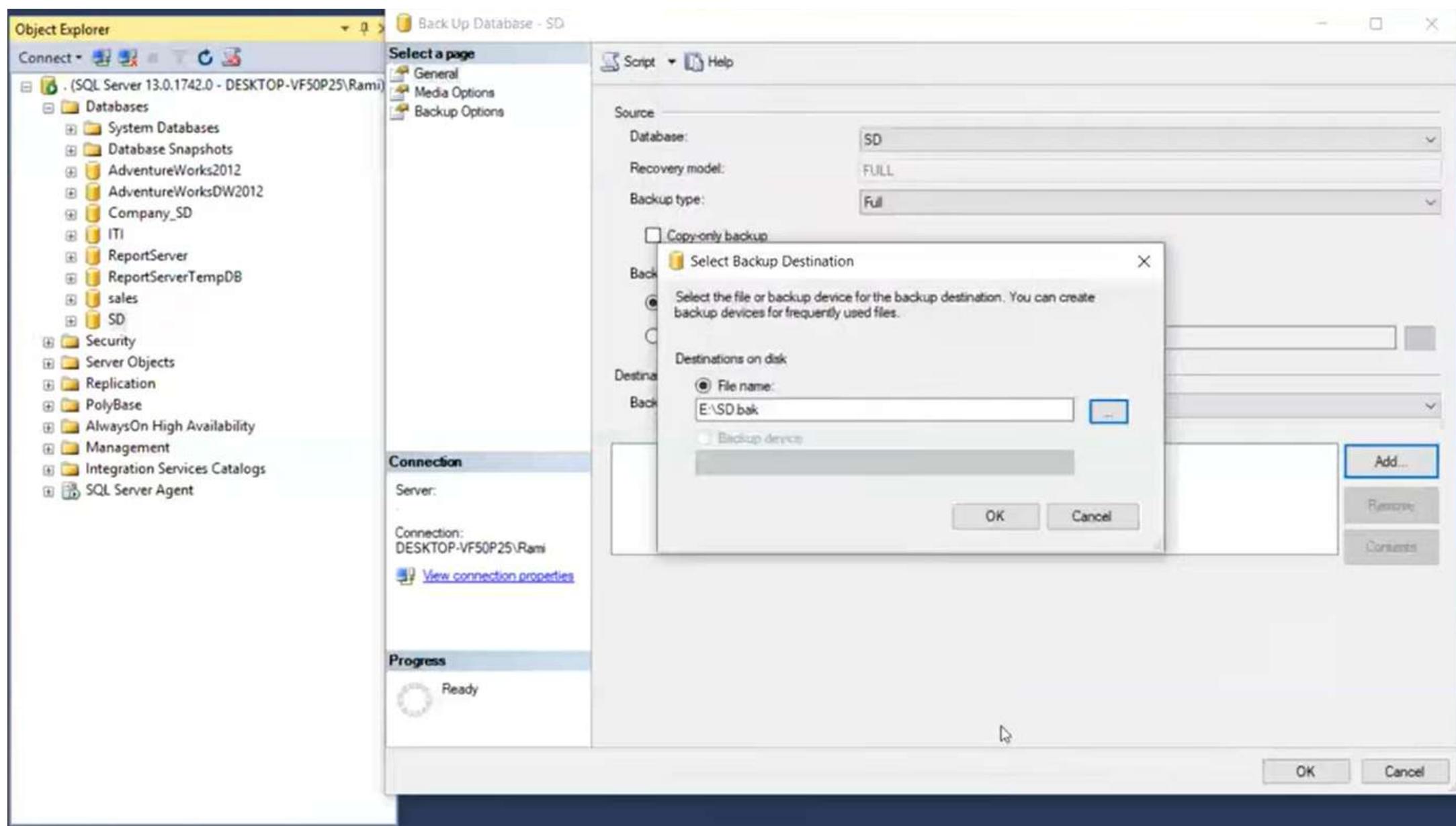
Export Data...

Copy Database...

Manage Database Encryption...

Output

Ready



Object Explorer

Connect ▾

. (SQL Server 13.0.1742.0 - DESKTOP-VF50P25\Ram)

- Databases
  - System Databases
  - Database Snapshots
  - AdventureWorks2012
  - AdventureWorksDW2012
  - Company\_SD
  - ITI
  - ReportServer
  - ReportServerTempDB
  - sales
  - SD
- Security
- Server Objects
- Replication
- PolyBase
- AlwaysOn High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent

Back Up Database - SD

Select a page

- General
- Media Options
- Backup Options

Script Help

Source

Database: SD

Recovery model: FULL

Backup type: Full

Copy-only backup

Backup component:

Database

Microsoft SQL Server Management Studio

The backup of database 'SD' completed successfully.

OK

Connection

Server: DESKTOP-VF50P25\Ram

[View connection properties](#)

Progress

Executing (100%)

Stop action now

OK Cancel

Add... Remove Connect

DESKTOP-VF5OP25...SD - dbo.Employee

Column Name	Data Type	Allow Nulls
SSN	int	<input type="checkbox"/>
FirstN	varchar(50)	<input type="checkbox"/>
LastN	varchar(20)	<input checked="" type="checkbox"/>
Gender	varchar(1)	<input checked="" type="checkbox"/>
BD	date	<input checked="" type="checkbox"/>
Dnum	int	<input checked="" type="checkbox"/>
Superid	int	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Object Explorer

Connect ▾

- .(SQL Server 13.0.1742.0 - DESKTOP-VF50P25)\Ram
- Databases
  - System Databases
  - Database Snapshots
  - AdventureWorks2012
  - AdventureWorksDW2012
  - Company\_SD
  - ITI
  - ReportServer
  - ReportServerTempDB
  - sales
  - SD
    - Database Diagrams
    - Tables
      - System Tables
      - FileTables
      - External Tables
      - dbo.Employee
        - Columns
        - Keys
        - Constraints
        - Triggers
        - Indexes
        - Statistics
      - Views
      - External Resources
      - Synonyms
      - Programmability
      - Service Broker
      - Storage
      - Security
    - Security
    - Server Objects
    - Replication

## DESKTOP-VF50P25.SD - dbo.Employee

Column Name	Data Type	Allow Nulls
SSN	int	<input type="checkbox"/>
FirstN	varchar(50)	<input type="checkbox"/>
LastN	varchar(20)	<input checked="" type="checkbox"/>
Gender	varchar(1)	<input checked="" type="checkbox"/>
BD	date	<input checked="" type="checkbox"/>
Dnum	int	<input checked="" type="checkbox"/>
Superid	int	<input checked="" type="checkbox"/>
age		<input type="checkbox"/>

## Column Properties

Object Explorer

Connect -

DESKTOP-VF50P25\SD - dbo.Employee\*

Column Name Data Type Allow Nulls

Column Name	Data Type	Allow Nulls
SSN	int	<input type="checkbox"/>
FirstN	varchar(50)	<input type="checkbox"/>
LastN	varchar(20)	<input checked="" type="checkbox"/>
Gender	varchar(1)	<input checked="" type="checkbox"/>
BD	date	<input checked="" type="checkbox"/>
Dnum	int	<input checked="" type="checkbox"/>
Superid	int	<input checked="" type="checkbox"/>
age	int	<input checked="" type="checkbox"/>

Column Properties

(General) (Name) age  
Allow Nulls Yes  
Data Type int  
Default Value or Binding

Table Designer (General)

Activate Windows

Properties Template Browser

Object Explorer

Connect ▾

- .(SQL Server 13.0.1742.0 - DESKTOP-VF50P25\Ram)
- Databases
  - System Databases
  - Database Snapshots
  - AdventureWorks2012
  - AdventureWorksDW2012
  - Company\_SD
  - ITI
  - ReportServer
  - ReportServerTempDB
  - sales
- SD
  - Database Diagrams
  - Tables
    - System Tables
    - FileTables
    - External Tables
    - dbo.Employee
      - Columns
      - Keys
      - Constraints
      - Triggers
      - Indexes
      - Statistics
  - Views
  - External Resources
  - Synonyms
  - Programmability
  - Service Broker
  - Storage
  - Security
- Security
- Server Objects
- Replication

DESKTOP-VF50P25.SD - dbo.Table\_1\*

Column Name	Data Type	Allow Nulls
Pnum	int	<input type="checkbox"/>
Pname	varchar(50)	<input checked="" type="checkbox"/>
City	varchar(50)	<input checked="" type="checkbox"/>
Loc	varchar(50)	<input checked="" type="checkbox"/>
Dnum	int	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Choose Name

Enter a name for the table:

Project

OK Cancel

Column Properties

(General) (Name) Pnum  
Allow Nulls No  
Data Type int  
Default Value or Binding  
Table Designer (General)

Object Explorer

Connect ▾

- (SQL Server 13.0.1742.0 - DESKTOP-VF50P25(Rami))
  - Databases
    - System Databases
    - Database Snapshots
    - AdventureWorks2012
    - AdventureWorksDW2012
    - Company\_SD
    - ITI
    - ReportServer
    - ReportServerTempDB
    - sales
    - SD
      - Database Diagrams
    - Tables
      - System Tables
      - FileTables
      - External Tables
      - dbo.Employee
      - dbo.Project
    - Views
    - External Resources
    - Synonyms
    - Programmability
    - Service Broker
    - Storage
    - Security
  - Security
  - Server Objects
  - Replication
  - PolyBase
  - AlwaysOn High Availability
  - Management
  - Integration Services Catalogs
  - SQL Server Agent

DESKTOP-VF50P25.S...P25.SD - dbo.work

Column Name	Data Type	Allow Nulls
SSN	int	<input type="checkbox"/>
Pnumber	int	<input type="checkbox"/>
Hours	int	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Properties

(General)

- Allow Nulls
- Data Type
- Default Value or Binding

Table Designer

- Collation

(General)

## Object Explorer

Connect

- . (SQL Server 13.0.1742.0 - DESKTOP-VF50P25\Ram)
- Databases
  - ⊕ System Databases
  - ⊕ Database Snapshots
  - ⊕ AdventureWorks2012
  - ⊕ AdventureWorksDW2012
  - ⊕ Company\_SD
  - ⊕ ITI
  - ⊕ ReportServer
  - ⊕ ReportServerTempDB
  - ⊕ sales
  - ⊕ SD
    - Database Diagrams
    - Tables
      - ⊕ System Tables
      - ⊕ FileTables
      - ⊕ External Tables
        - ⊕ dbo.Employee
        - ⊕ dbo.Project
        - ⊕ dbo.work
    - ⊕ Views
    - ⊕ External Resources
    - ⊕ Synonyms
    - ⊕ Programmability
    - ⊕ Service Broker
    - ⊕ Storage
    - ⊕ Security
  - ⊕ Security
  - ⊕ Server Objects
  - ⊕ Replication
  - ⊕ PolyBase
  - ⊕ AlwaysOn High Availability
  - ⊕ Management
  - ⊕ Integration Services Catalogs
  - ⊕ SQL Server Agent

## DESKTOP-VF50P25.SD - Diagram\_0

## Add Table

## Tables

Employee

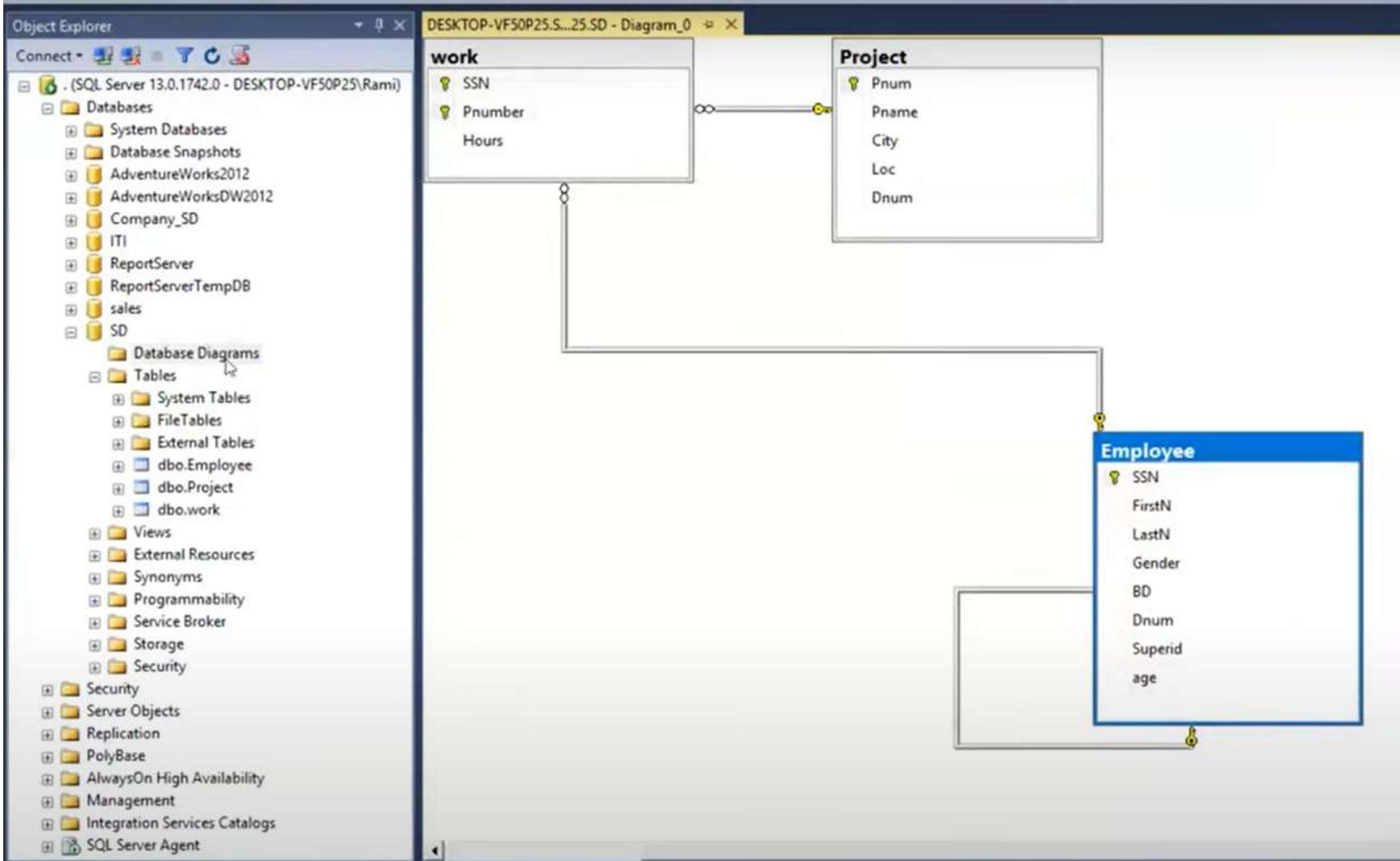
Project

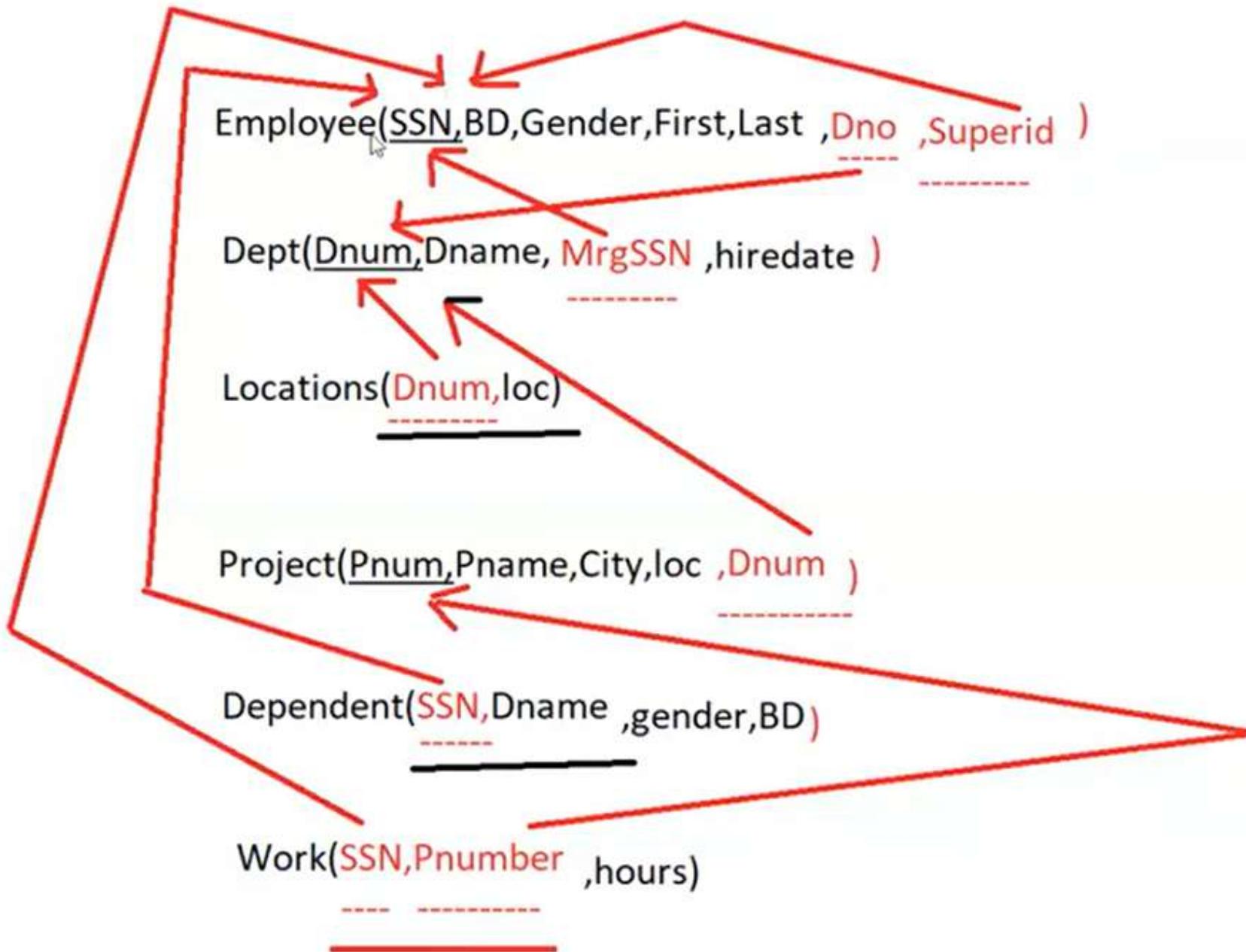
work

Refresh

Add

Close





```
- use ITI
```

```
-| select * from student
```

sdfs  
 dfsd  
fsdf  
 sdf  
 sdf

200 %

Results Messages

	St_Id	St_Fname	St_Lname	St_Address	St_Age	Dept_Id	St_super
1	1	Ahmed	Hassan	Cairo	20	10	NULL
2	2	Amr	Magdy	Cairo	21	10	1
3	3	Mona	Saleh	Cairo	22	10	1
4	4	Ahmed	Mohamed	Alex	23	10	1
5	5	Khalid	Moahmed	Alex	24	10	1
6	6	Heba	Farouk	Mansoura	25	20	NULL
7	7	Ali	Hussien	Cairo	25	20	6
8	8	Mohamed	Fars	Alex	28	20	6
9	9	Saly	Ahmed	Mansoura	24	30	NULL
10	10	Fady	Ali	Alex	24	30	9
11	11	Marwa	Ahmed	Cairo	24	30	9
12	12	Noha	Omar	Cairo	21	40	NULL
13	13	Said	NULL	NULL	NULL	40	12



Query executed successfully.

City of San José.

Microsoft Business Applications Developer.

FK a Primary key in another table.

unique constraint on foreign key 1 To 1.

المغرب في المسجد.

San Jose, California.

```
- use ITI
```

```
select * from student
```

```
- insert into Student(st_id,st_fname)  
values(55, 'ali')
```

```
- /*
```

200 %

 Messages

Command(s) completed successfully.

```
- use ITI
```

```
select * from student
```

```
- insert into Student(st_id,st_fname)  
values(55,'ali')
```

```
- /*
```

200 %

 Messages

(1 row(s) affected)

```
create table emp
(
    eid int Primary key,
    ename varchar(20) not null,
    eage int,
    eadd varchar(20) default 'cairo',
    hiredate date default getdate(),
    Dnum int
```

```
create table emp
(
    eid int Primary key,
    ename varchar(20) not null,
    eage int,
    eadd varchar(20) default 'cairo',
    hiredate date default getdate(),
    Dnum int
)
```

```
    hiredate date default getdate(),
Dnum int
)

alter table emp add sal int

alter table emp alter column sal bigint

alter table emp drop column sal

drop table emp
```

SQLQuery2.sql - (I...-VF50P25\Rami (63)) \* X

```
alter table emp alter column sal tinyint
```

```
alter table emp drop column sal
```

```
drop table emp
```

```
--DML
```

```
--insert update delete
```

```
insert into emp
```

```
values(1,'ali',NULL,'alex','1/1/2010',NULL)
```

```
alter table emp drop column sal
```

```
drop table emp
```

```
--DML
```

```
--insert update delete
```

```
- insert into emp
```

```
values(1,'ali',NULL,'alex','1/1/2010',NULL)
```

```
- insert into emp(ename,eid)
```

```
values('eman',2)
```

```
[
```

SQLQuery2.sql - (1...-VF50P25\Rami (63))

```
- insert into emp
  values(1, 'ali',NULL,'alex','1/1/2010',NULL)

- insert into emp(ename,eid)
  values('eman',9)

--insert constructor
- insert into emp(ename,eid)
  values('eman',8),('ali',12),('nada',7)
```

```
- insert into emp(ename,eid)
  values('eman',8),('ali',12),('nada',7)

- update emp
    set ename='omar'
  where eid=1

- update emp
    set ename='omar',eage=22
  where eid=1
```

```
values('eman',8),('ali',12),('nada',7)
```

```
- update emp  
    set ename='omar'  
  where eid=1
```

```
- update emp  
    set ename='omar', eage=22  
  where eid=1
```

```
- update emp  
    set eage+=1
```

200 %

Messages

(6 row(s) affected)

SQLQuery2.sql - (L...-VF50P25\Rami (63)) X

upunur clip

set ename='omar'

delete from emp

drop table emp

SQLQuery2.sql - (L...-VF50P25\Rami (63)) \* X

drop table emp

-- DQL

select \* from Student

SQLQuery2.sql - (L...-VF50P25\Rami (63))

```
select * from Student
```

```
select st_id,st_fname from Student
```

```
select st_id,st_fname from Student
```

```
select st_id,st_fname from Student
```

```
- select st_id,st_fname from Student  
where st_age>=25
```

```
select * from Student
```

SQLQuery2.sql - (L...-VF50P25\Rami (63)) X

SELECT st\_id,st\_fname FROM Student

- select st\_id,st\_fname from Student  
where st\_age>=25

- select \* from Student  
order by st\_age

SQLQuery2.sql - (L...-VFSOP25\Rami (63)) \* X

SELECT st\_id, st\_fname FROM Student

- select st\_id, st\_fname from Student  
where st\_age >= 25

- select \* from Student  
order by st\_age desc

WITH T AS (SELECT \* FROM Student)

```
- select * from Student  
order by st_age desc
```

```
- select fullname  
from student
```

SQLQuery2.sql - (L...-VF50P25\Rami (63)) \* X

```
select * from Student  
order by st_age desc
```

```
select st_fname+' '+st_lname  
from student
```

SQLQuery2.sql - (L...-VF50P25\Rami (63))

virtual query editor mode

```
- select st_fname+' '+st_lname [full name]
  from student
```

```
| select [full name]
| from Student
```

SQLQuery2.sql - (I...-VF50P25\Rami (63)) \* X

```
      view usage user
[-] select st_fname+' '+st_lname [full name]
  from student
```

```
[-] select *
  from Student
```

SQLQuery2.sql - (L...-VF50P25\Rami (63))

```
from Student  
where st_fname is NULL
```

```
select *  
from Student  
where st_fname is not NULL
```

SQLQuery2.sql - (I...-VF50P25\Rami (63))\* X

```
select *  
from Student  
where st_fname is NULL  
  
select *  
from Student  
where st_fname is not NULL and st_lname is not null
```

SQLQuery2.sql - (L...VF50P25\Rami (63)) \* X

```
select  
from Student  
where st_fname is NULL
```

```
- select *  
from Student  
where st_fname is not NULL and st_lname is not null
```

SQLQuery2.sql - (l...-VF50P25\Rami (63))\*

```
- select distinct st_fname  
  from Student  
  
-|select *  
  from student  
 where st_address='mansoura' and st_Address='alex'
```

SQLQuery2.sql - (I...-VF50P25\Rami (63))\*

```
- select distinct st_fname  
  from Student  
  
-| select *  
  from student  
  where st_address='mansoura' and st_Address='alex'
```

SQLQuery2.sql - (l...-VF50P25\Rami (63))\*

```
from student  
where st_address='mansoura' or st_Address='alex'
```

```
- select *  
from student  
where st_address in('cairo', 'alex', 'mansoura') I
```

SQLQuery2.sql - (L...-VF50P25\Rami (63)) \* X

```
from student
where st_address in('cairo','alex','mansoura')
```

```
-|select *
from student
where st_address='mansoura' and st_age>=25
```

SQLQuery2.sql - (L...-VF50P25\Rami (63)) X

```
-| select *
  from student
  where st_age >= 23 and st_age >= 25
```

```
-| select *
  from Student
  where st_age between 20 and 30
```

SQLQuery2.sql - (I...-VFSOP25\Rami (63)) \* X

```
- select *
  from student
 where st_age >= 23    and st_age >= 25
```

```
- select *
  from Student
 where st_age between 23 and 25
```

SQLQuery2.sql - (...-VF50P25\Rami (63)) \* X

-- DML Structure

where st\_age between 23 and 25

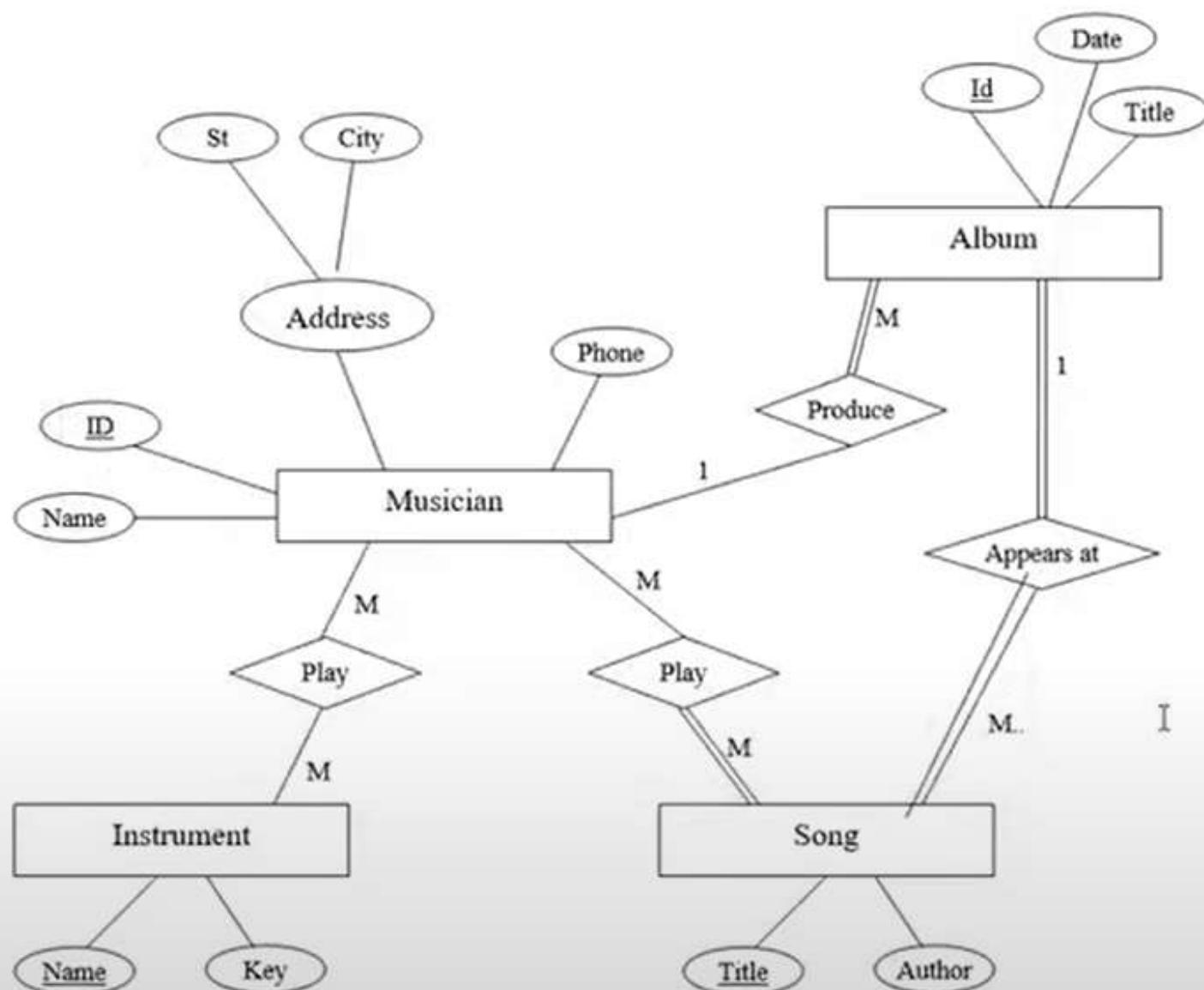
- --Mapping

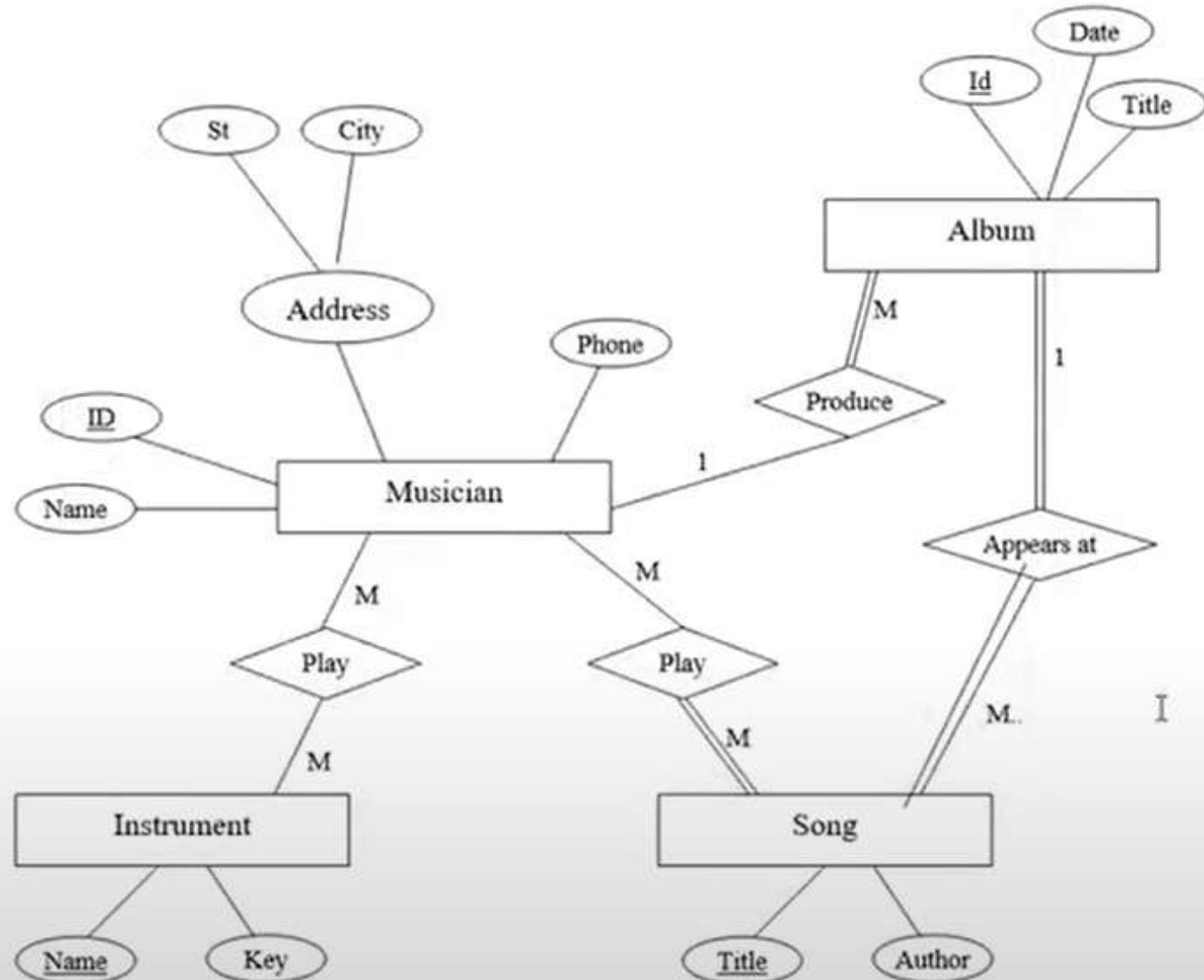
--Create DB wizard

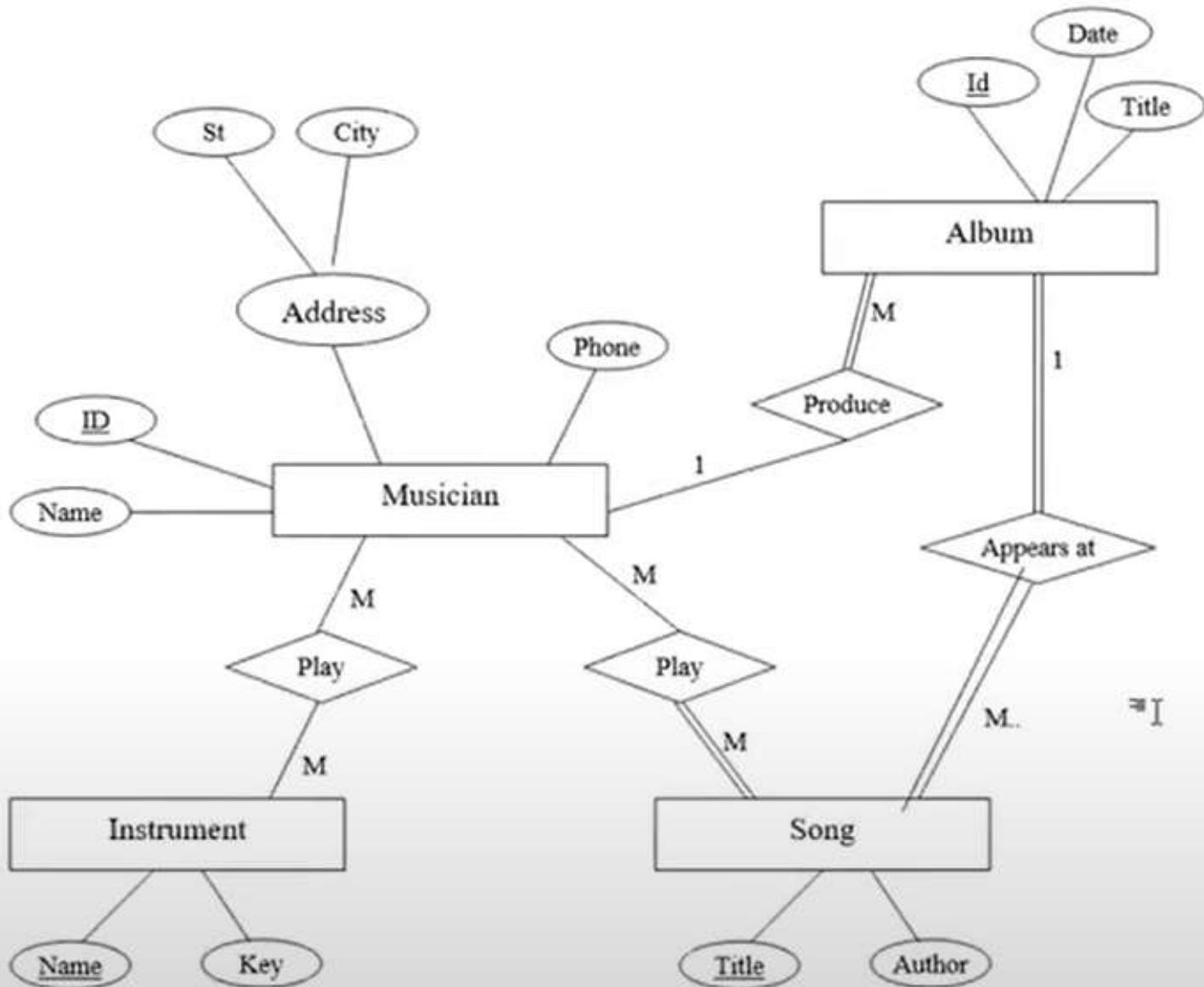
--DDL

--DML

--DQL|



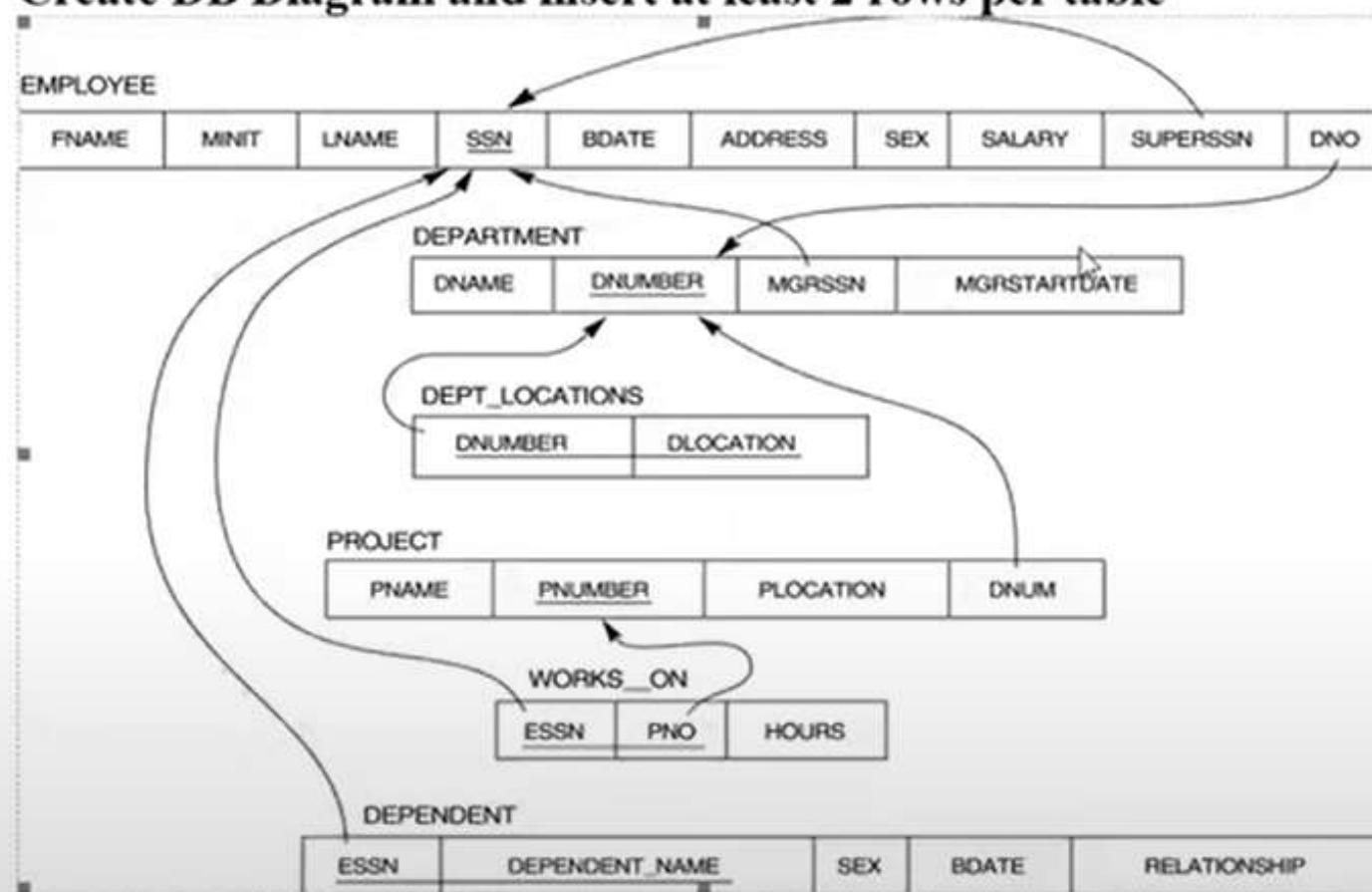




## Company Database Schema

1) Create Database using wizard with the following schema  
then

Create DB Diagram and insert at least 2 rows per table

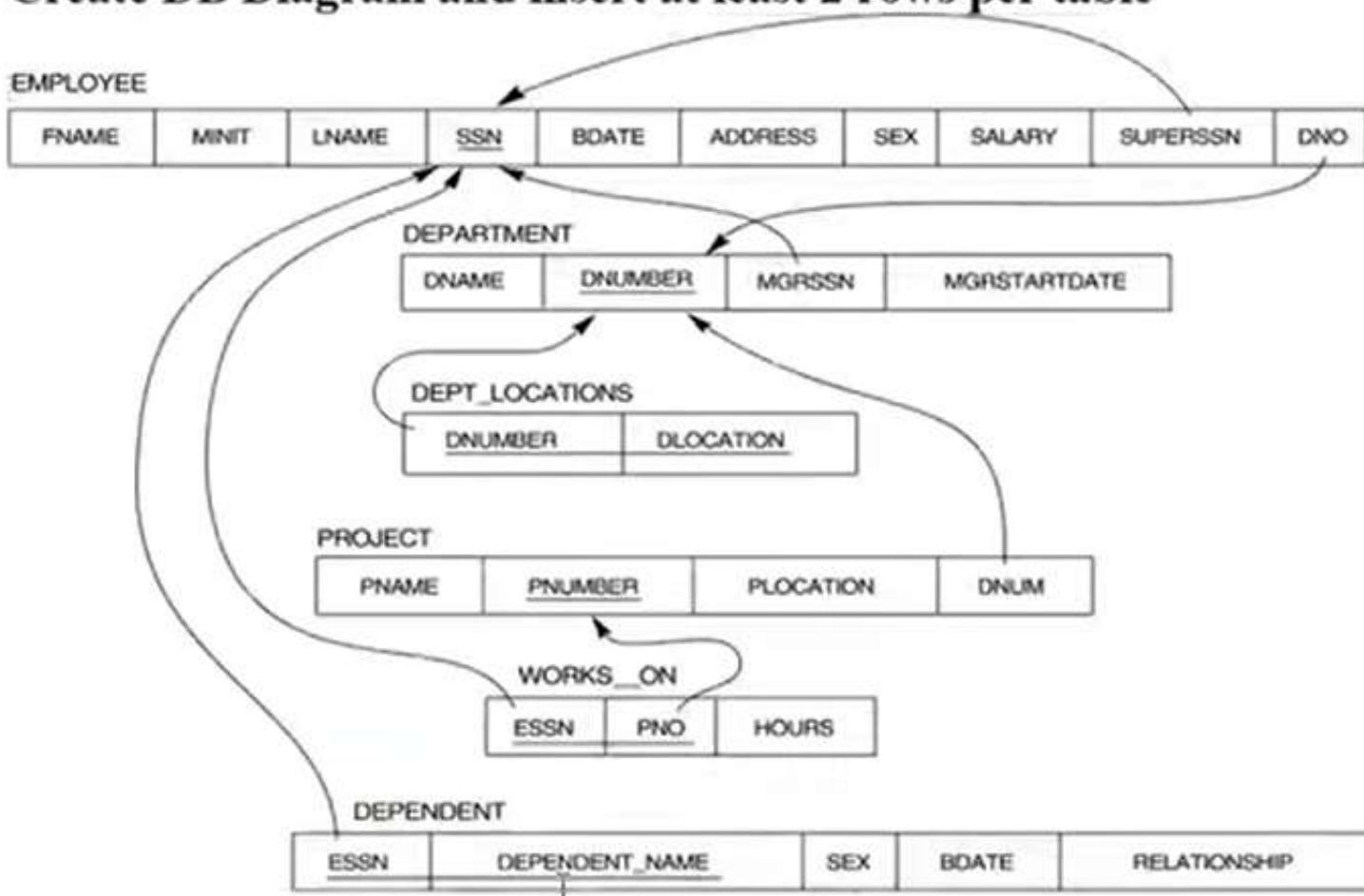


2) Restore Company DB then  
Try to create the following Queries:

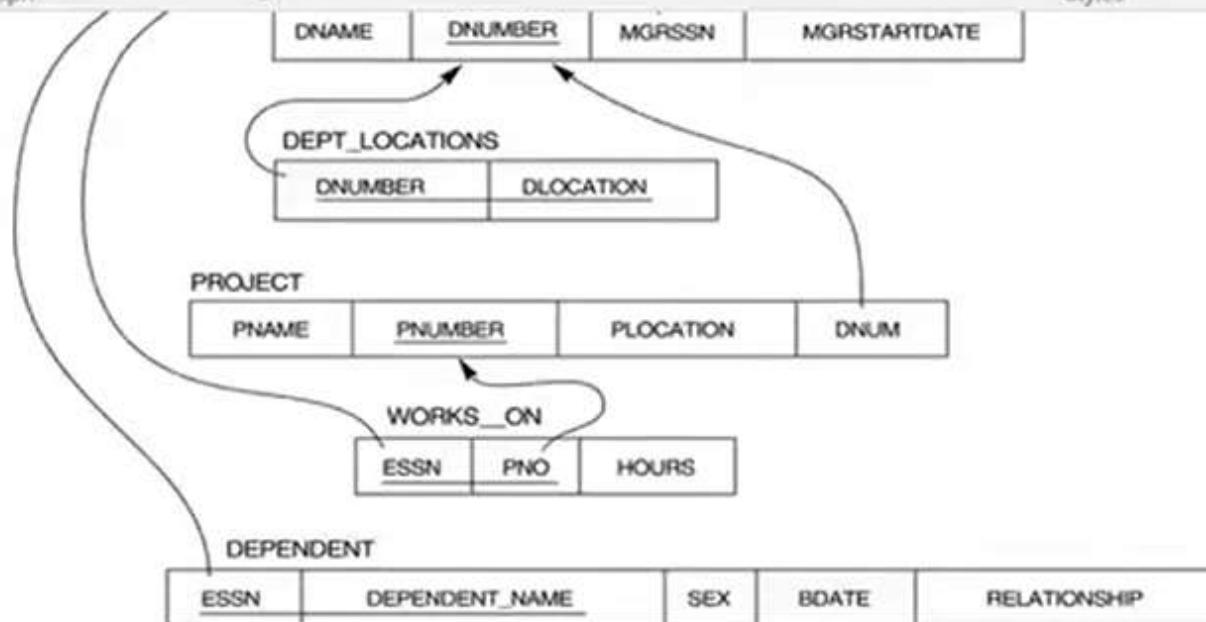
## Company Database Schema

1) Create Database using wizard with the following schema  
then

Create DB Diagram and insert at least 2 rows per table

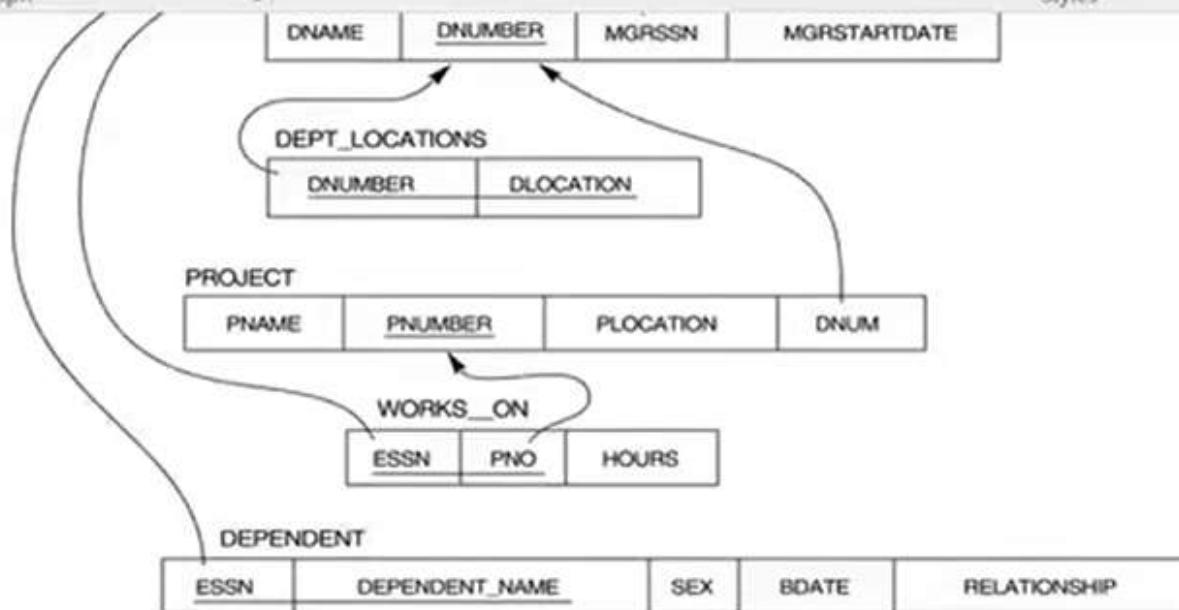


2) Restore Company DB then  
Try to create the following Queries:



**2) Restore Company DB then  
Try to create the following Queries:**

1. Display all the employees Data.
2. Display the employee First name, last name, Salary and Department number.
3. Display all the projects names, locations and the department which is responsible about it.
4. If you know that the company policy is to pay an annual commission for each employee with specific percent equals 10% of his/her annual salary .Display each employee full name and his annual commission in an ANNUAL COMM column (alias).
5. Display the employees Id, name who earns more than 1000 LE monthly.
6. Display the employees Id, name who earns more than 10000 LE annually.
7. Display the names and salaries of the female employees
8. Display each department id, name which managed by a manager with id equals 968574.
9. Dispaly the ids, names and locations of the pojects which controled with department 10.



**2) Restore Company DB then  
Try to create the following Queries:**

1. Display all the employees Data.
2. Display the employee First name, last name, Salary and Department number.
3. Display all the projects names, locations and the department which is responsible about it.
4. If you know that the company policy is to pay an annual commission for each employee with specific percent equals 10% of his/her annual salary .Display each employee full name and his annual commission in an ANNUAL COMM column (alias).
5. Display the employees Id, name who earns more than 1000 LE monthly.
6. Display the employees Id, name who earns more than 10000 LE annually.
7. Display the names and salaries of the female employees
8. Display each department id, name which managed by a manager with id equals 968574.
9. Dispaly the ids, names and locations of the projects which controled with department 10.

SQL Server (MSSQLSERVER)	Provides sto...	Running	Automatic	\Rami
SQL Server Agent (ADEX)	Executes job...	Running	Automatic	\Rami

## Types of joins

--Cross join

-----Cartesian Product

--Inner Join

-----Equi Join

--Outer Join

-----Left Outer join

-----Right Outer Join

-----Full Outer Join

--Self Join

(Unary relationship)

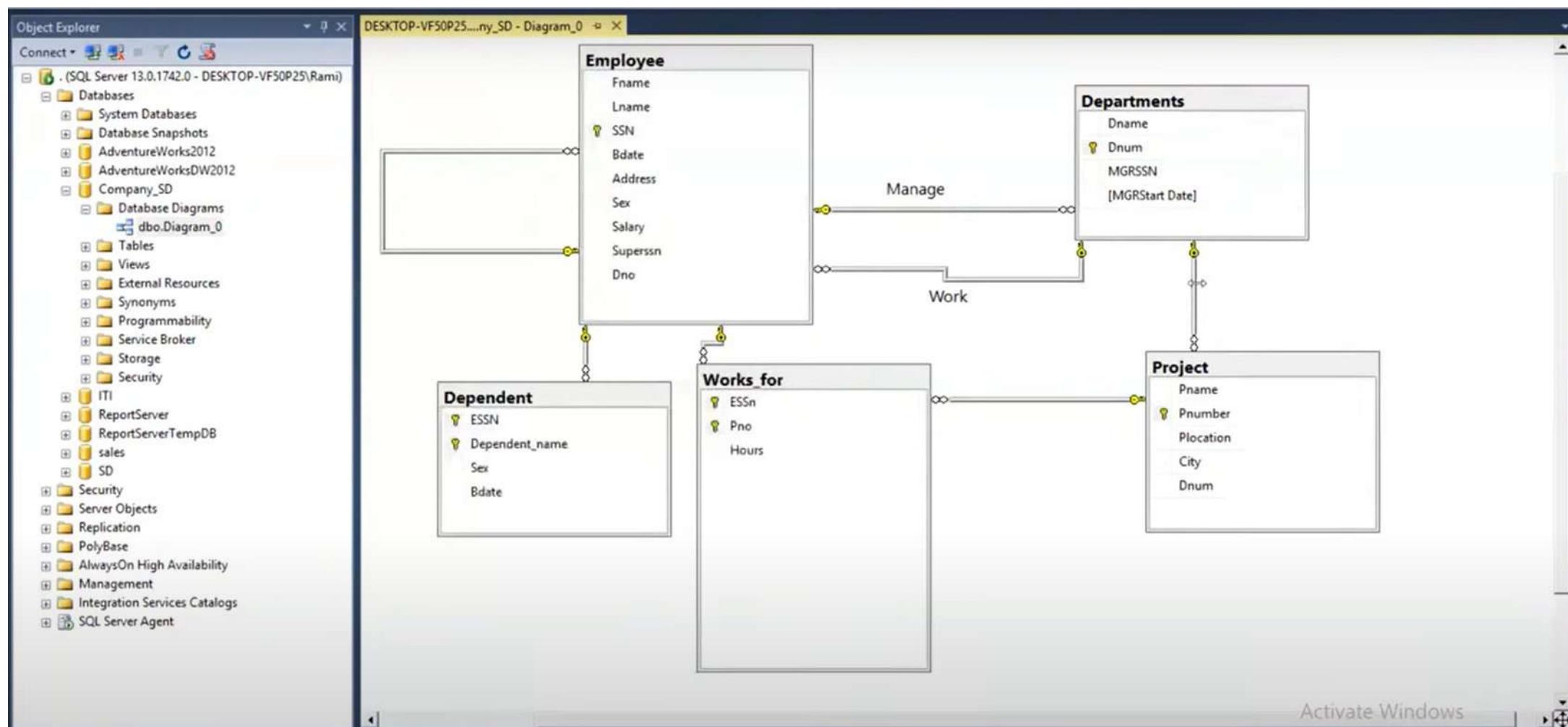
Student

Sid	Sname	did
1	ahmed	10
2	khalid	10
3	eman	20
3	omar	NULL

Dept

Did	Dname
10	SD
20	HR
30	IS
40	Admin

Activate Windows



## Types of joins

--Cross join  
-----Cartesian Product -- 16 rows

--Inner Join  
-----Equi Join

--Outer Join  
-----Left Outer join  
-----Right Outer Join  
-----Full Outer Join

--Self Join  
(Unary relationship)

Select Sname , Dname  
From Student    Cross join    Dept

Sname	Dname
ahmed	SD
khalid	SD
eman	SD
Omar	SD
ahmed	HR
khalid	HR
Eman	HR
Omar	HR

Student

Sid	Sname	did
1	ahmed	10
2	khalid	10
3	eman	20
3	omar	NULL

Dept

Did	Dname
10	SD
20	HR
30	IS
40	Admin

Activate Windows

Get a free Windows 10 license at [www.microsoft.com/windows/activation](http://www.microsoft.com/windows/activation)

----- Cartesian Product -- 16 rows

--Inner Join

-----Equi Join PK = FK

--Outer Join

-----Left Outer join

-----Right Outer Join

-----Full Outer Join

--Self Join

(Unary relationship)

Select Sname , Dname

From Student S , Dept D

Where Dept.did = Student.did

X = Y

Sname	Dname
ahmed	SD
khalid	SD
eman	HR

Student

Sid	Sname	did
1	ahmed	10
2	khalid	10
3	eman	20
3	omar	NULL

Dept

Did	Dname
10	SD
20	HR
30	IS
40	Admin

Activate Windows

## Types of joins

--Cross join  
-----Cartesian Product -- 16 rows

--Inner Join -- 3 rows  
-----Equi Join PK = FK

--Outer Join  
-----Left Outer join -- 4 rows  
-----Right Outer Join -- 5 rows  
-----Full Outer Join -- 6 rows

--Self Join  
(Unary relationship)

Select Sname , Dname  
 left right  
 From Student S full outer join Dept D  
 On D.Did = S.Did

I	
Sname	Dname
ahmed	SD
khalid	SD
eman	HR
omar	NULL
+	
Sname	Dname
ahmed	SD
Kahlid	SD
eman	HR
NULL	IS
NULL	Admin
=	
Sname	Dname
ahmed	SD
khalid	SD
eman	HR
Omar	NULL
NULL	IS
NULL	Admin

Student Y		
Sid	Sname	did
1	ahmed	10
2	khalid	10
3	eman	20
3	omar	NULL

Dept X	
Did	Dname
10	SD
20	HR
30	IS
40	Admin

Activate Windows

## Self join

Select X.Ename as EmpName , Y.Ename as SuperName

From Employee X , Employee Y

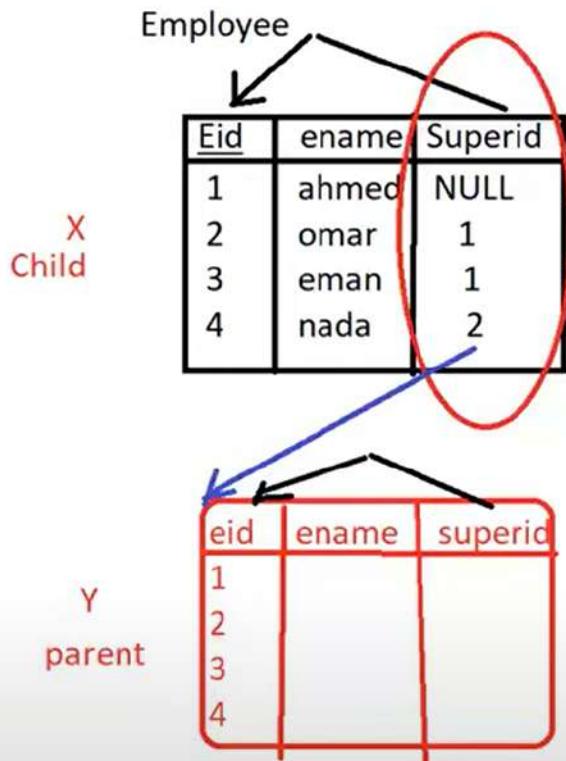
where Y.eid = X.superid

Empname	SuperName
omar	Ahmed
eman	Ahmed
nada	omar

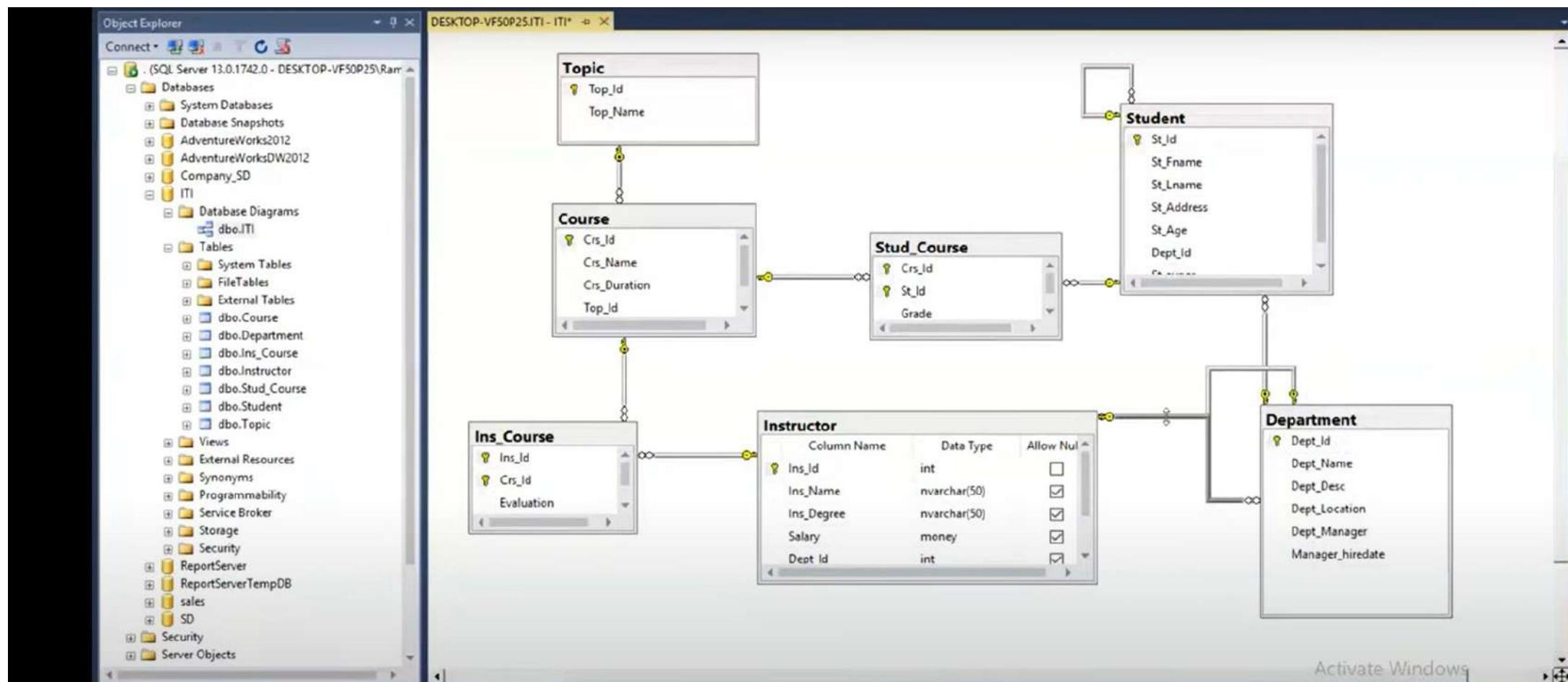
Select X. Ename , Y.\*

From employee X , employee Y

where y.eid=x.superid  
parent      Child  
Super      employee



Activate Windows



```
- select st_fname,dept_name  
  from Student , Department  
  
- select st_fname,dept_name  
  from Student cross join Department I
```

```
- [+] select st_fname,dept_name  
      from Student cross join Department  
  
- [+] select st_fname,dept_name  
      from Student , Department  
      where Department.Dept_Id=Student.Dept_Id
```

```
- select st_fname,dept_name  
  from Student , Department  
 where Department.Dept_Id=Student.Dept_Id  
  
- select st_fname,dept_name  
  from Student S , Department D  
 where D.Dept_Id=S.Dept_Id
```

I

```
- select st_fname,dept_name,d.dept_id  
from Student S , Department D  
where D.Dept_Id=S.Dept_Id
```

```
- select st_fname,*  
from Student S , Department D  
where D.Dept_Id=S.Dept_Id
```

```
where D.Dept_Id=S.Dept_Id
```

```
- select st_fname,dept_name  
from Student S , Department D  
where D.Dept_Id=S.Dept_Id and st_address='alex'  
order by dept_name
```

```
- select st_fname,dept_name  
  from Student S inner join Department D  
    on D.Dept_Id=S.Dept_Id
```

```
- select st_fname,dept_name  
  from Student S inner join Department D  
    on D.Dept_Id=S.Dept_Id and St_Address='alex'
```

```
- select st_fname,dept_name  
  from Student S left outer join Department D  
    on D.Dept_Id=S.Dept_Id
```

```
- select st_fname,dept_name  
  from Student S right outer join Department D  
    on D.Dept_Id=S.Dept_Id
```

```
└─ Select X.St_Fname as Sname , Y.*  
  from Student X , Student Y  
  where Y.St_Id=X.St_super
```

```
Select X.St_Fname as Sname , Y.*  
from Student X inner join| Student Y  
where Y.St_Id=X.St_super
```

```
- Select X.St_Fname as Sname , Y.*  
from Student X , Student Y  
where Y.St_Id=X.St_super
```

```
- Select X.St_Fname as Sname , Y.*  
from Student X inner join Student Y  
on Y.St_Id=X.St_super
```

```
--join Multi tables
select st_fname,crs_name,grade
from Student S , Stud_Course SC , Course C
where S.St_Id=Sc.St_Id and
      c.Crs_Id=Sc.Crs_Id
```

```
- select st_fname,crs_name,grade,dept_name  
from Student S inner join Stud_Course SC  
on S.St_Id=Sc.St_Id  
inner join  
Course C  
on c.Crs_Id=Sc.Crs_Id  
inner join  
Department D  
on d.Dept_Id=s.Dept_Id
```

```
--join DML
-- join update
update Stud_Course
    set grade+=10

select grade
from Student S , Stud_Course sc
where s.St_Id=sc.St_Id and St_Address='cairo'
```

```
-- join update
update Stud_Course
    set grade+=10
from Student S , Stud_Course sc
where s.St_Id=sc.St_Id and St_Address='cairo'

update SC
    set grade+=10
from Student S , Stud_Course SC
where s.St_Id=sc.St_Id and St_Address='cairo'
```

```
- select isnull(st_fname,'')
  from Student
```

```
- select isnull(st_fname,'has No Name')
  from Student
```

```
select isnull(st_fname,s)
  from Student
```

```
[-] select isnull(st_fname,'has No Name')
      from Student

[-] select isnull(st_fname,st_lname)
      from Student

[-] select Coalesce(st_fname,st_lname,st_address,'No Data')
      from Student
```

```
from Student  
      -  
select Coalesce(st_fname,st_lname,st_address,'No Data')  
      from Student  
      -  
select st_fname,st_age  
      from student  
      -  
select st_fname+' '+convert(varchar(2),st_age)  
      from student
```

```
from Student  
-  
select st_fname,st_age  
from student  
-  
select st_fname+' '+convert(varchar(2),st_age)  
from student  
-  
select 'studName= '+st_fname+ ' &age= '+convert(varchar(2),st_age)  
from student
```

```
- select *
  from Student
 where st_fname = 'ahmed'
```

```
- select *
  from Student
 where st_fname like 'ahmed'
```

```
where st_fname = 'ahmed'
```

```
- select *
```

```
from Student
```

```
where st_fname like 'a%'
```

```
- one char
```

```
% zero or More char
```

```
from Student  
where st_fname like 'a%'
```

- one char

% zero or More char

```
select *
```

```
from Student
```

```
where st_fname like '_a%'
```

'a%h'  
'%a\_'

'ahm%'

'[ahm]%'

'[^ahm]%'

'[a-h]%'

'[^a-h]%'

'[346]%'

'%[%]'

'%[\_]%'

'[][]'

'[\_]%'[\_] \_ahmed\_

- select st\_fname,dept\_id,st\_age  
from Student  
order by st\_address

-|select st\_fname,dept\_id,st\_age  
from Student  
order by 1 |

```
- select st_fname,dept_id,st_age  
  from Student  
  order by 3  
  
- select st_fname,dept_id,st_age  
  from Student  
  order by dept_id asc,st_age desc
```

```
order by dept_id asc,st_age desc
```

```
[  
select st_fname,dept_id,st_age  
from Student  
order by st_id asc,st_age desc
```

```
select top
```

```
select st_name, st_prcnt, st_age  
from Student  
order by st_id asc, st_age desc
```

I

```
select top(2)*  
from Student  
where  
order by
```

```
from Student
order by 3

select st_fname,dept_id,st_age
from Student
order by dept_id asc,st_age desc

select st_fname,dept_id,st_age
from Student
order by st_id asc,st_age desc
```

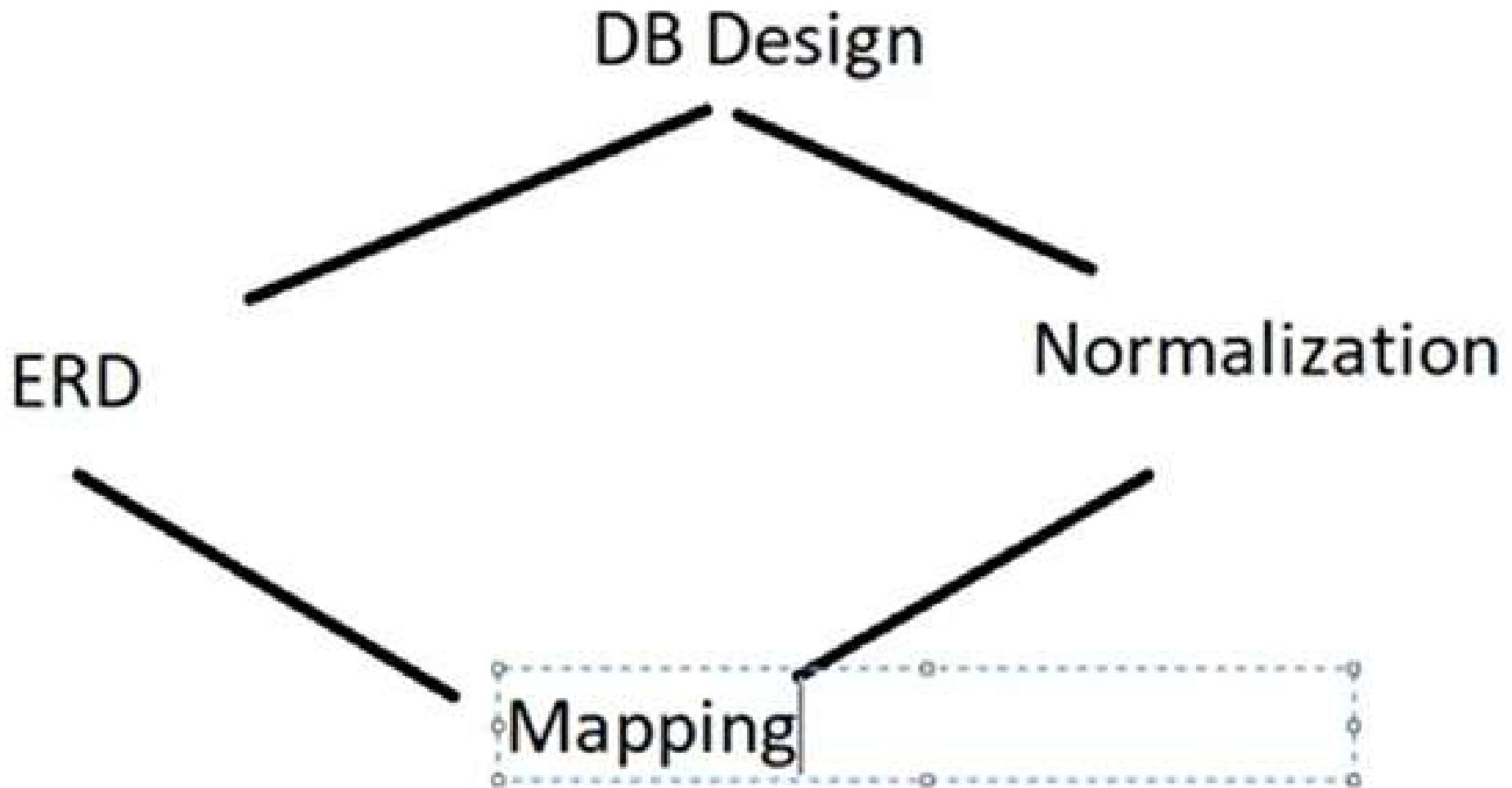
```
- select st_fname,dept_id,st_age  
from Student  
order by st_address
```

I

```
- select st_fname,dept_id,st_age  
from Student  
order by 1
```

```
- select st_fname,dept_id,st_age  
from Student
```

```
'[^ahm]%'  
'[a-h]%'      --range  
'[^a-h]%'  
'[346]%'  
'%[%]'          -----%  
'%[_]%'          ahmed_ali      eman_omar  
'[_]%[_]'        _ahmed_  
  
select st_fname,dept_id,st_age  
from student
```



# Database Normalization

- ▶ **Normalization:** The process of structuring data to minimize duplication and inconsistencies.
- ▶ The process usually involves breaking down a **single Table** into two or more tables and defining 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
  - **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

a table should not have more than one entity type

# Example

<u>SID</u>	<u>Sname</u>	<u>Bdate</u>	<u>City</u>	<u>ZipCode</u>	<u>Subject</u>	<u>Grade</u>	<u>Teacher</u>
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 anomalies exist?

Because we've combined two themes (entity types) into one relation. This results in data redundancy, inconsistency, duplication, and an unnecessary dependency between the entities.

## Functional dependency

- ▶ a constraint between two attributes (columns) or two sets of columns
- ▶  $A \rightarrow B$  if “for every valid instance of A, that value of A uniquely determines the value of B”
- ▶ or ... $A \rightarrow B$  if “existing of B depending on a value of A”

## ... functional dependency

some examples

- social security number determines employee name  
 $SSN \rightarrow ENAME$
- project number determines project name and location  
 $PNUMBER \rightarrow \{PNAME, PLOCATION\}$
- employee ssn and project number determines the hours per week that the employee works on the project  
 $\{SSN, PNUMBER\} \rightarrow HOURS$

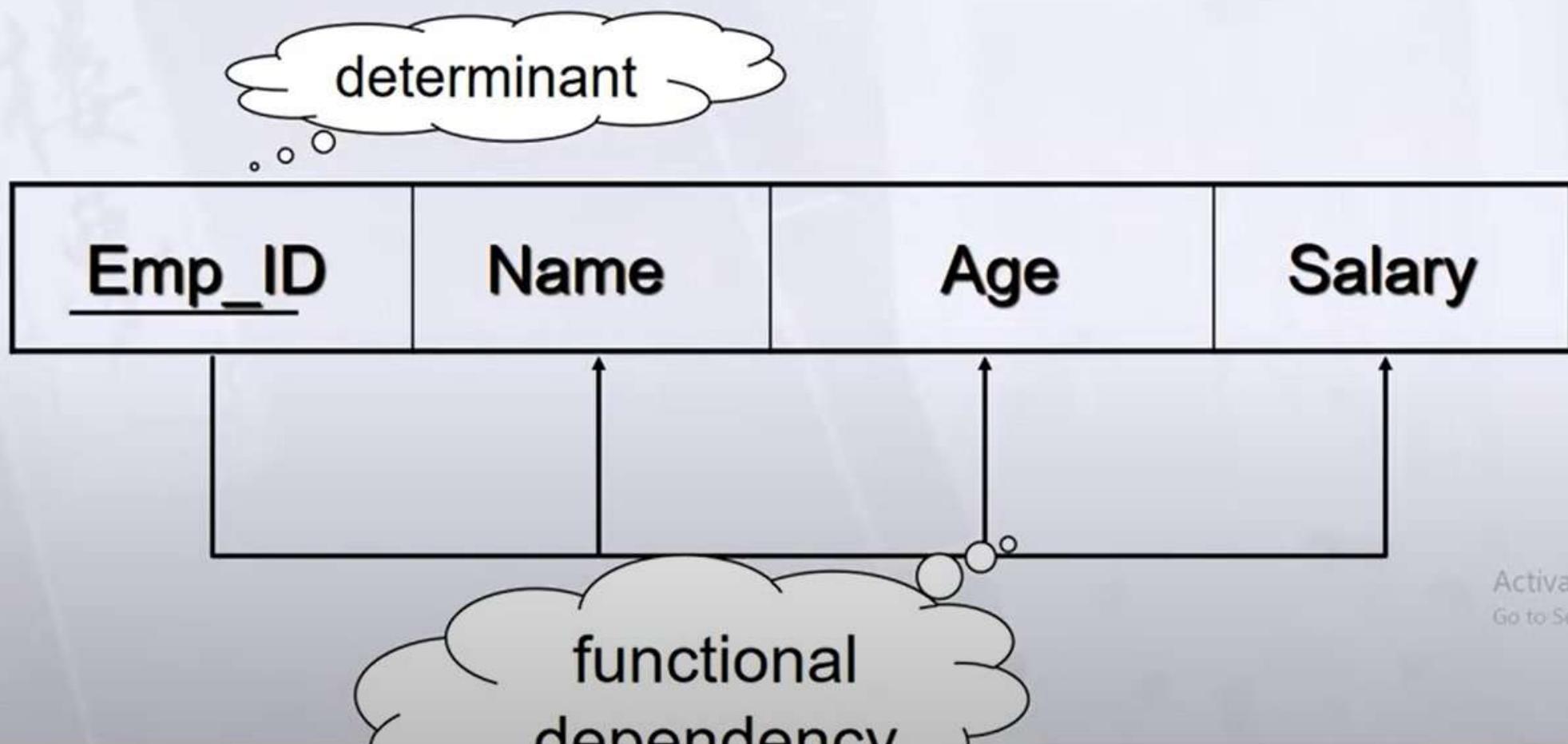
## ... functional dependency

some examples

- social security number determines employee name  
 $SSN \rightarrow ENAME$
- project number determines project name and location  
 $PNUMBER \rightarrow \{PNAME, PLOCATION\}$
- employee ssn and project number determines the hours per week that the employee works on the project  
 $\{SSN, PNUMBER\} \rightarrow HOURS$

# keys and dependencies

## EMPLOYEE1 (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**

- **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)

- **Transitive Functional Dependency**

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

## ... functional dependency

some examples

- social security number determines employee name  
 $SSN \rightarrow ENAME$
- project number determines project name and location  
 $PNUMBER \rightarrow \{PNAME, PLOCATION\}$
- employee ssn and project number determines the hours per week that the employee works on the project  
 $\{SSN, PNUMBER\} \rightarrow HOURS$

# Types of functional dependency

- **Full Functional Dependency**

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

- **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)

- **Transitive Functional Dependency**

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

# Example

<b>SID</b>	<b>SName</b>	<b>Birthdate</b>	<b>City</b>	<b>Zip Code</b>	<b>Subject</b>	<b>Grade</b>	<b>Teacher</b>
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

$\text{Sid}, \text{Subject} \rightarrow \text{Grade}$

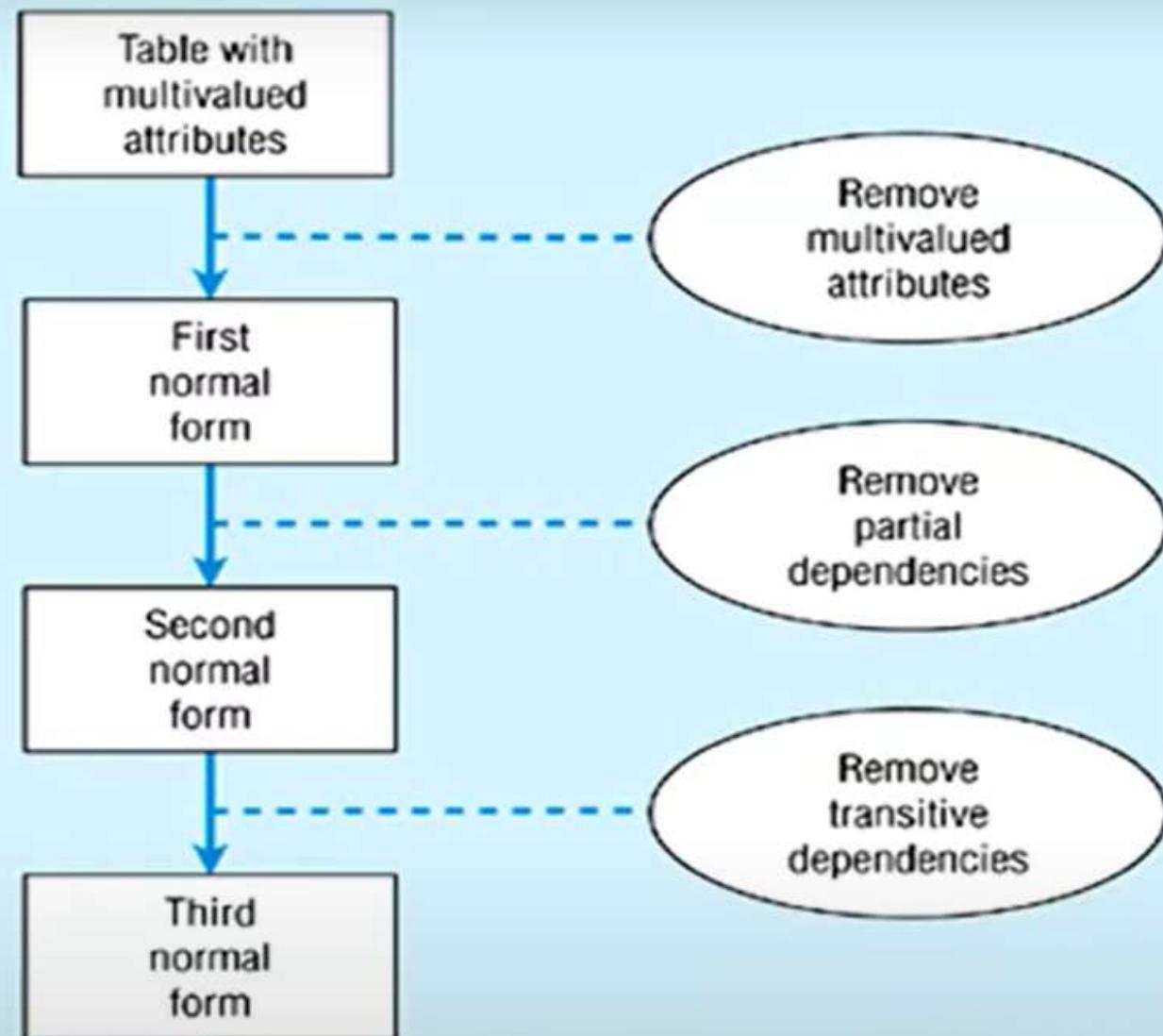
Partial Functional Dependency

$\text{Sid} \rightarrow \text{SName}$   
 $\text{Subect} \rightarrow \text{Teacher}$

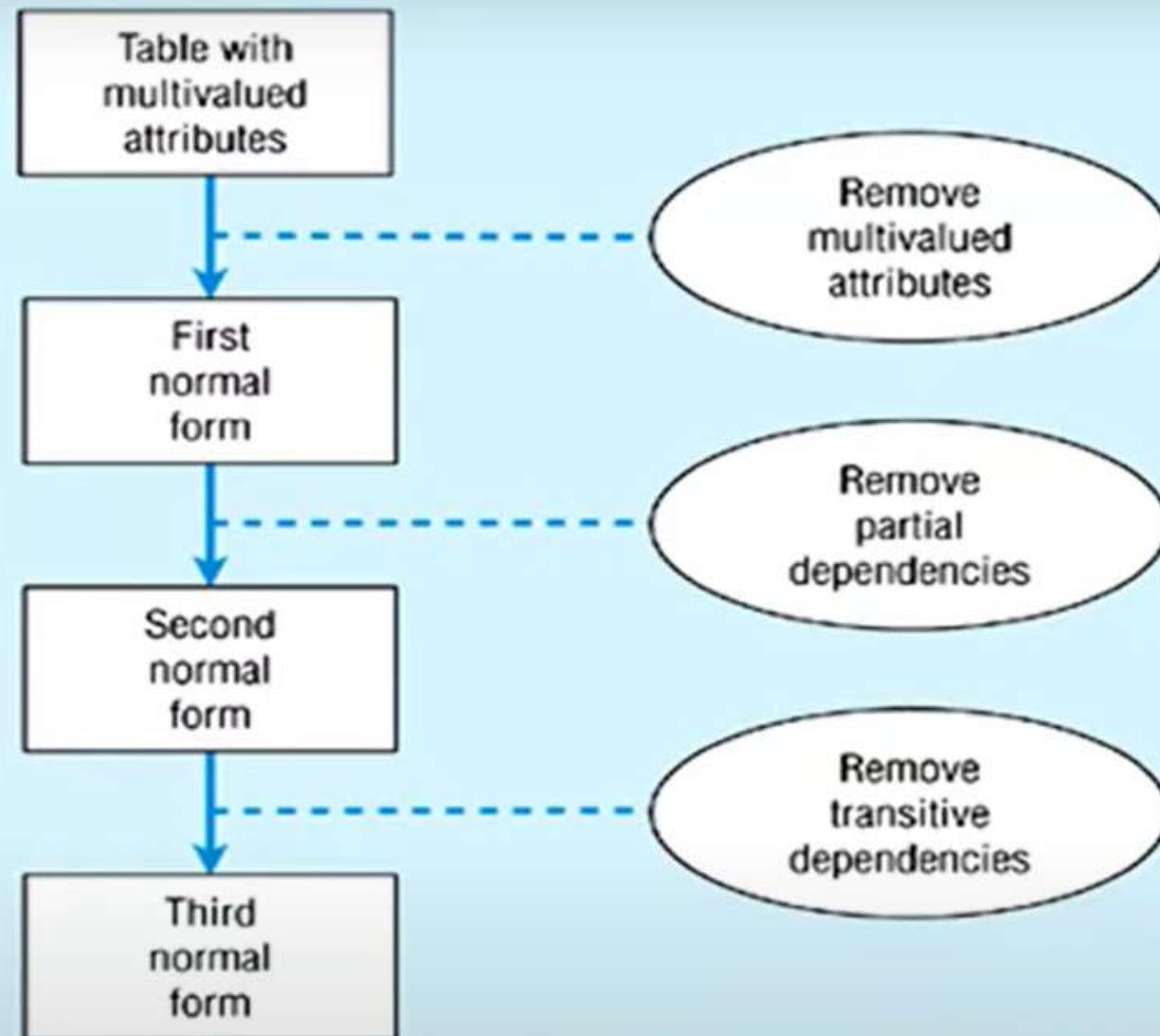
Transitive Functional Dependency

$\text{ZipCode} \rightarrow \text{City}$

# Steps in normalization



# Steps in normalization



# 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

$\text{Sid}, \text{Subject} \rightarrow \text{Grade}$

Partial Functional Dependency

$\text{Sid} \rightarrow \text{SName}$   
 $\text{Subect} \rightarrow \text{Teacher}$

Transitive Functional Dependency

$\text{ZipCode} \rightarrow \text{City}$

You said:

6 thousand 4 hundred Seventeen



ChatGPT said:

The number **6 thousand 4 hundred seventeen** is written numerically as:

**6,417**



You said:

6 thousand 4 hundred Seventy



ChatGPT said:

The number **6 thousand 4 hundred seventy** is written numerically as:

**6,470**



# 1NF

- 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

# Example

<u>SID</u>	<u>SName</u>	<u>Birthdate</u>	<u>City</u>	<u>Zip Code</u>	<u>Subject</u>	<u>Grade</u>	<u>Teacher</u>
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

<u>SID</u>	<u>SName</u>	<u>Birthdate</u>	<u>City</u>	<u>Zip Code</u>	<u>Subject</u>	<u>Grade</u>	<u>Teacher</u>
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



Activate Wi  
Go to Settings>

# Example

<b>SID</b>	<b>SName</b>	<b>Birthdate</b>	<b>City</b>	<b>Zip Code</b>	<b>Subject</b>	<b>Grade</b>	<b>Teacher</b>
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

<b>SID</b>	<b>SName</b>	<b>Birthdate</b>	<b>City</b>	<b>Zip Code</b>	<b>Subject</b>	<b>Grade</b>	<b>Teacher</b>
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



Activate Wi  
Go to Settings

# 1NF

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

<u>SID</u>	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)

<u>SID</u>	<u>Subject</u>	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

# 1NF

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

<u>SID</u>	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)

<u>SID</u>	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

11/3/2020

Activate Windows  
Go to Settings to activate Windows.

Activate Windows  
Go to Settings to activate Windows.



2:24:03 / 3:08:49

seyedEkalawy... hanamahmoud...

Kiro.Abdalier98 shaima7ramadan eren geres

maylie.maped muhammedamr.95 khNoudfawzy07 estam9096 maliamsatataf98

azmakkhaled01 mohamedahmedfathy1995 ryhab\_farous

mohamedahmedfathy1995 ryhab\_farous



# Example

<u>SID</u>	<u>SName</u>	<u>Birthdate</u>	<u>City</u>	<u>Zip Code</u>	<u>Subject</u>	<u>Grade</u>	<u>Teacher</u>
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

<u>SID</u>	<u>SName</u>	<u>Birthdate</u>	<u>City</u>	<u>Zip Code</u>	<u>Subject</u>	<u>Grade</u>	<u>Teacher</u>
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



Activate Win  
Go to Settings

# 1NF

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

<u>SID</u>	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)

<u>SID</u>	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 nonkey attribute is fully functionally dependant on the primary key
- ▶ i.e. remove partial functional dependencies, so no nonkey attribute depends on just part of the key

## 2NF

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

# 1NF

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

<u>SID</u>	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)

<u>SID</u>	<u>Subject</u>	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 nonkey attribute is fully functionally dependant on the primary key
- ▶ i.e. remove partial functional dependencies, so no nonkey attribute depends on just part of the key

# 2NF

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

<u>SID</u>	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

2NF

Because there is no  
Composite PK

Stud\_Subject (SID, Subject, Grade, Teacher)

<u>SID</u>	<u>Subject</u>	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

SID, Subject → Grade.....FFD

Subject → Teacher.....PFD

# 2NF

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

<u>SID</u>	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)

<u>SID</u>	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

# 2NF

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

<u>SID</u>	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

2NF

Because there is no  
Composite PK

Stud\_Subject (SID, Subject, Grade, Teacher)

<u>SID</u>	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

SID, Subject → Grade.....FFD

Subject → Teacher.....PFD

# Example

<b>SID</b>	<b>SName</b>	<b>Birthdate</b>	<b>City</b>	<b>Zip Code</b>	<b>Subject</b>	<b>Grade</b>	<b>Teacher</b>
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

<b>SID</b>	<b>SName</b>	<b>Birthdate</b>	<b>City</b>	<b>Zip Code</b>	<b>Subject</b>	<b>Grade</b>	<b>Teacher</b>
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



Activate Win  
Go to Settings to

# 2NF

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

<u>SID</u>	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)

<u>SID</u>	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

## Third Normal Form

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

# 2NF

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

<u>SID</u>	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

Zip Code  $\rightarrow$  City ..... TFD

Stud\_Subject (SID, Subject, Grade)

<u>SID</u>	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

Because there is no Transitive Functional Dependency

# 3NF

Student(SID, Sname, Birthdate,)

<u>SID</u>	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	<u>Zip Code</u>
Cairo	1010
Alex	1111

Stud\_Subject (SID, Subject, Grade)

<u>SID</u>	<u>Subject</u>	Grade
1	DB	A
1	Math	B
1	WinXP	A
2	DB	B
2	SWE	B
3	NC	C

Subject (Subject, Teacher)

<u>Subject</u>	Teacher
DB1	Hany
Math	Eman
WinXP	khalid
DB2	Hany
SWE	Heba
NC	Mona

# ITI Example

## ITI Students Sheet

**Platform Name : SWE**

**Platform Description: Software Engineering**

**Graduate Manager: Dr.Baha**

Appno	Name	F-code	Faculty	Address	Telno	Grade	Att. Hrs	Sdate
123	Ahmed	SC-phy	Science	Haram	3386842	A	600	14 Sep
124	Mona	Eng-cs	Engineering	Dokki	3389745, 3389744, 5123445	B	591	15 Sep
127	Ali	Com-ac	Commerce	Nasr City	2241593, 2222345	A	550	21 Sep
223	Karim	Med-bio	Medicine	Sheraton	2286845	C	600	14 Sep

# ITI Example

## ITI Students Sheet

**Platform Name : SWE**

**Platform Description: Software Engineering**

**Graduate Manager: Dr.Baha**

Appno	Name	F-code	Faculty	Address	Telno	Grade	Att. Hrs	Sdate
123	Ahmed	SC-phy	Science	Haram	3386842	A	600	14 Sep
124	Mona	Eng-cs	Engineering	Dokki	3389745, 3389744, 5123445	B	591	15 Sep
127	Ali	Com-ac	Commerce	Nasr City	2241593, 2222345	A	550	21 Sep
223	Karim	Med-bio	Medicine	Sheraton	2286845	C	600	14 Sep

# ITI Example

## ITI Students Sheet

**Platform Name : SWE**

**Platform Description: Software Engineering**

**Graduate Manager: Dr.Baha**

Appno	Name	F-code	Faculty	Address	Telno	Grade	Att. Hrs	Sdate
123	Ahmed	SC-phy	Science	Haram	3386842	A	600	14 Sep
124	Mona	Eng-cs	Engineering	Dokki	3389745, 3389744, 5123445	B	591	15 Sep
127	Ali	Com-ac	Commerce	Nasr City	2241593, 2222345	A	550	21 Sep
223	Karim	Med-bio	Medicine	Sheraton	2286845	C	600	14 Sep

## ... functional dependency

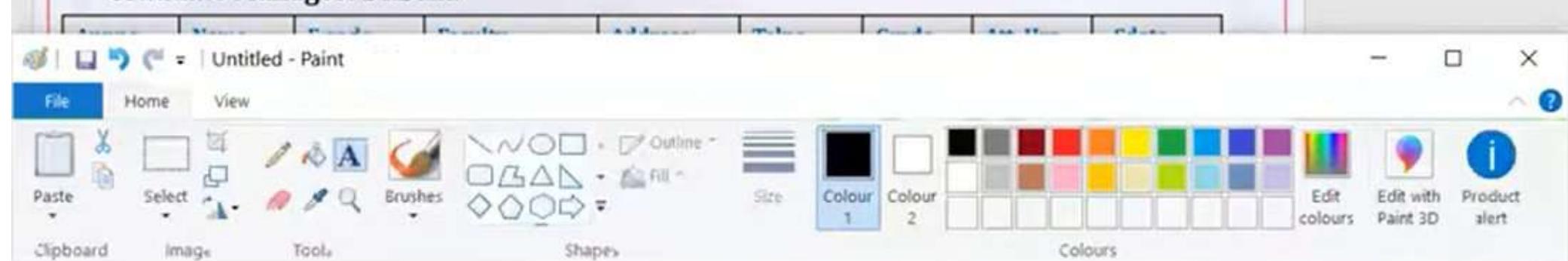
some examples

- social security number determines employee name  
 $SSN \rightarrow ENAME$
- project number determines project name and location  
 $PNUMBER \rightarrow \{PNAME, PLOCATION\}$
- employee ssn and project number determines the hours per week that the employee works on the project  
 $\{SSN, PNUMBER\} \rightarrow HOURS$

# ITI Example

## ITI Students Sheet

**Platform Name : SWE      Platform Description: Software Engineering  
Graduate Manager: Dr.Baha**



PName , PDesc, Pmanager,AppNO,Name,Code,Fac,Add,Tel,G,Att,Sdate

SWE , Sofware,Bahaa,123.....

SWE,Software,bahaa,124.....

||||||||||||||||||,125

||||||||||||||||||,126

Activate Windows  
Go to Settings to activate

☰ 3 SQL Joins, Normalization,

## 1NF :

- **Platform** : pfname , pfdesc , pfManager
- **Students**: pfname, appno, name , faculty , F-Code, address, grade, attd , start\_date
- **Std\_Tel**: appno, telno

## **1NF :**

- **Platform** : pfname , pfdesc , pfManager
- **Students**: pfname, appno, name , faculty , F-  
Code, address, grade, attd , start\_date
- **Std\_Tel**: appno, telno

## **1NF :**

- **Platform** : pfname , pfdesc , pfManager
- **Students**: pfname, appno, name , faculty , F-  
Code, address, grade, attd , start\_date
- **Std\_Tel**: appno, telno

# 2NF

- **Students:** appno, name , faculty , FCode, address
- **Students\_pf:** pfname,appno, grade, attd , start\_date

## Unchanged Tables

- **Platform :**pfname , pfdesc , pfManager
- **Std\_Tel:** appno, telno

## 2NF

- ▶ **Students:** appno, name , faculty , FCode, address
- ▶ **Students\_pf:** pfname,appno, grade, attd , start\_date

### Unchanged Tables

- ▶ **Platform :**pfname , pfdesc , pfManager
- ▶ **Std\_Tel:** appno, telno

- **Students:** appno, name , faculty , FCode, address
- **Students\_pf:** pfname,appno, grade, attd , start\_date

## Unchanged Tables

- **Platform :**pfname , pfdesc , pfManager
- **Std\_Tel:** appno, telno

## 3NF

- **Students:** appno, name , FCode, address
- **Fac\_majors:**faculty , FCode

### Unchanged Tables

- **Platform :**pfname , pfdesc , pfManager
- **Std\_Tel:** appno, telno
- **Students\_pf:** pfname,appno, grade, attd , start\_date

Repeated Cols

Repeating Groups

Eid,Ename,Eadd,Pro\_ID,Pro\_Name,PlocID,PlocCity,hours,Bouns

1st

Employee(Eid,ename,eadd)

Emp\_Pro(Eid,Pro\_id,Pro\_name,PlocID,PlocCity,hours,Bonus)

2nd

Employee (Eid,ename,eadd)

Project(Pro\_id,Pro\_name,PlocID,PLocCity)

Emp\_pro(Eid,Pro\_id,hours,bonus)

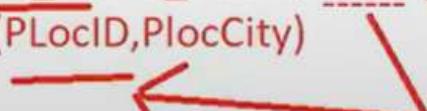
3rd

employee OK

Emp\_Pro ok

project(Pro\_id,Pro\_Name,Plocid)

locations(PLocID,PlocCity)



Repeated Cols

Repeating Groups

Eid,Ename,Eadd,Pro\_ID,Pro\_Name,PlocID,PlocCity,hours,Bouns

1St

Employee(Eid,ename,eadd)

Emp\_Pro(Eid,Pro\_id,Pro\_name,PlocID,PlocCity,hours,Bonus)

2nd

Employee (Eid,ename,eadd)

Project(Pro\_id,Pro\_name,PlocID,PLocCity)

Emp\_pro(Eid,Pro\_id,hours,bonus)

3rd

employee OK

Emp\_Pro ok

project(Pro\_id,Pro\_Name,Plocid)

locations(PLocID,PlocCity)

Repeated Cols

Repeating Groups

Eid,Ename,Eadd,Pro\_ID,Pro\_Name,PlocID,PlocCity,hours,Bouns

1st

Employee(Eid,ename,eadd)

Emp\_Pro(Eid,Pro\_id,Pro\_name,PlocID,PlocCity,hours,Bonus)

2nd

Employee (Eid,ename,eadd)

Project(Pro\_id,Pro\_name,PlocID,PLocCity)

Emp\_pro(Eid,Pro\_id,hours,bonus)

3rd

employee OK

Emp\_Pro ok

project(Pro\_id,Pro\_Name,Plocid)

locations(PLocID,PlocCity)

Repeated Cols                  Repeating Groups

Eid,Ename,Eadd,Pro\_ID,Pro\_Name,PlocID,PlocCity,hours,Bouns

1st

-----  
Employee(Eid,ename,eadd)  
Emp\_Pro(Eid,Pro\_id,Pro\_name,PlocID,PlocCity,hours,Bonus)

2nd

-----  
Employee (Eid,ename,eadd)  
Project(Pro\_id,Pro\_name,PlocID,PLocCity)  
Emp\_pro(Eid,Pro\_id,hours,bonus)

3rd

-----  
employee    OK  
Emp\_Pro    ok  
project(Pro\_id,Pro\_Name,Plocid)  
locations(PLocID,PlocCity)

Activate Windows

## 3NF

- ▶ **Students:** appno, name , FCode, address
- ▶ **Fac\_majors:** faculty , FCode

## Unchanged Tables

- ▶ **Platform :**pfname , pfdesc , pfManager
- ▶ **Std\_Tel:** appno, telno
- ▶ **Students\_pf:** pfname,appno, grade, attd , start\_date

# ITI Example

## ITI Students Sheet

Platform Name : SWE

Platform Description: Software Engineering

Graduate Manager: Dr.Baha

Appno	Name	F-code	Faculty	Address	Telno	Grade	Att. Hrs	Sdate
123	Ahmed	SC-phy	Science	Haram	3386842	A	600	14 Sep
124	Mona	Eng-cs	Engineering	Dokki 3389745, 3389744, 5123445		B	591	15 Sep
127	Ali I	Com-ac	Commerce	Nasr City 2241593, 2222345		A	550	21 Sep
223	Karim	Med-bio	Medicine	Sheraton	2286845	C	600	14 Sep

'[^ahm]%'

'[a-h]%' -- range

'[^a-h]%'

'[346]%'

'%[%]' -----%

'%[\_]%' ahmed\_ali eman\_omar

'[\_]%[\_]' \_ahmed\_

select st\_fname,dept\_id,st\_age  
from student

Repeated Cols

Repeating Groups

Eid,Ename,Eadd,Pro\_ID,Pro\_Name,PlocID,PlocCity,hours,Bouns

1St

Employee(Eid,ename,eadd)

Emp\_Pro(Eid,Pro\_id,Pro\_name,PlocID,PlocCity,hours,Bonus)

2nd

Employee (Eid,ename,eadd)

Project(Pro\_id,Pro\_name,PlocID,PLocCity)

Emp\_pro(Eid,Pro\_id,hours,bonus)

3rd

employee OK

Emp\_Pro ok

project(Pro\_id,Pro\_Name,Plocid)

locations(PLocID,PlocCity)

|Example1

## Sales Order

*Fiction Company  
202 N. Main  
Manhattan, KS 66502*

<b>CustomerNumber:</b>	<i>1001</i>	<b>Sales Order Number:</b>	<i>405</i>
<b>Customer Name:</b>	<i>ABC Company</i>	<b>Sales Order Date:</b>	<i>2/1/2000</i>
<b>Customer Address:</b>	<i>100 Points Manhattan, KS 66502</i>	<b>Clerk Number:</b>	<i>210</i>
		<b>Clerk Name:</b>	<i>Martin Lawrence</i>

Item Ordered	Description	Quantity	Unit Price	Total
800	<i>widgit small</i>	40	60.00	2,400.00
801	<i>tingimajigger</i>	20	20.00	400.00
805	<i>thingibob</i>	10	100.00	1,000.00

**\* Try to create the following Queries:**

1. Display the Department id, name and id and the name of its manager.
2. Display the name of the departments and the name of the projects under its control.
3. Display the full data about all the dependence associated with the name of the employee they depend on him/her.
4. Display the Id, name and location of the projects in Cairo or Alex city.
5. Display the Projects full data of the projects with a name starts with "a" letter.
6. display all the employees in department 30 whose salary from 1000 to 2000 LE monthly
7. Retrieve the names of all employees in department 10 who works more than or equal10 hours per week on "AL ~~Rabwah~~" project.
8. Find the names of the employees who directly supervised with ~~Kamel~~ Mohamed.
9. Retrieve the names of all employees and the names of the projects they are working on, sorted by the project name.
10. For each project located in Cairo City , find the project number, the controlling department name ,the department manager last name ,address and birthdate.
11. Display All Data of the mangers
12. Display All Employees data and the data of their dependents even if they have no dependents

I

**Data Manipulating Language:**

1. Insert your personal data to the employee table as a new employee in department number 30, SSN = 102672, ~~Superssn~~ = 112233, salary=3000.
2. Insert another employee with personal data your friend as new employee in department number 30, SSN = 102660, but don't enter any value for salary or manager number to him.
3. Upgrade your salary by 20 % of its last value.

	<b>Order Total</b>				3,800.00

**Primary Key:** Sales Order Number,

# Sales Order

*Fiction Company  
202 N. Main  
Manhattan, KS 66502*

<b>CustomerNumber:</b>	<i>1001</i>	<b>Sales Order Number:</b>	<i>405</i>
<b>Customer Name:</b>	<i>ABC Company</i>	<b>Sales Order Date:</b>	<i>2/1/2000</i>
<b>Customer Address:</b>	<i>100 Points Manhattan, KS 66502</i>	<b>Clerk Number:</b>	<i>210</i>
		<b>Clerk Name:</b>	<i>Martin Lawrence</i>

Item Ordered	Description	Quantity	Unit Price	Total
800	<i>widgit small</i>	40	60.00	2,400.00
801	<i>tingimajigger</i>	20	20.00	400.00
805	<i>thingibob</i>	10	100.00	1,000.00
<b>Order Total</b>				<b>3,800.00</b>

Primary Key: Sales Order Number.

**CustomerNumber:** 1001  
**Customer Name:** ABC Company  
**Customer Address:** 100 Points  
Manhattan, KS 66502

**Sales Order Number:** 405  
**Sales Order Date:** 2/1/2000  
**Clerk Number:** 210  
**Clerk Name:** Martin Lawrence

Item Ordered	Description	Quantity	Unit Price	Total
800	widgit small	40	60.00	2,400.00
801	tingimajigger	20	20.00	400.00
805	thingibob	10	100.00	1,000.00
<b>Order Total</b>				3,800.00

**Primary Key:** Sales Order Number,

**CustomerNumber:** 1001  
**Customer Name:** ABC Company  
**Customer Address:** 100 Points  
Manhattan, KS 66502

**Sales Order Number:** 405  
**Sales Order Date:** 2/1/2000  
**Clerk Number:** 210  
**Clerk Name:** Martin Lawrence

Item Ordered	Description	Quantity	Unit Price	Total
800	widgit small	40	60.00	2,400.00
801	tingimajigger	20	20.00	400.00
805	thingibob	10	100.00	1,000.00
<b>Order Total</b>				3,800.00

**Primary Key:** Sales Order Number,

Repeated Cols

Repeating Groups

Eid,Ename,Eadd,Pro\_ID,Pro\_Name,PlocID,PlocCity,hours,Bouns

1st

Employee(Eid,ename,eadd)

Emp\_Pro(Eid,Pro\_id,Pro\_name,PlocID,PlocCity,hours,Bonus)

2nd

Employee (Eid,ename,eadd)

Project(Pro\_id,Pro\_name,PlocID,PLocCity)

Emp\_pro(Eid,Pro\_id,hours,bonus)

3rd

employee OK

Emp\_Pro ok

project(Pro\_id,Pro\_Name,Plocid)

locations(PLocID,PlocCity)

|Example2:

Unnormalized Table									
Dept #	Dept Name	Location	Mgr Name	Mgr ID No.	Tel Extn.	Cust #	Cust Name	Date of Complaint	Nature of Complaint
11232	Soap Division	Cincinnati	Mary Samuel	S11	7711	P10451	Robert Drumtree	12/01/1998	Poor Service
						P10480	Steven Parks	14/01/1998	Discourteous Attendant

The diagram illustrates the decomposition of a primary key in an unnormalized table. It features two horizontal arrows pointing upwards from the bottom of the table's rows to specific columns. The top arrow originates from the second row and points to the 'Cust #', 'Cust Name', 'Date of Complaint', and 'Nature of Complaint' columns. The bottom arrow originates from the fourth row and points to the 'Mgr Name', 'Mgr ID No.', 'Tel Extn.', and 'Cust #' columns. These four columns are highlighted with thick black borders, indicating they form a composite primary key.

هشتنغل و ارجاع على Documentation.

SQL Joins.

>>Saudi Arabia << عمل و عبادة

Make Normalization using ChatGPT.

Big Table



+201033332689



## Aggregate Function

Count , Max ,Min , Avg ,Sum

Select **Sum(Salary)** 62500  
From employee

Select **Min(Salary),Max(salary)**  
From Employee 1000 9000

Select **Count(eid)**  
from employee 15  
  
Count(\*) -----> 15  
Count(eid) -----> 15  
count(ename) -----> 14

Select **Avg(salary)**  
from employee

Select **Min(salary),did**  
from employee  
group by did

1000	10
2000	20
1500	30

Select **Count(eid),Address**  
from employee  
group by Address

6	Cairo
5	Alex
4	Mansoura

4 Select Count(eid),address  
1 from employee  
2 where did in(10,30)  
3 group by address

3	Cairo
3	alex
4	Mansoura

Select **Min(salary),did**  
from employee  
where address like '\_a%'  
group by did

2000	10
2000	20
1500	30

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	ala	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

Select Sum(salary), did  
from employee  
group by did  
having Sum(salary) >= 22000

28000 20

Select Count(eid), address  
from employee  
group by address  
having Count(eid) >= 5

6	cairo
5	alex

Select Sum(salary), did  
from employee  
group by did  
having Count(eid) > 5

20000 10

Select Min(salary), did  
from employee  
group by did

1000	10
2000	20
1500	30

Select Count(eid), Address  
from employee  
group by Address

6	Cairo
5	Alex
4	Mansoura

4 Select Count(eid), address  
1 from employee  
2 where did in(10,30)  
3 group by address

3	Cairo
3	alex
4	Mansoura

Select Min(salary), did  
from employee  
where address like '\_a%'  
group by did

2000	10
2000	20
1500	30

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	om	3000	mansoura	30

NULL

Select Sum(salary), did  
from employee  
group by did  
having Sum(salary) >= 22000

28000 20

Select Count(eid), address  
from employee  
group by address  
having Count(eid) >= 5

6	cairo
5	alex

Select Sum(salary), did  
from employee  
group by did  
having Count(eid) > 5

20000 10

Select Sum(salary), did  
from employee  
where address like '\_a%'  
group by did  
having sum(salary) > 12000

15000	20
14500	30

Select Max(salary), address  
from employee  
where did in(10,30)  
group by address  
having Count(eid) > 3

8000 mansoura

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



Select Sum(salary), did  
from employee  
group by did  
having Sum(salary) >= 22000

28000 20

Select Count(eid), address  
from employee  
group by address  
having Count(eid) >= 5

6	cairo
5	alex

Select Sum(salary), did  
from employee  
group by did  
having Count(eid) > 5

20000 10

Select Sum(salary), did  
from employee  
where address like '\_a%'  
group by did  
having sum(salary) > 12000

15000	20
14500	30

Select Max(salary), address  
from employee  
where did in(10,30)  
group by address  
having Count(eid) > 3

8000 mansou a

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	on/ <del>re</del>	3000	mansoura	30

NULL



```
□ select salary  
      from Instructor
```

```
□ select Sum(salary)  
      from Instructor
```

```
□ select Min(Salary),Max(Salary)  
      from Instructor
```

```
select Min(Salary) as Min_Val,Max(Salary) as Max_val  
from Instructor  
  
select count(*),count(st_id),count(st_lname),count(st_age)  
from Student
```

```
[-] select count(*),count(st_id),count(st_lname),count(st_age)
      from Student

[-] select avg(st_age)
      from Student

[-] select sum(st_age)/count(*)
      from Student
```

```
from Student
```

```
    select avg(st_age)
```

```
from Student
```

```
    select avg(isnull(st_age, 0))
```

```
from Student
```

```
    select sum(st_age)/count(*)
```

```
from Student
```

```
select avg(st_age)
```

```
from Student
```

```
select avg(isnull(st_age,0))
```

```
from Student
```

```
select sum(st_age)/count(*)
```

```
select sum(salary),dept_id  
from Instructor  
group by dept_id
```

```
select sum(salary),d.dept_id,dept_name  
from Instructor i inner join Department d  
on d.Dept_Id=i.Dept_Id  
group by d.dept_id,dept_name
```

```
select avg(st_age),st_address,dept_id  
from Student  
group by st_address,dept_id
```

```
select sum(salary),dept_id
from Instructor
group by dept_id

select sum(salary),d.dept_id,dept_name
from Instructor i inner join Department d
on d.Dept_Id=i.Dept_Id
group by d.dept_id,dept_name

select avg(st_age),st_address,dept_id
from Student
group by st_address,dept_id
```

column Dept\_Id(int, null)

```
select sum(salary),dept_id  
from Instructor  
group by dept_id
```

```
select sum(salary),dept_id  
from Instructor  
where salary>1000  
group by dept_id
```

```
select sum(salary),dept_id  
from Instructor  
group by dept_id
```

```
select sum(salary),dept_id  
from Instructor  
group by dept_id  
having sum(salary)>100000
```

```
from Instructor  
group by dept_id  
having sum(salary)>100000
```

```
select sum(salary),dept_id  
from Instructor  
group by dept_id  
having Count(ins_id)<6
```

```
from Instructor  
group by dept_id  
having Count(ins_id)<6
```

```
select sum(salary),dept_id  
from Instructor  
where salary>1000  
group by dept_id  
having Count(ins_id)<6
```

Select Sum(salary), did  
from employee  
group by did  
having Sum(salary) >= 22000

28000 20

Select Count(eid), address  
from employee  
group by address  
having Count(eid) >= 5

6	cairo
5	alex

Select Sum(salary), did  
from employee  
group by did  
having Count(eid) > 5

20000 10

Select Sum(salary), did  
from employee  
where address like '\_a%'  
group by did  
having sum(salary) > 12000

15000	20
14500	30

Select Max(salary), address  
from employee  
where did in(10,30)  
group by address  
having Count(eid) > 3

8000 mansoura

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

```
group by dept_id  
having Count(ins_id)<6
```

```
-- Subqueries
```

```
select *  
from Student  
where st_age<23
```

```
- [+] select *
  from Student
  where st_age < (select avg(st_age) from student)

- [+] select *,(select count(st_id) from Student)
  from student

- [+] select dept_name
  from Department
  from dept_id in (sele
```

```
- select dept_name  
  from Department  
 where dept_id in (select distinct(dept_id)  
                      from Student  
                     where dept_id is not null)
```

```
select dept_name  
from Department  
where dept_id not in (select distinct(dept_id)  
                      from Student  
                      where dept_id is not null)
```

```
select dept_name  
from Department  
where dept_id in (select distinct(dept_id)  
                   from Student  
                   where dept_id is not null)
```

```
from Student s inner join Department d  
on d.Dept_Id =s.Dept_Id
```

--Subquery + DML

```
- delete from Stud_Course  
where st_id=1
```

```
from Student s inner join Department d
    on d.Dept_Id =s.Dept_Id

--Subquery + DML
delete from Stud_Course
where st_id=1

delete from Stud_Course
where st_id=(select st_id from Student where St_Address='cairo')
```

```
from Student s inner join Department d
  on d.Dept_Id =s.Dept_Id

--Subquery  +  DML
delete from Stud_Course
where st_id=1

delete from Stud_Course
where st_id in (select st_id from Student where St_Address='cairo')
```

```
    on d.Dept_Id =s.Dept_Id
```

```
--Subquery + DML
```

```
delete from Stud_Course  
where st_id=1
```

```
delete from Stud_Course  
where st_id in (select st_id from Student where St_Address='cairo')
```

```
--union family  
union all      union      intersect      except
```

```
select st_fname  
from Student
```

```
select ins_name  
from Instructor
```

## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various databases and tables. The central Query Editor window contains the following SQL code:

```
--union family
union all    union      intersect   except

select st_fname
from Student
union all
select ins_name
from Instructor
```

The Results pane below displays the output of the query:

st_fname
Ahmed
Amr
Mona
Ahmed
NULL
Heba
Ali
Mohamed
Saly
NULL
Mawya

At the bottom of the screen, there is a red status bar with various icons and text, including "Activate Windows" and "Go to Settings to activate Windows".

```
select ins_name,ins_id  
from Instructor
```

```
- select st_id  
from Student
```

```
union all
```

```
select ins_name  
from Instructor
```

```
select ins_name  
from Instructor
```

```
B select st_fname  
from Student
```

```
union
```

```
select ins_name  
from Instructor
```

```
select ins_name  
from Instructor
```

```
select st_fname  
from Student  
union  
select ins_name  
from Instructor
```

```
--union family
```

```
union all      union    intersect    except
```

```
select st_fname as names
```

```
from Student
```

```
union all
```

```
select ins_name
```

```
from Instructor
```

from Instructor

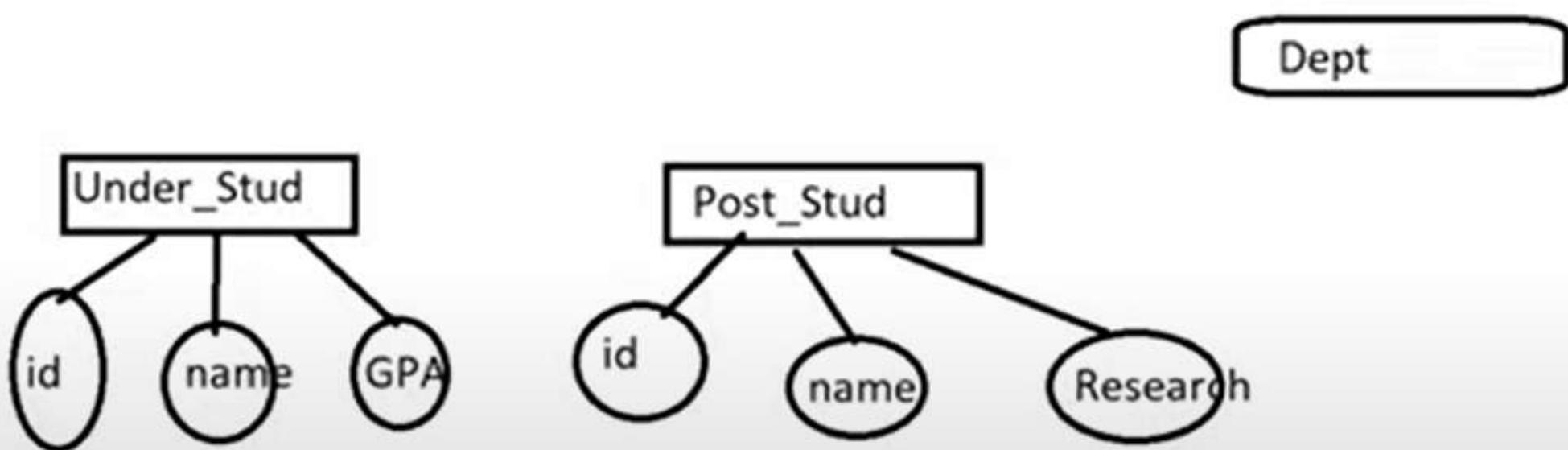
```
select st_fname  
from Student  
except  
select ins_name  
from Instructor
```

```
select st_fname,st_id  
from Student  
intersect  
select ins_name,ins_id  
from Instructor
```

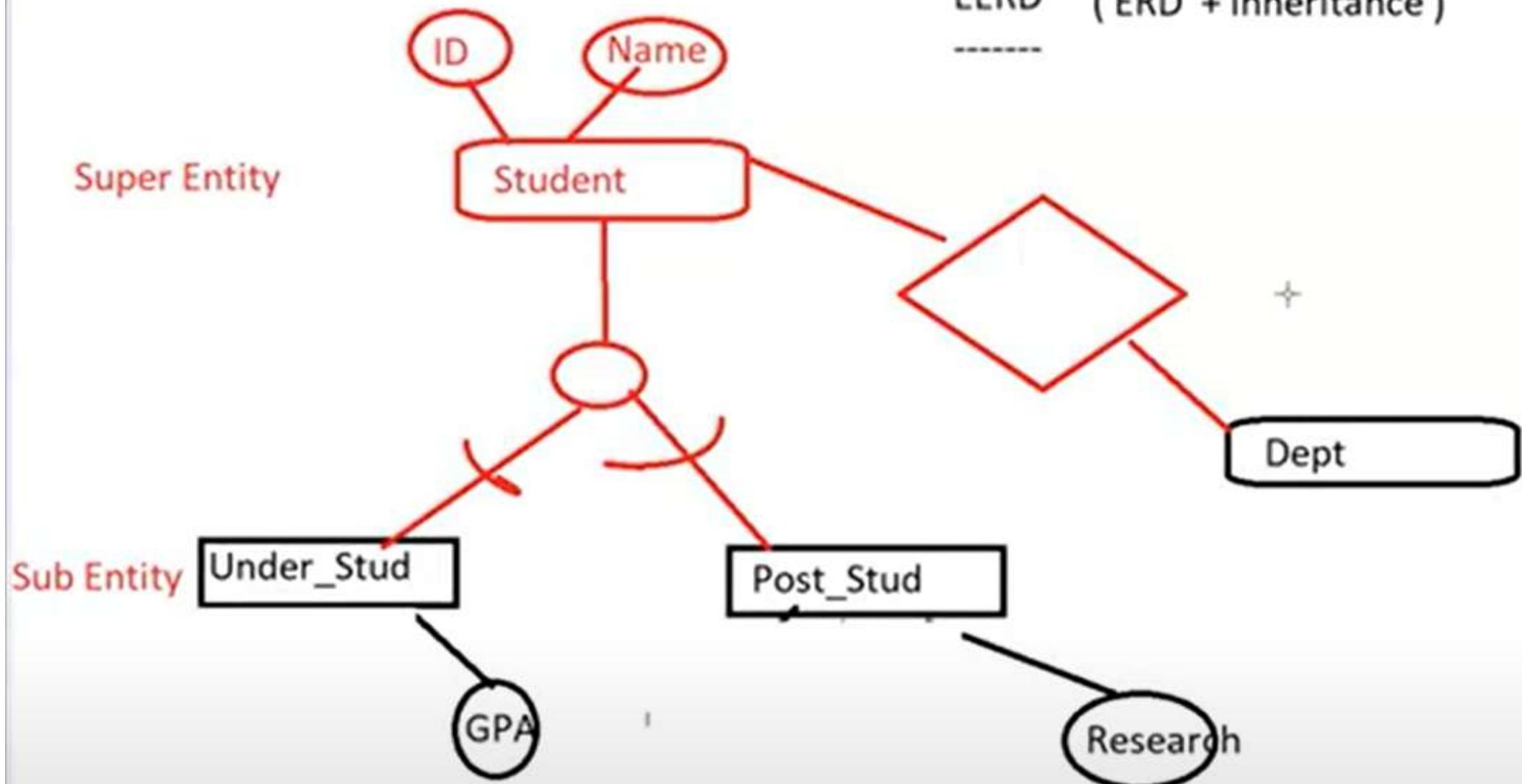
```
select st_fname,st_id  
from Student  
intersect  
select ins_name,ins_id  
from Instructor
```

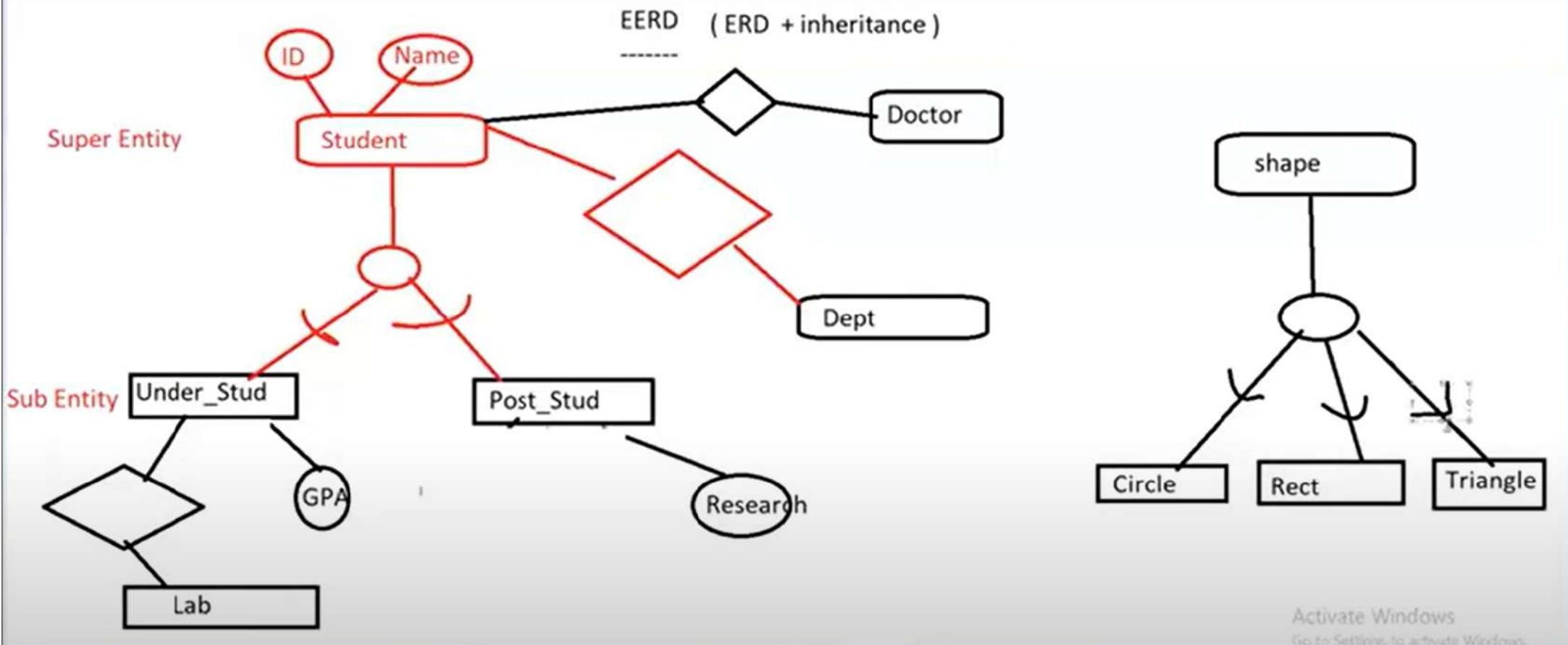
EERD ( ERD + inheritance )

-----



EERD ( ERD + inheritance )





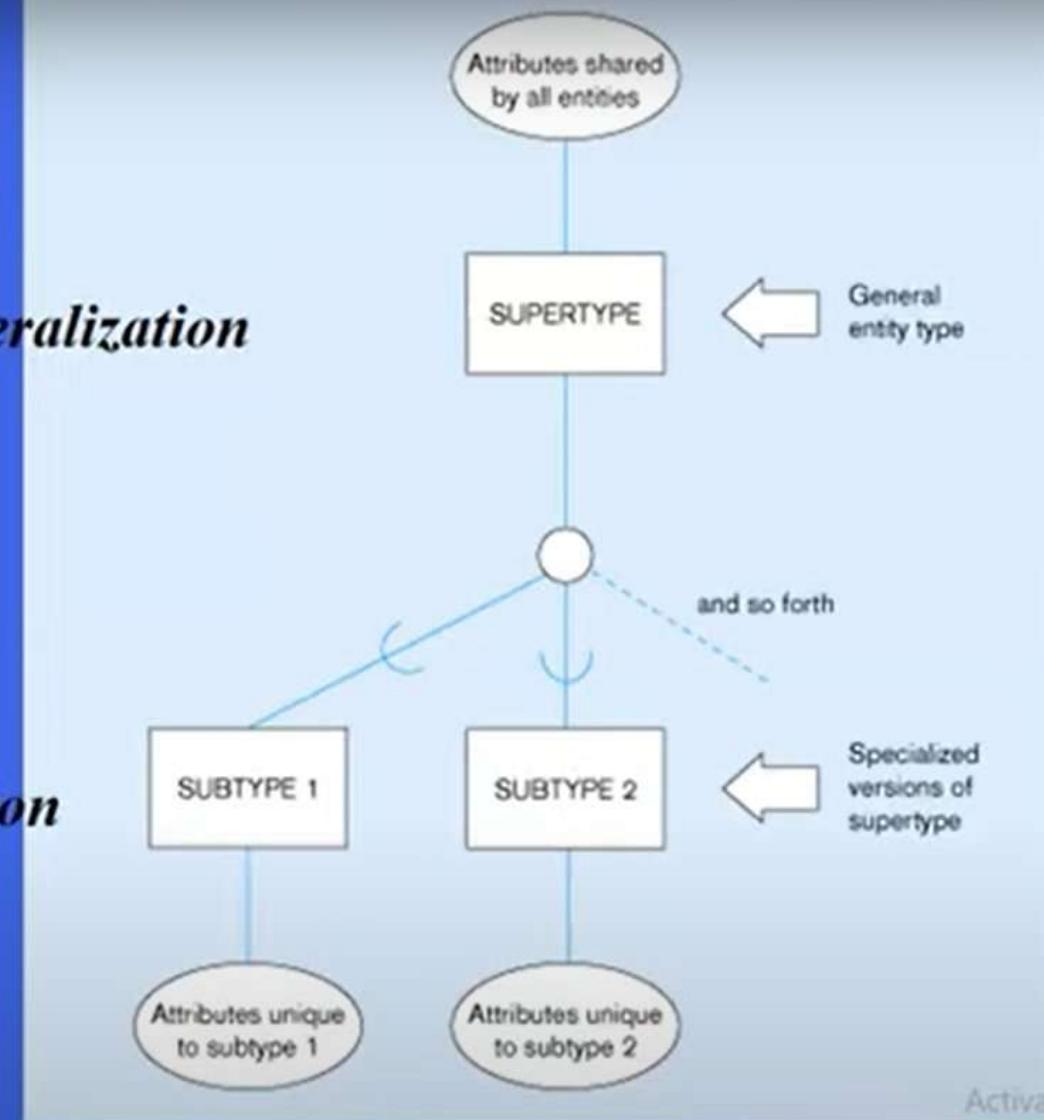
# Supertypes and Subtypes

- ↪ **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

## Basic notation for supertype/subtype relationships

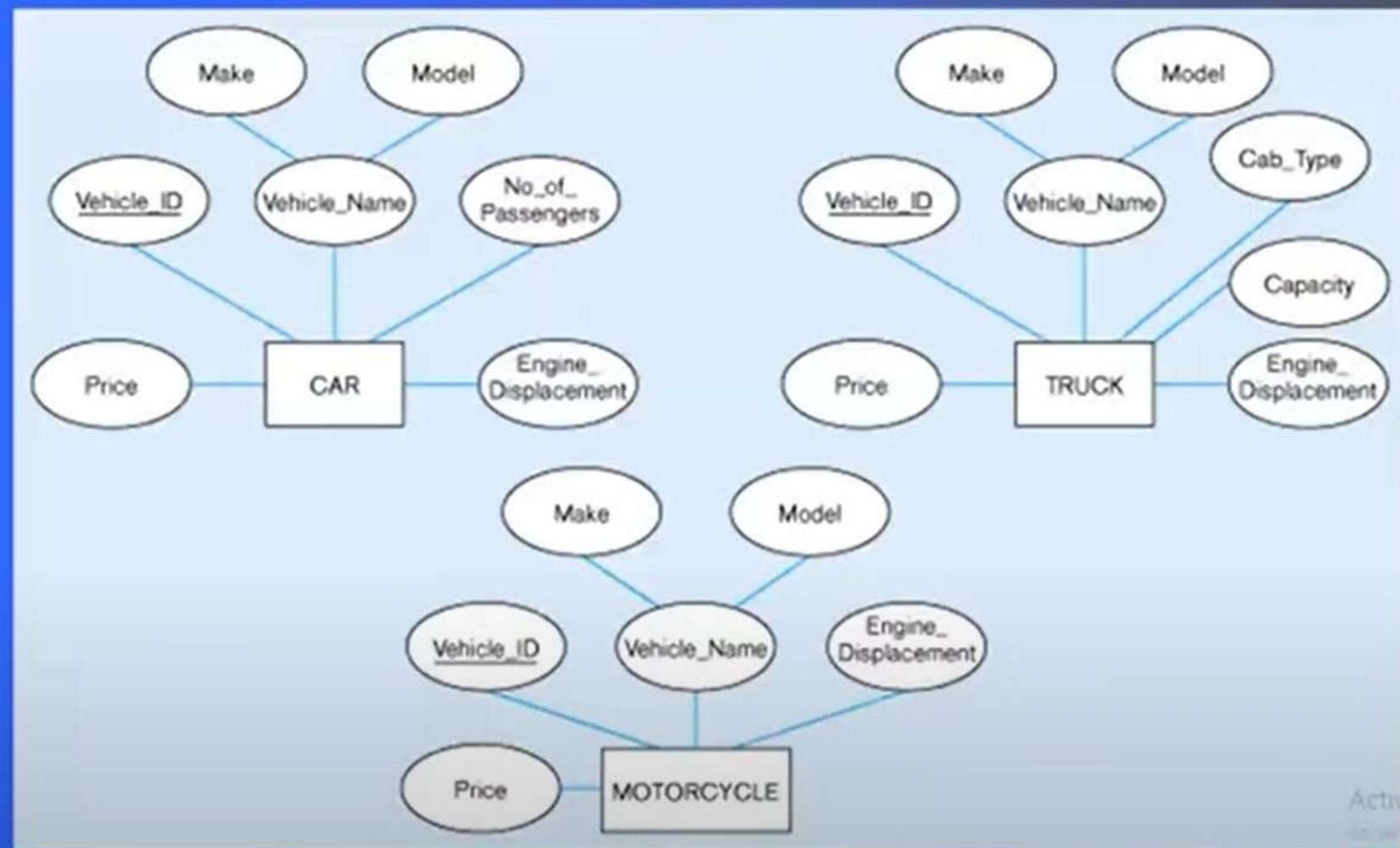
### *Generalization*

### *Specialization*



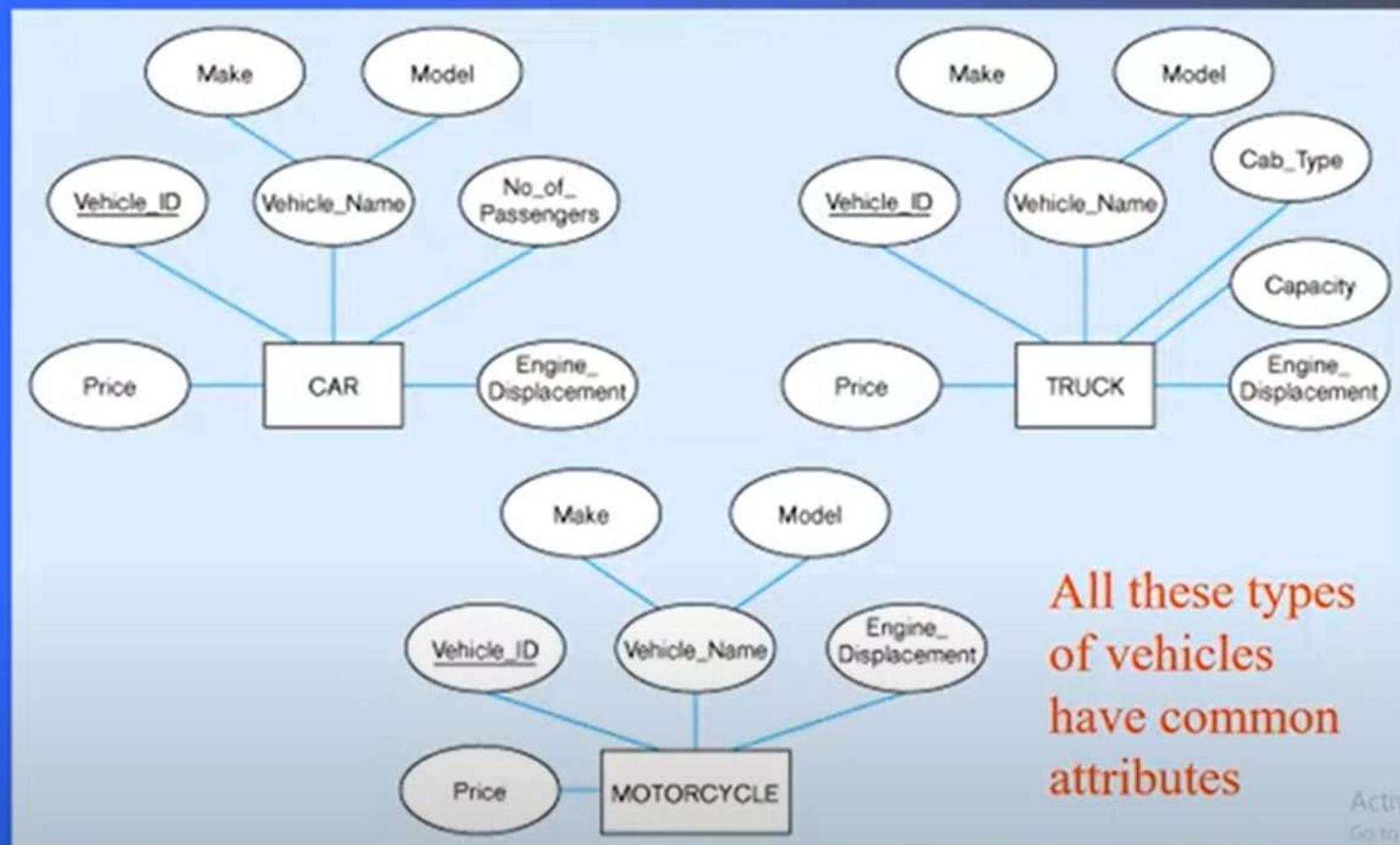
## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

### (a) Three entity types: CAR, TRUCK, and MOTORCYCLE



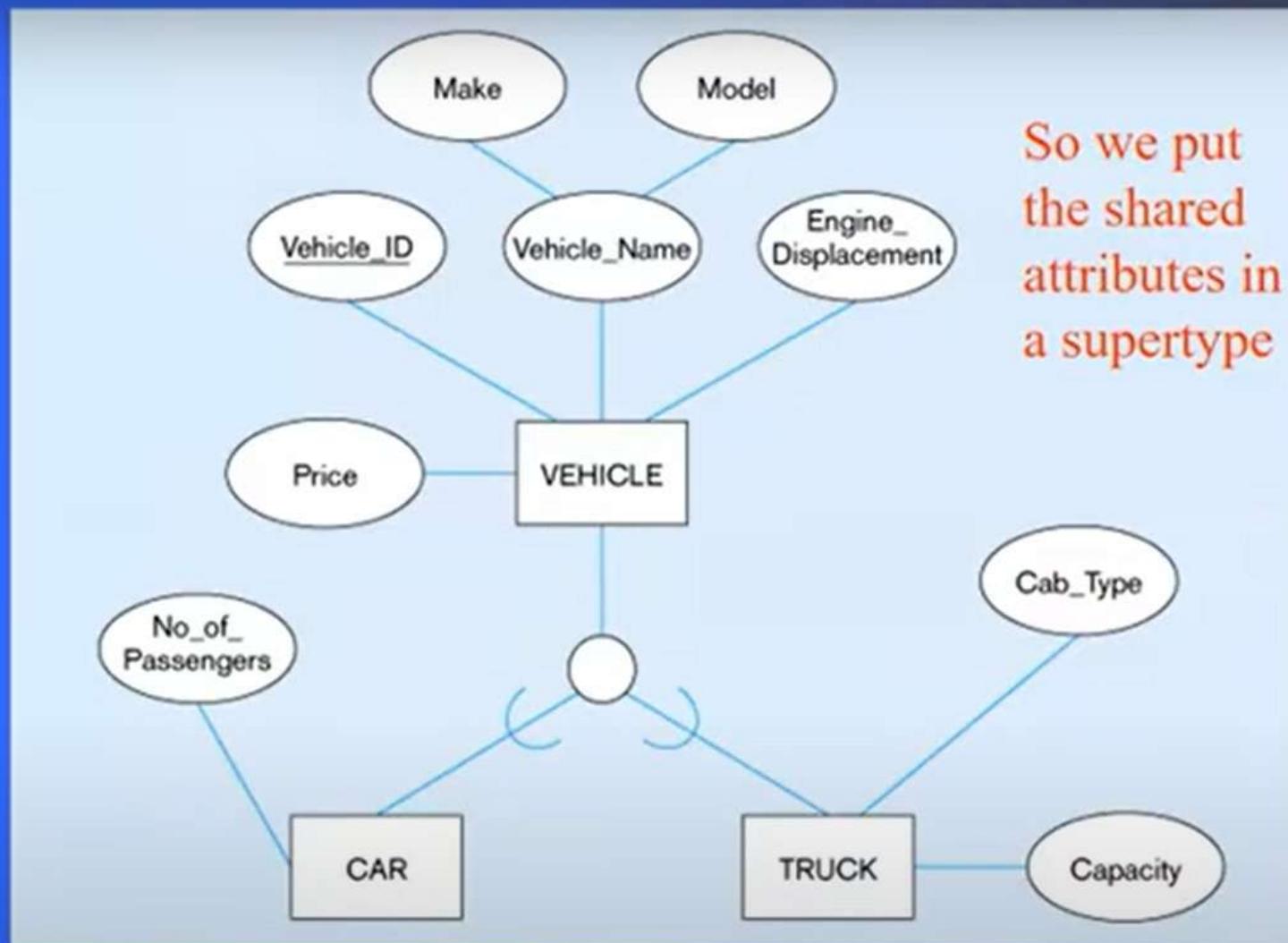
## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

### (a) Three entity types: CAR, TRUCK, and MOTORCYCLE



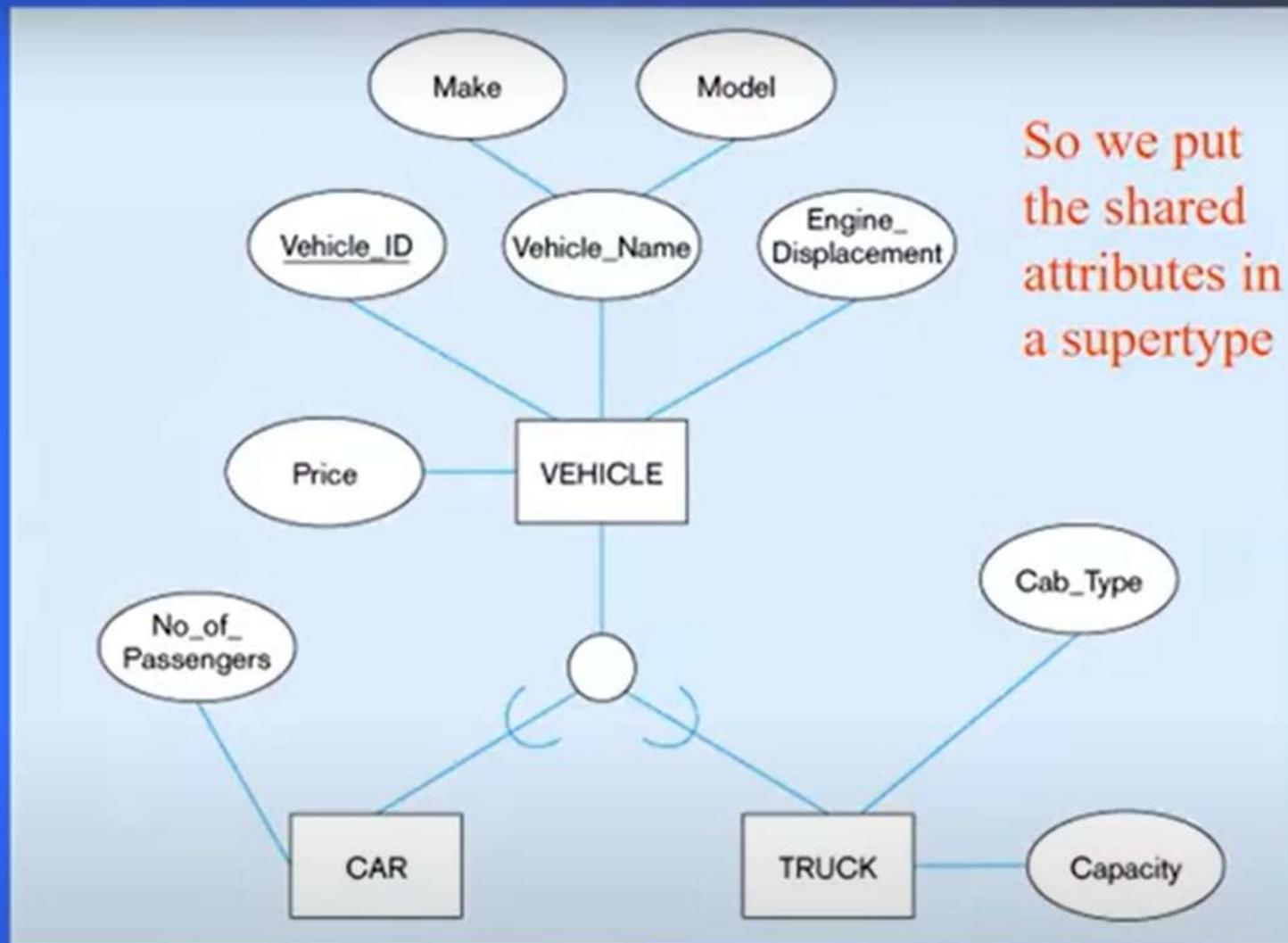
## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

### Generalization to VEHICLE supertype



Note: no subtype for motorcycle, since it has no unique attributes

## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

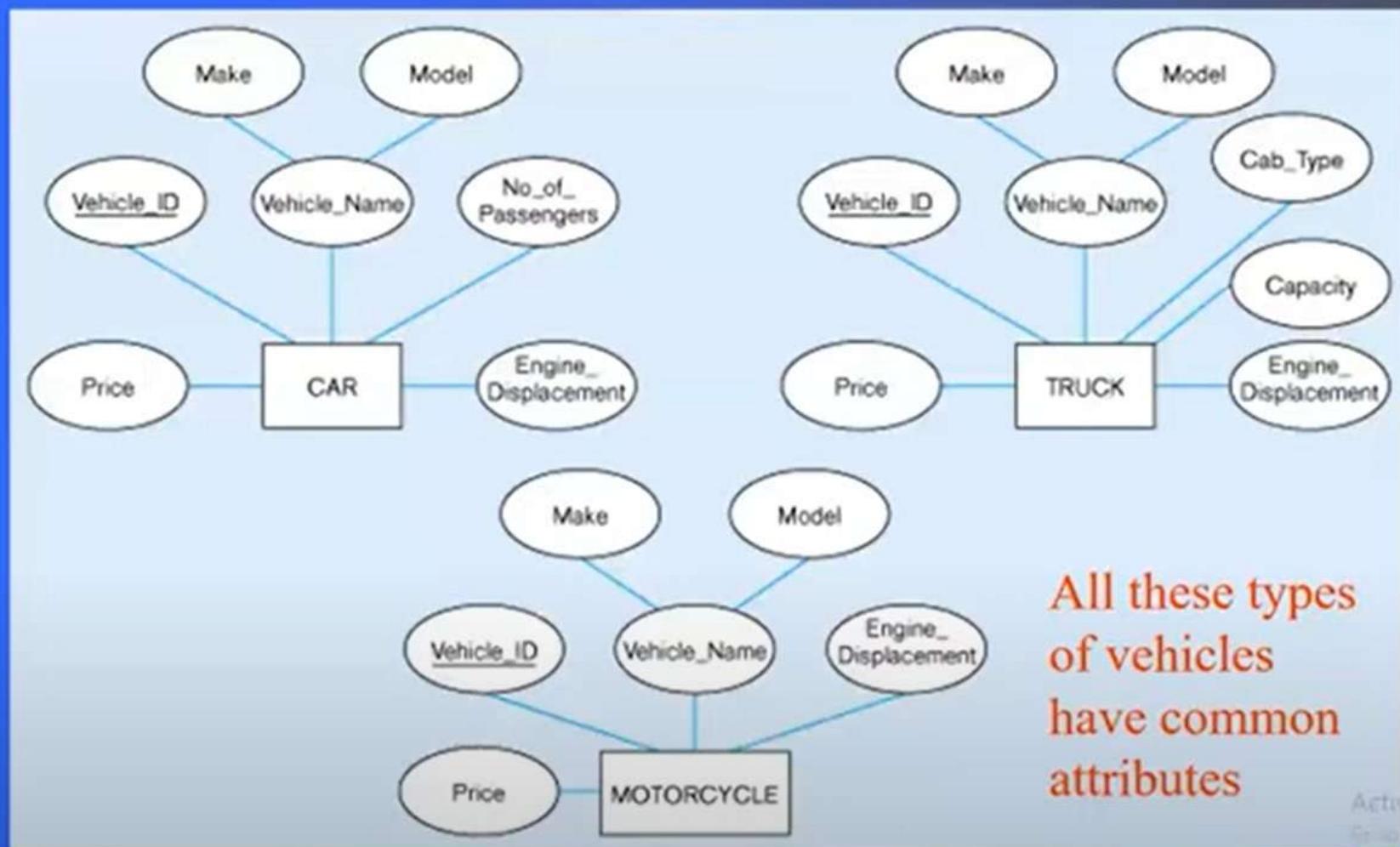


Note: no subtype for motorcycle, since it has no unique attributes

## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

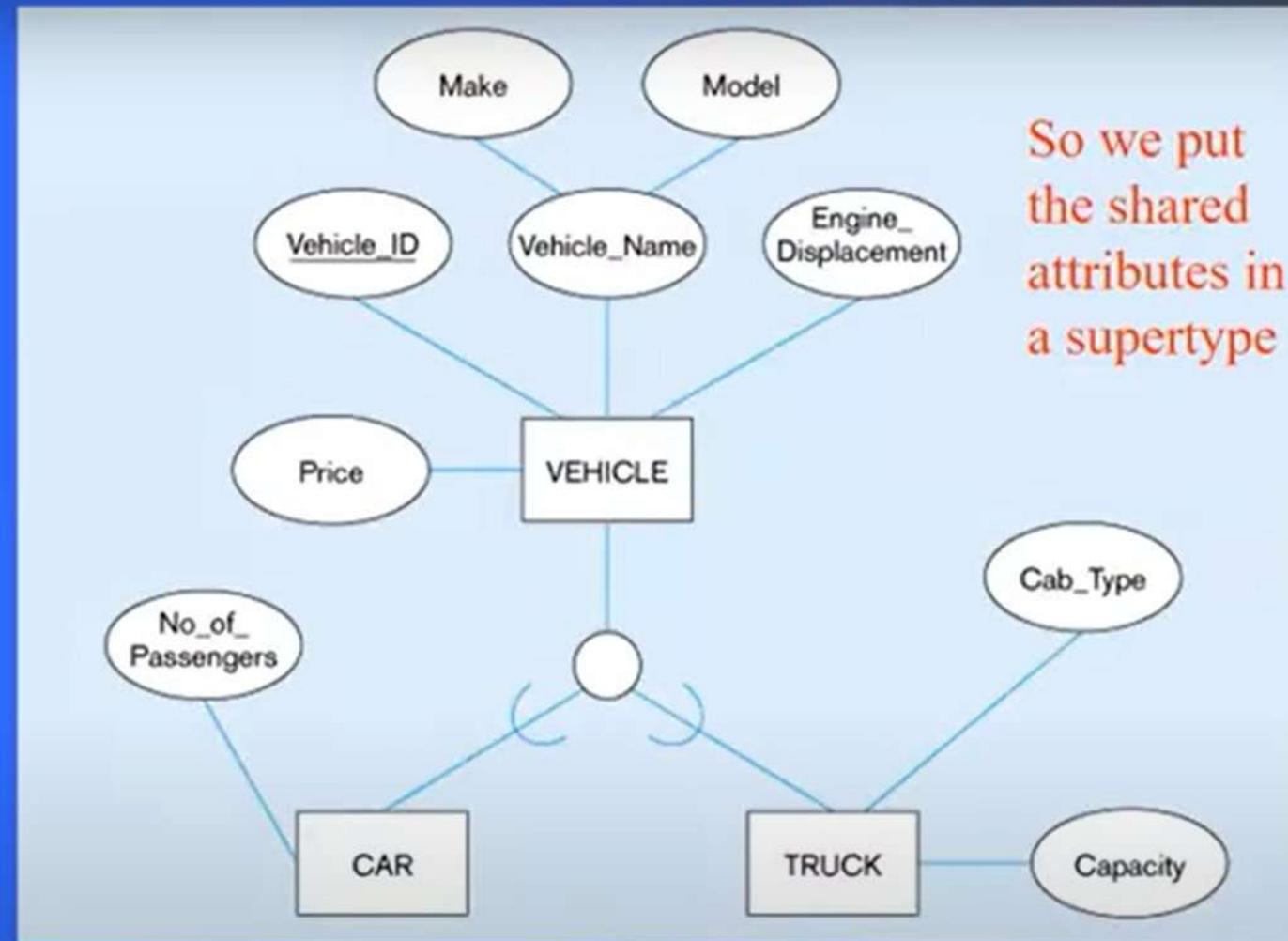
### Example of generalization

#### (a) Three entity types: CAR, TRUCK, and MOTORCYCLE



⇒ 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

Generalization to VEHICLE supertype

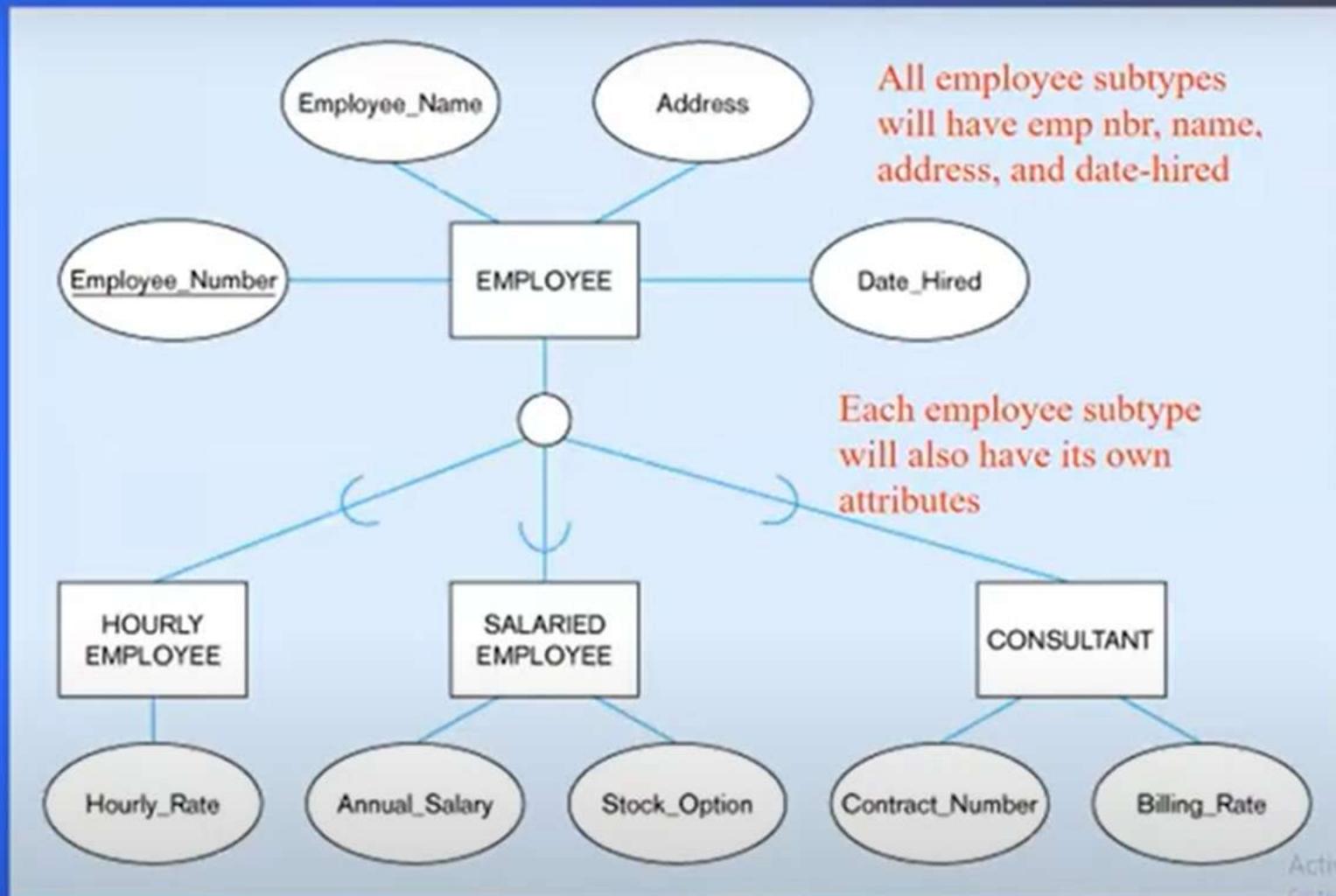


So we put  
the shared  
attributes in  
a supertype

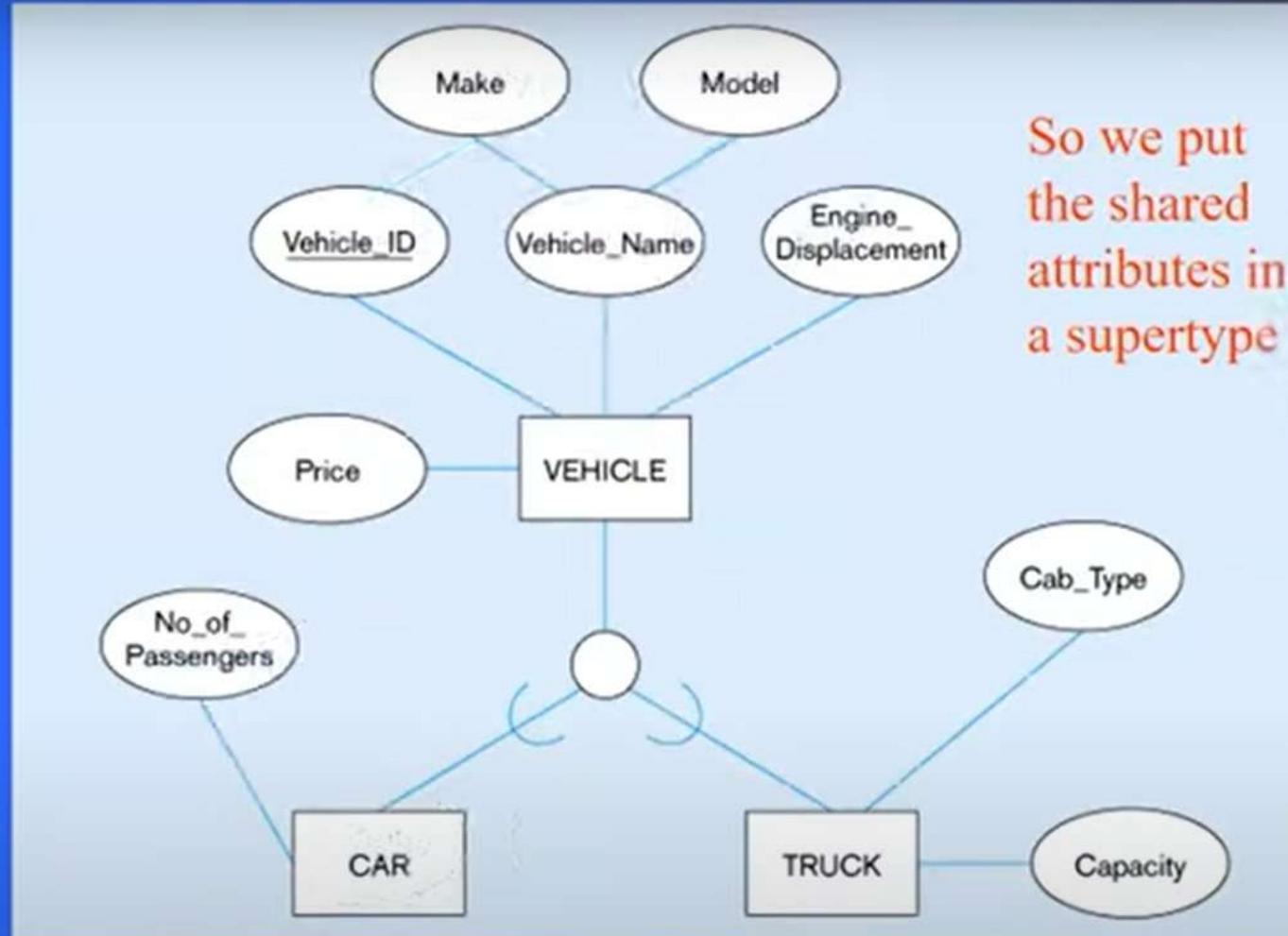
Note: no subtype for motorcycle, since it has no unique attributes

4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

## Employee supertype with three subtypes



## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

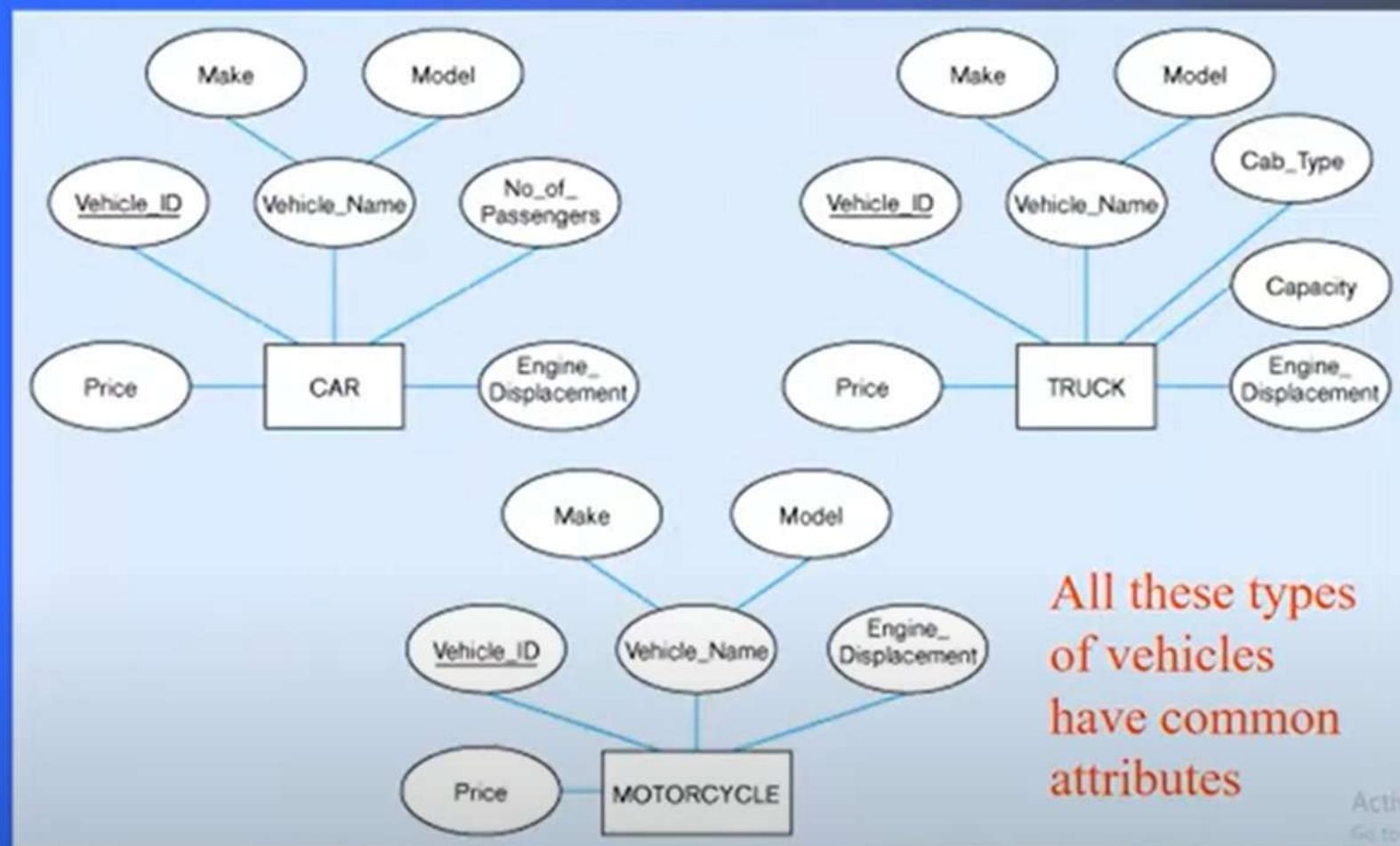


Note: no subtype for motorcycle, since it has no unique attributes

## Example of generalization

4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

(a) Three entity types: CAR, TRUCK, and MOTORCYCLE



- 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

## Employee supertype with three subtypes



4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

## Employee supertype with three subtypes



# 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

unction, Grouping, Union, Subqueries, EERD

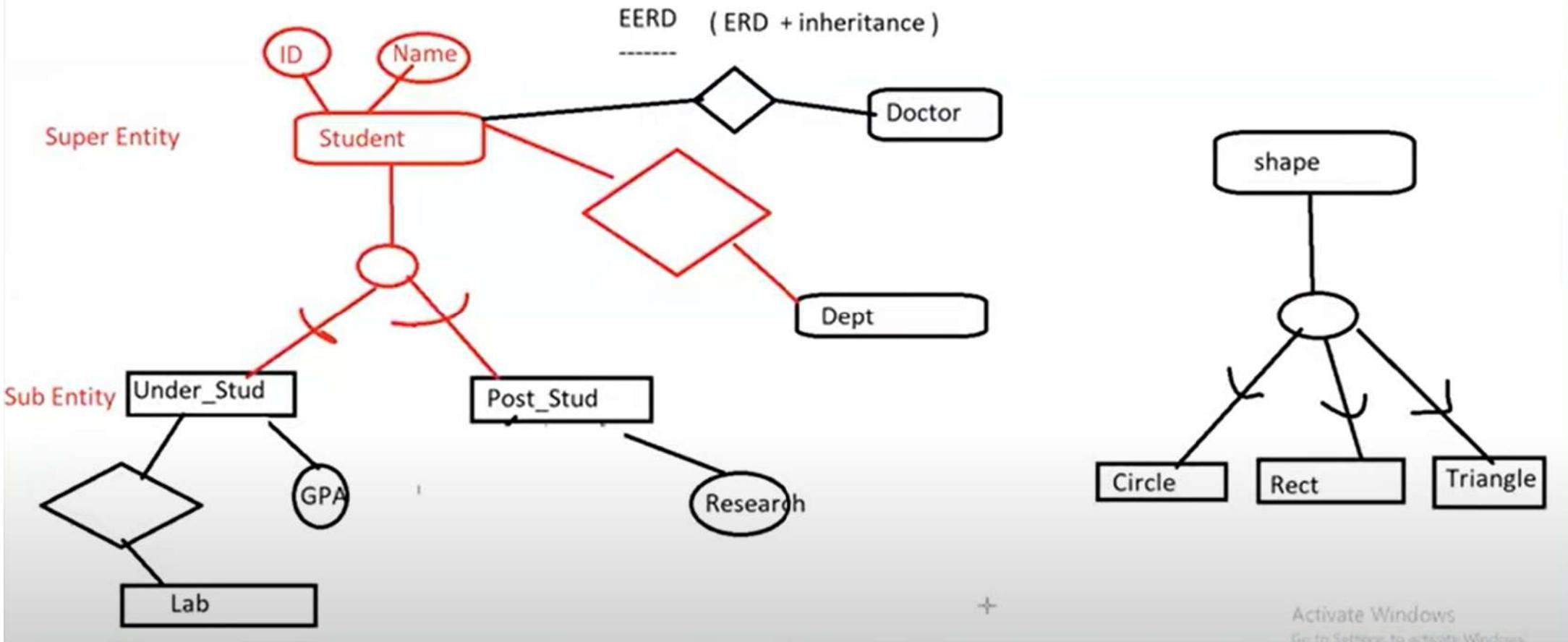
## Employee supertype with three subtypes

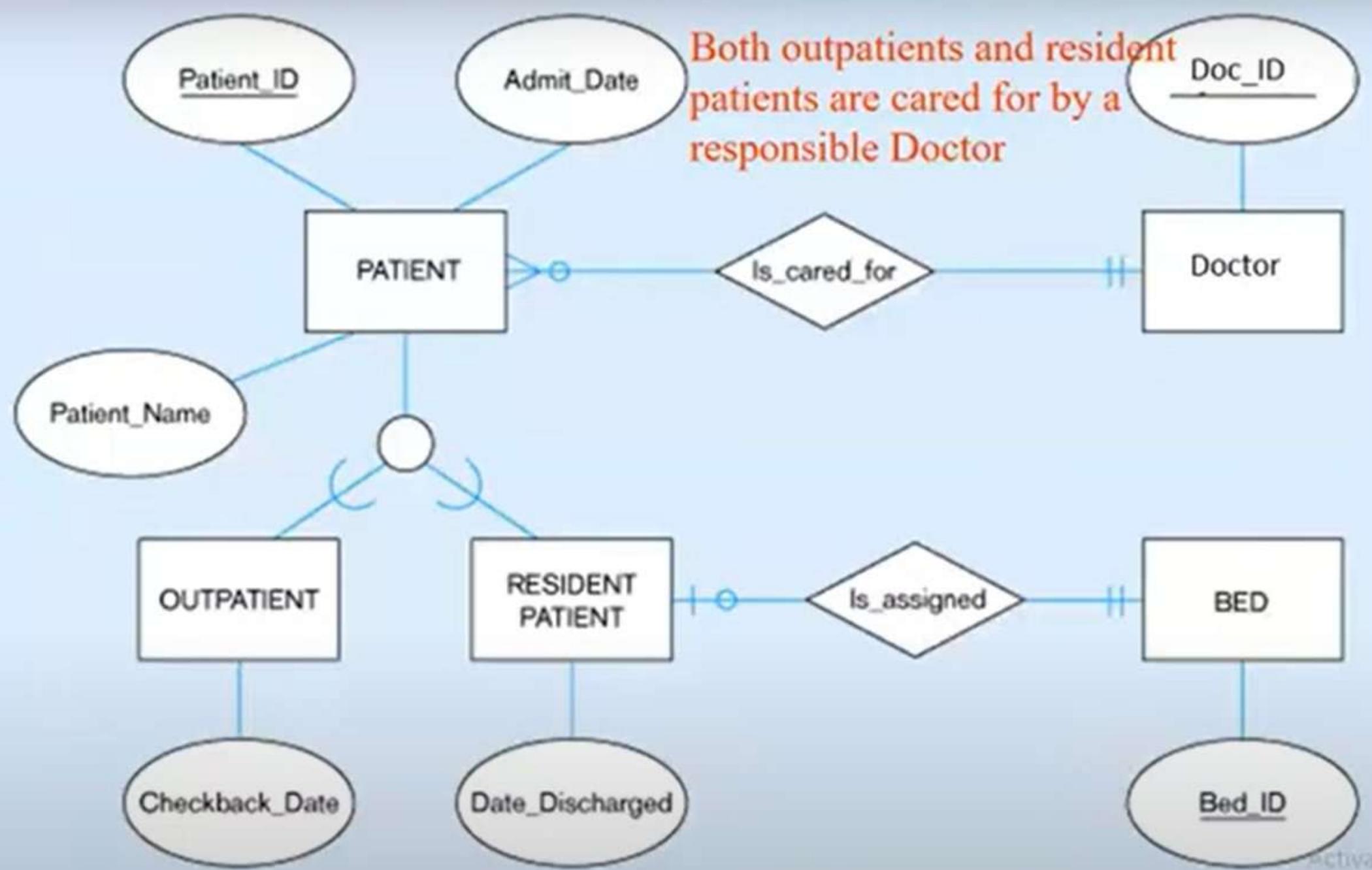


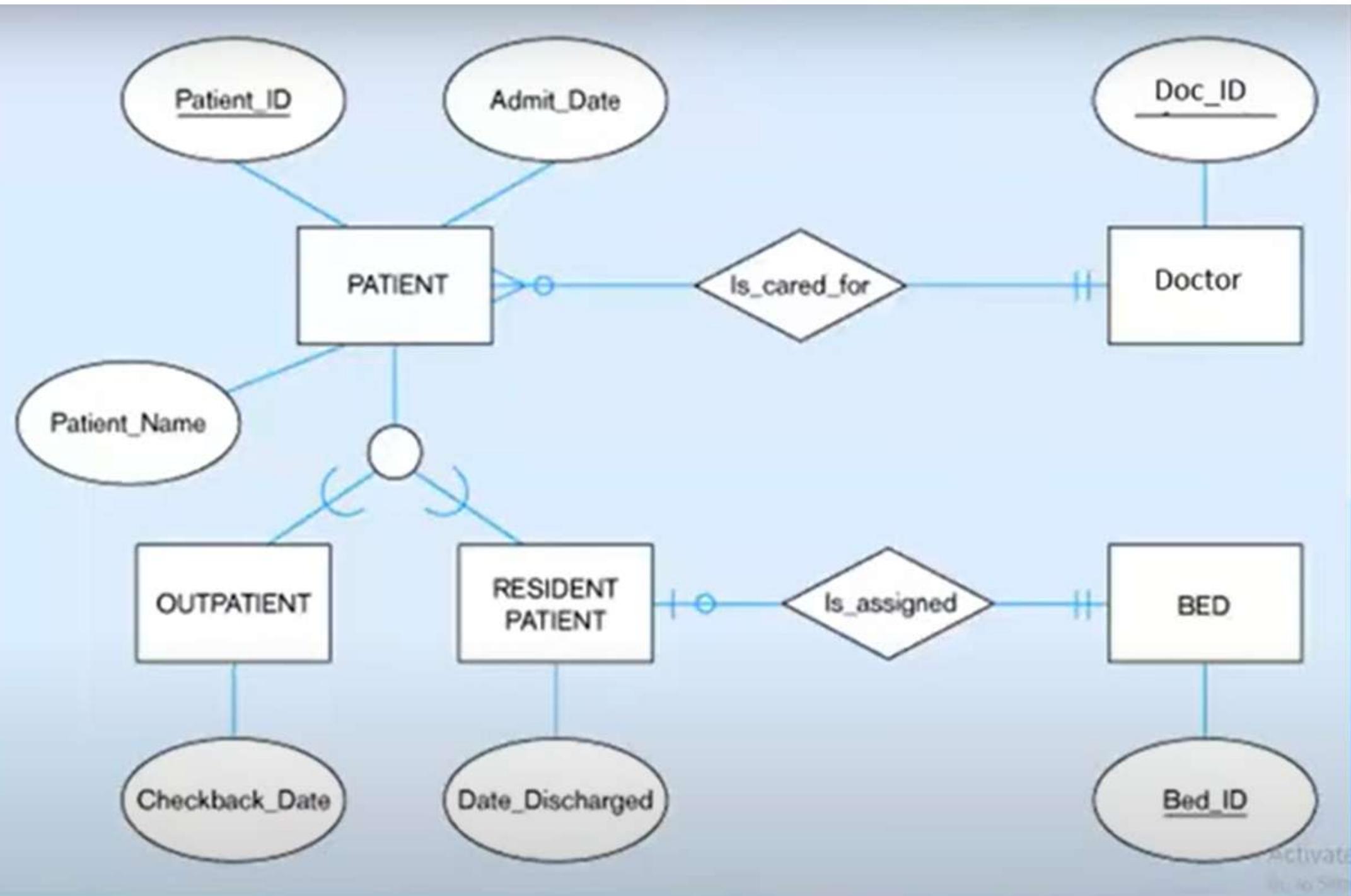
# Relationships and Subtypes

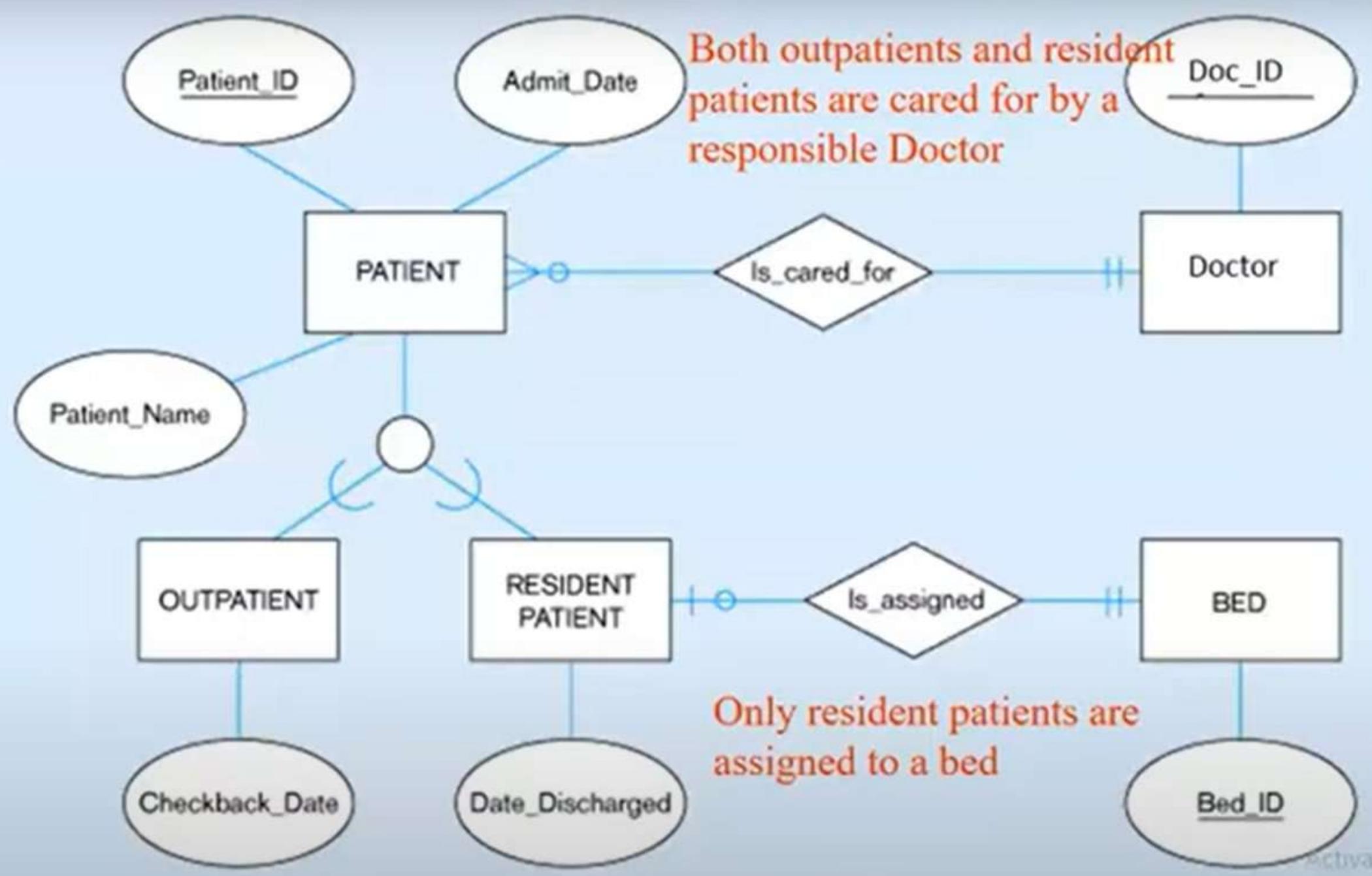
Relationships at the *supertype* level indicate that all subtypes will participate in the relationship

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









# Constraints in Supertype

## Completeness Constraints:

- Total Specialization Rule: Yes (double line)
- Partial Specialization Rule: No (single line)

## Disjointness Constraints:

- Total Disjoint (d)
- Overlap Rule (o)

# Constraints in Supertype

## Completeness Constraints:

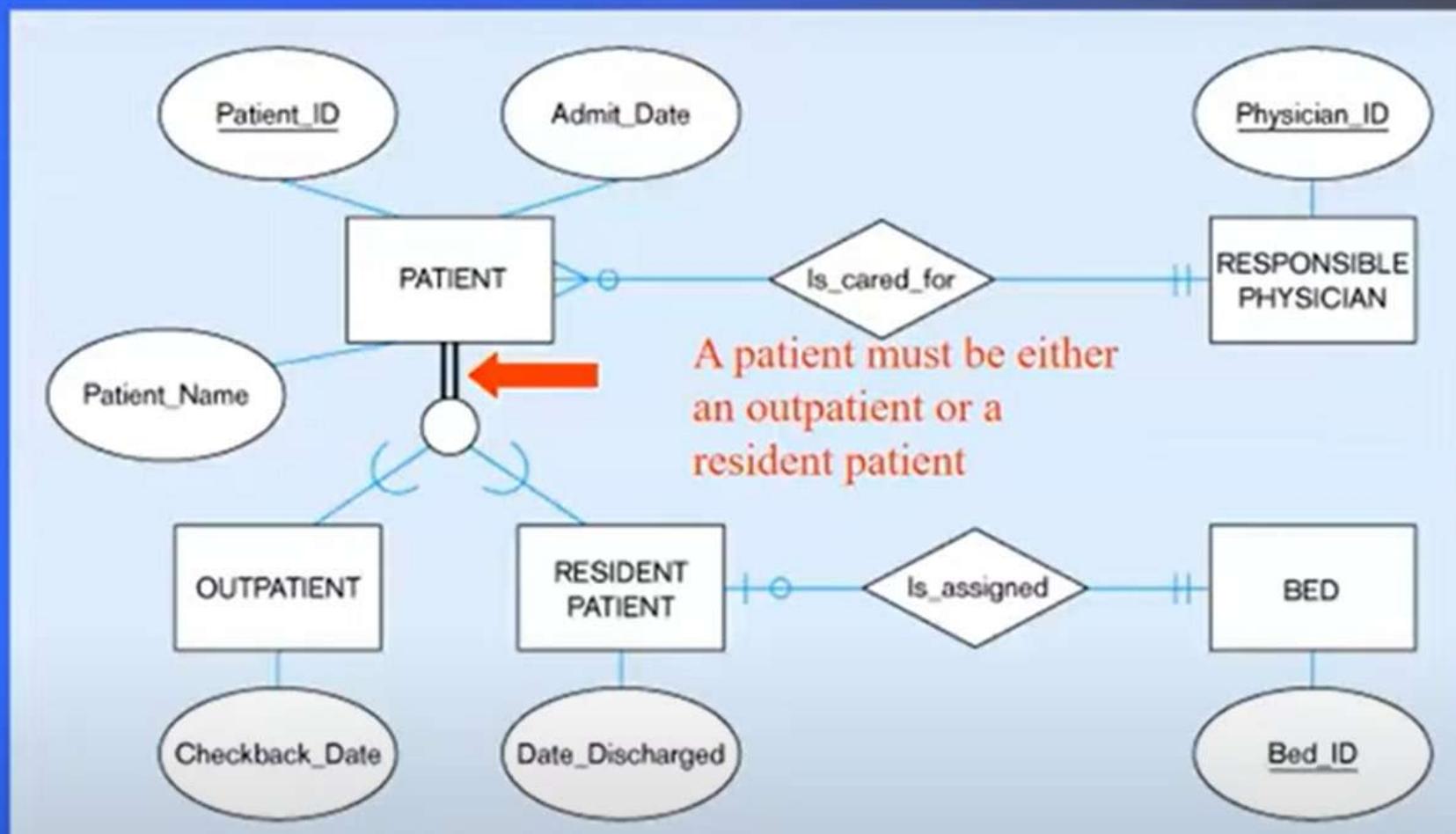
- Total Specialization Rule: Yes (double line)
- Partial Specialization Rule: No (single line)

## Disjointness Constraints:

- Total Disjoint (d)
- Overlap Rule (o)

## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

### Total specialization rule



# Constraints in Supertype

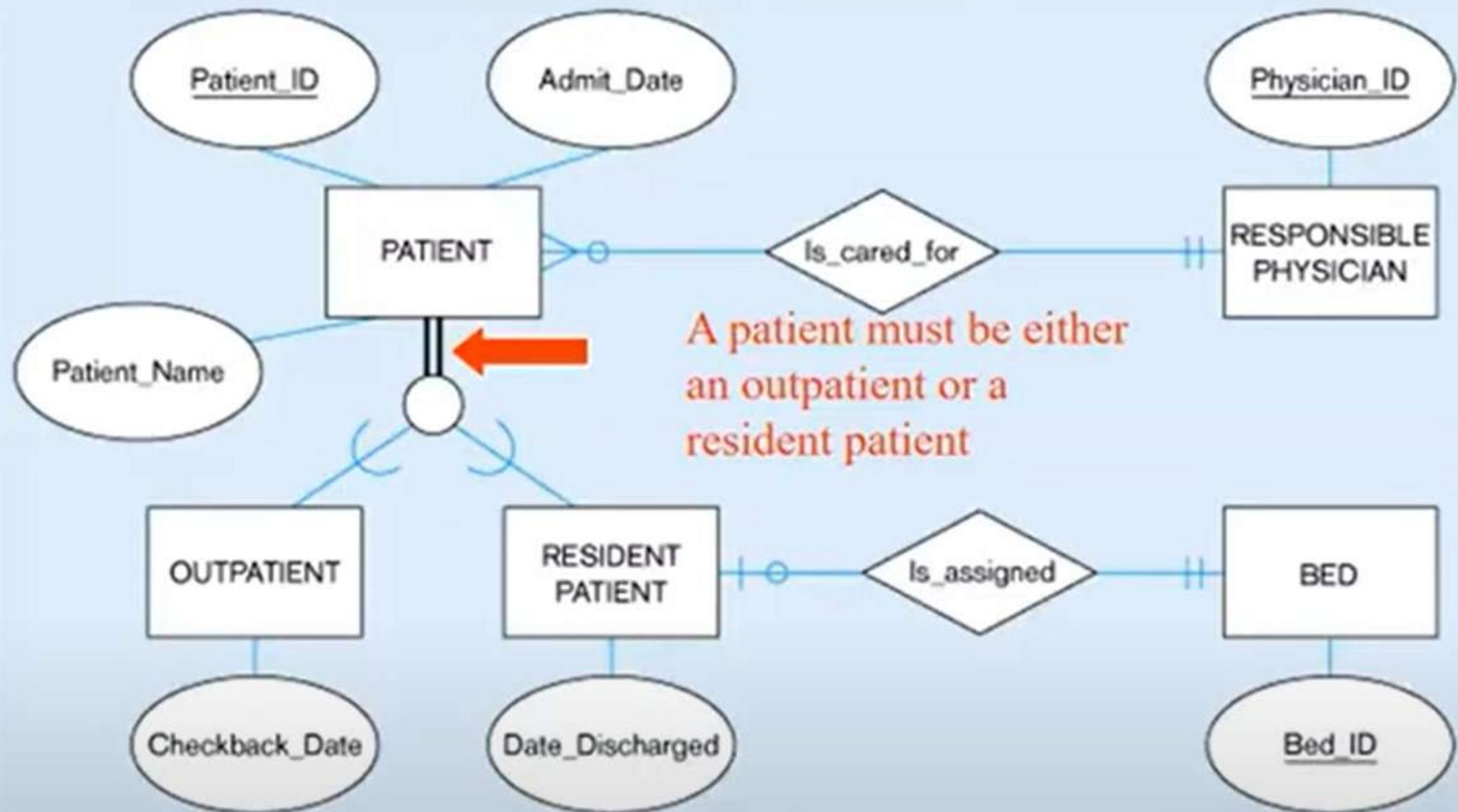
## Completeness Constraints:

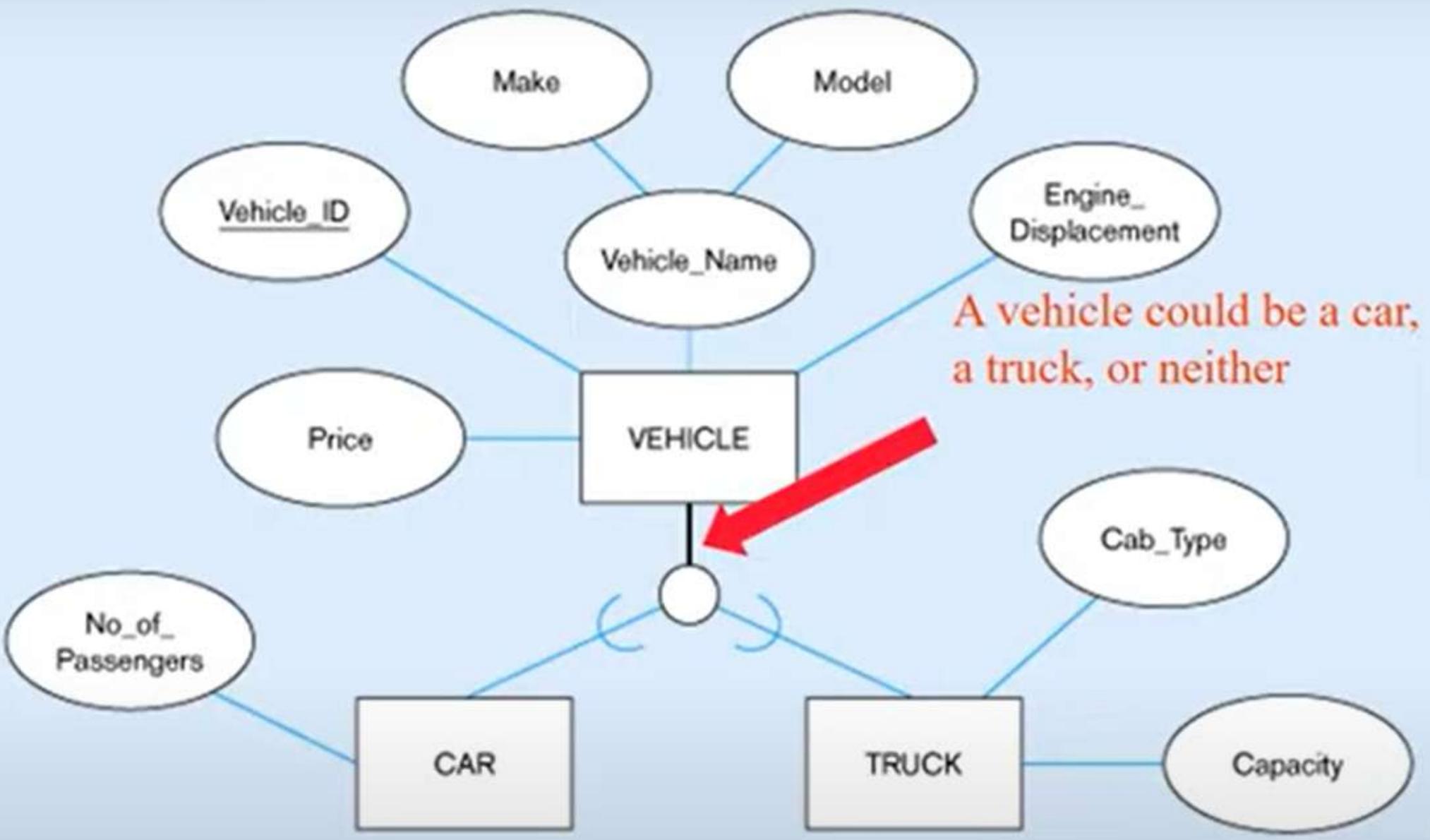
- Total Specialization Rule: Yes (double line)
- Partial Specialization Rule: No (single line)

## Disjointness Constraints:

- Total Disjoint (d)
- Overlap Rule (o)

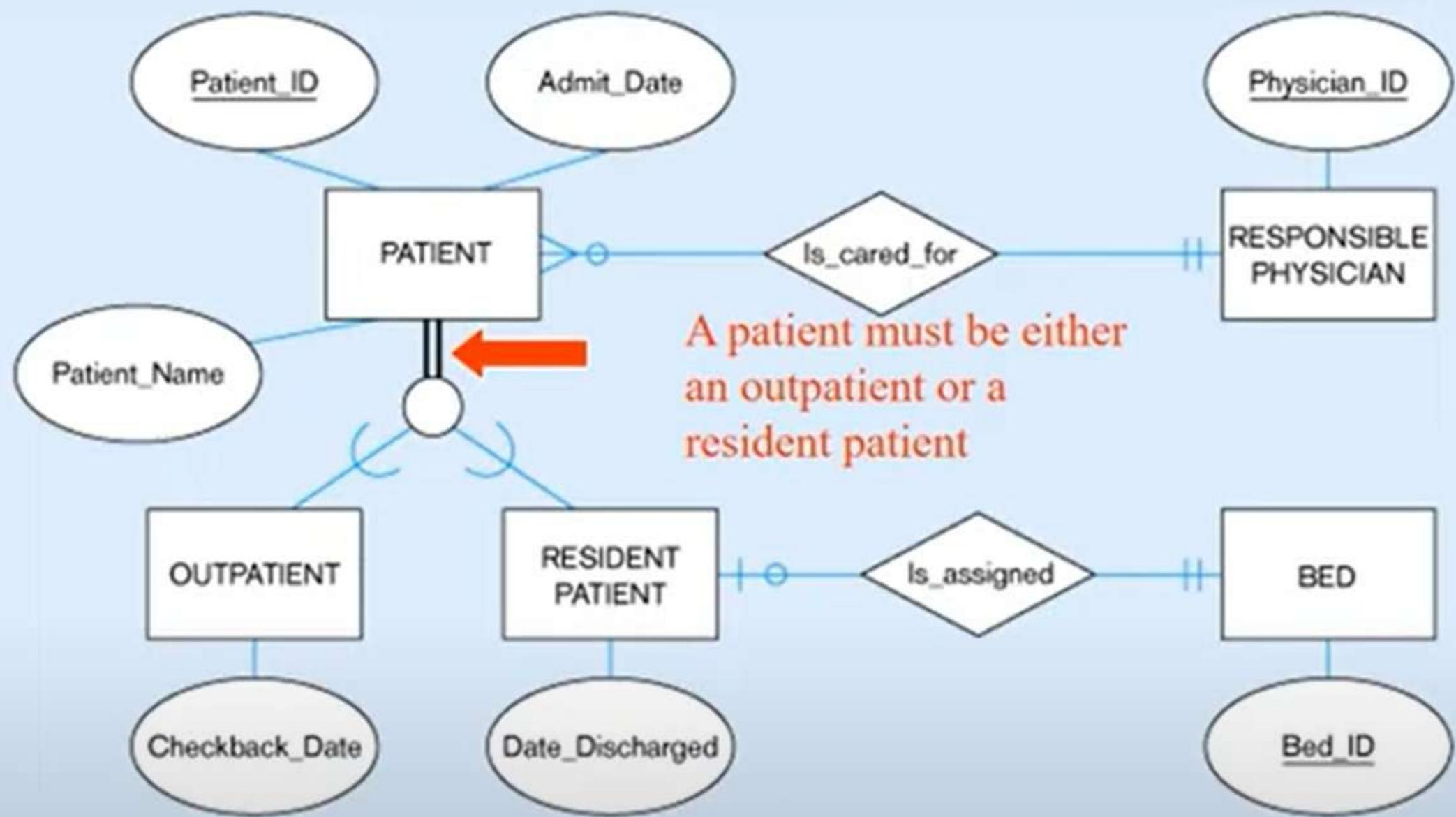
## Total specialization rule

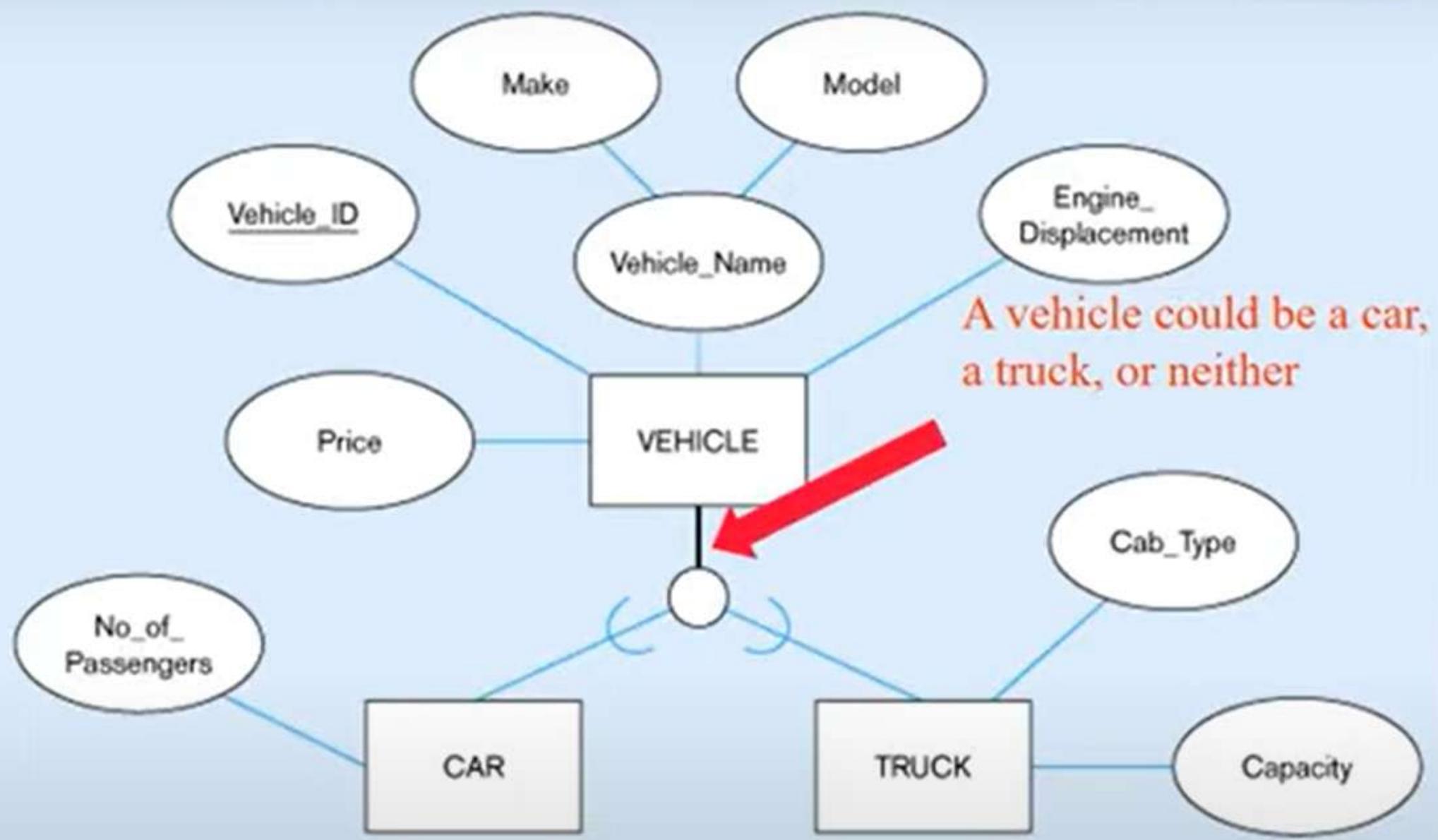




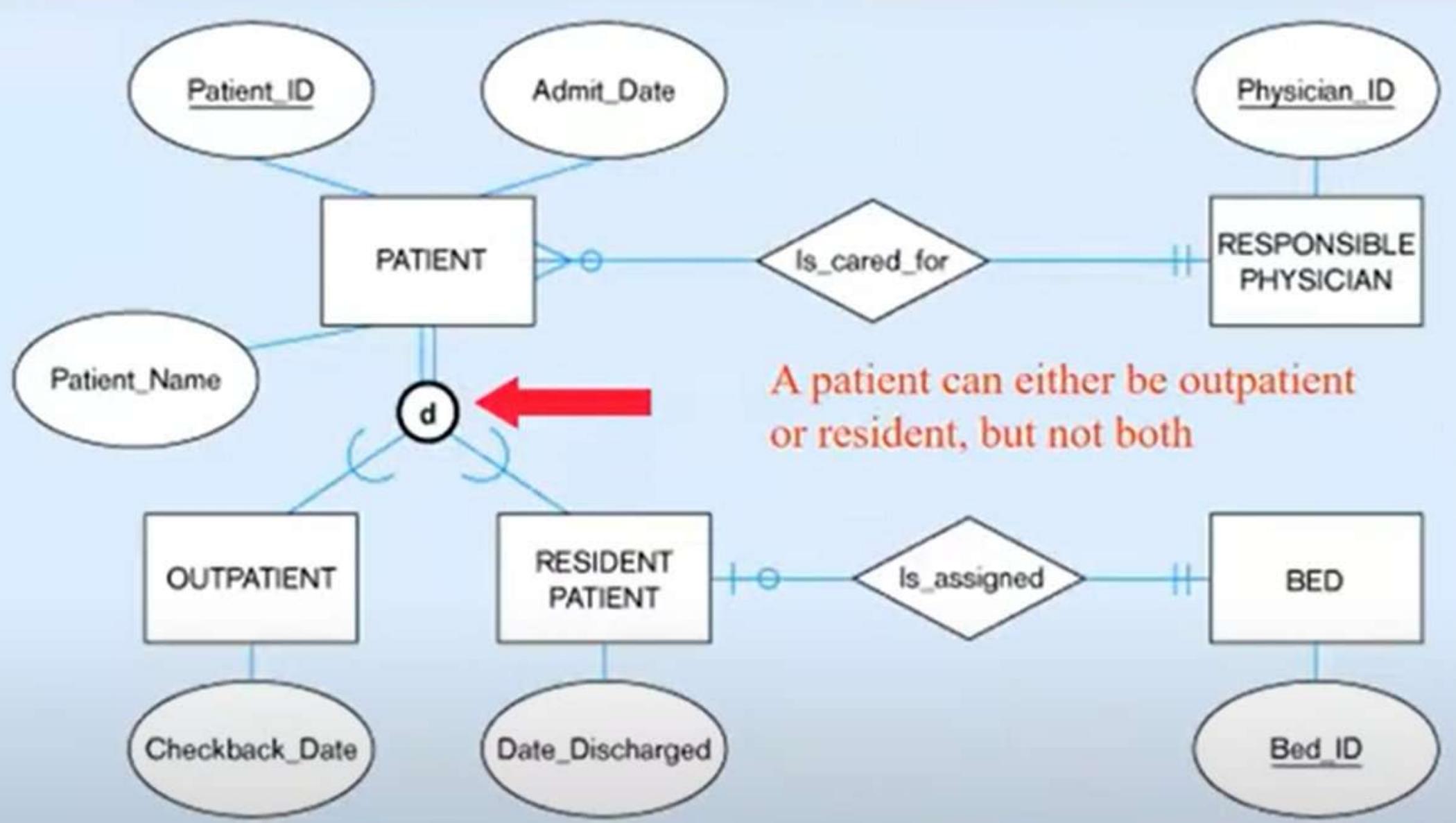
A vehicle could be a car,  
a truck, or neither

## Total specialization rule



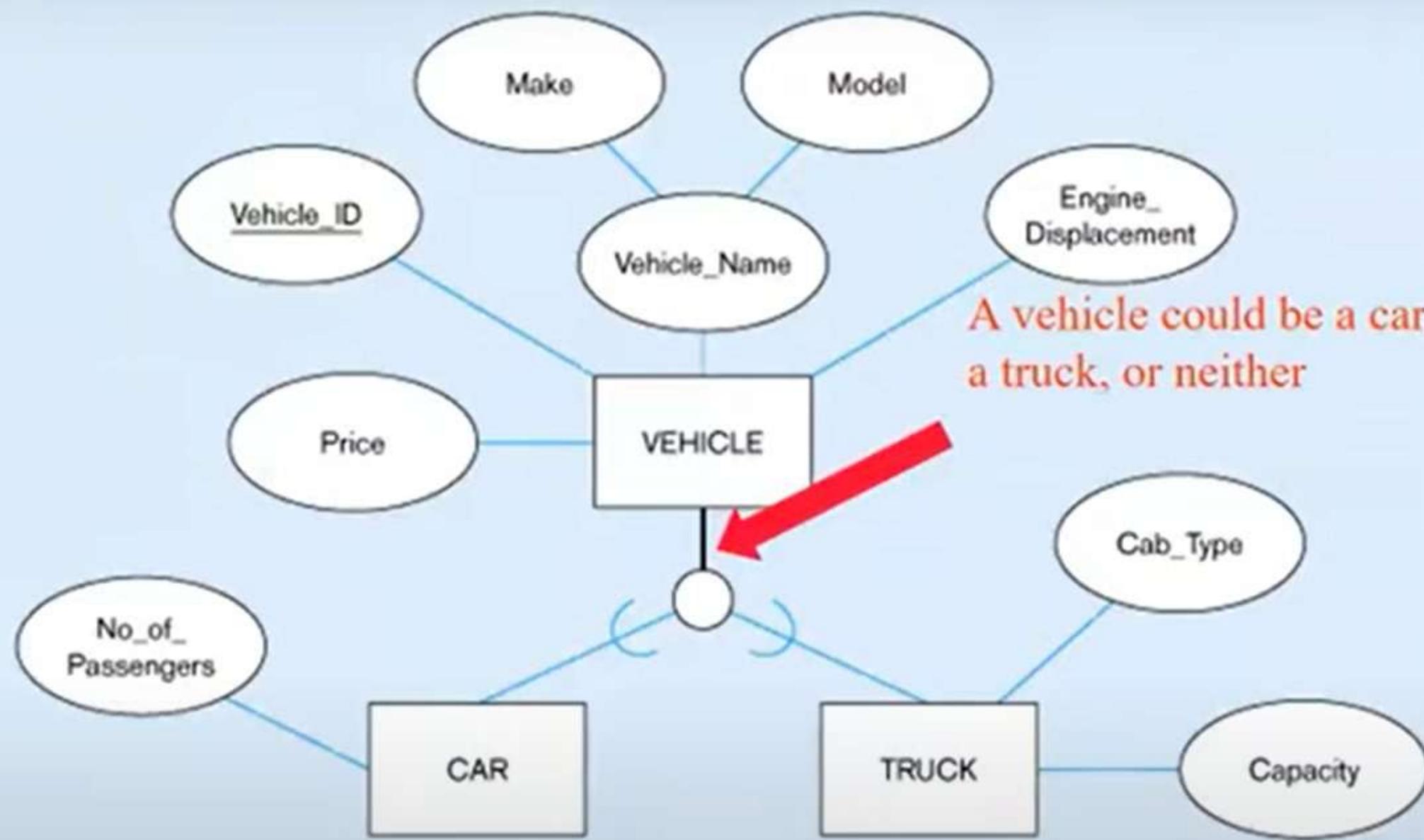


A vehicle could be a car,  
a truck, or neither



Activate

A vehicle could be a car,  
a truck, or neither



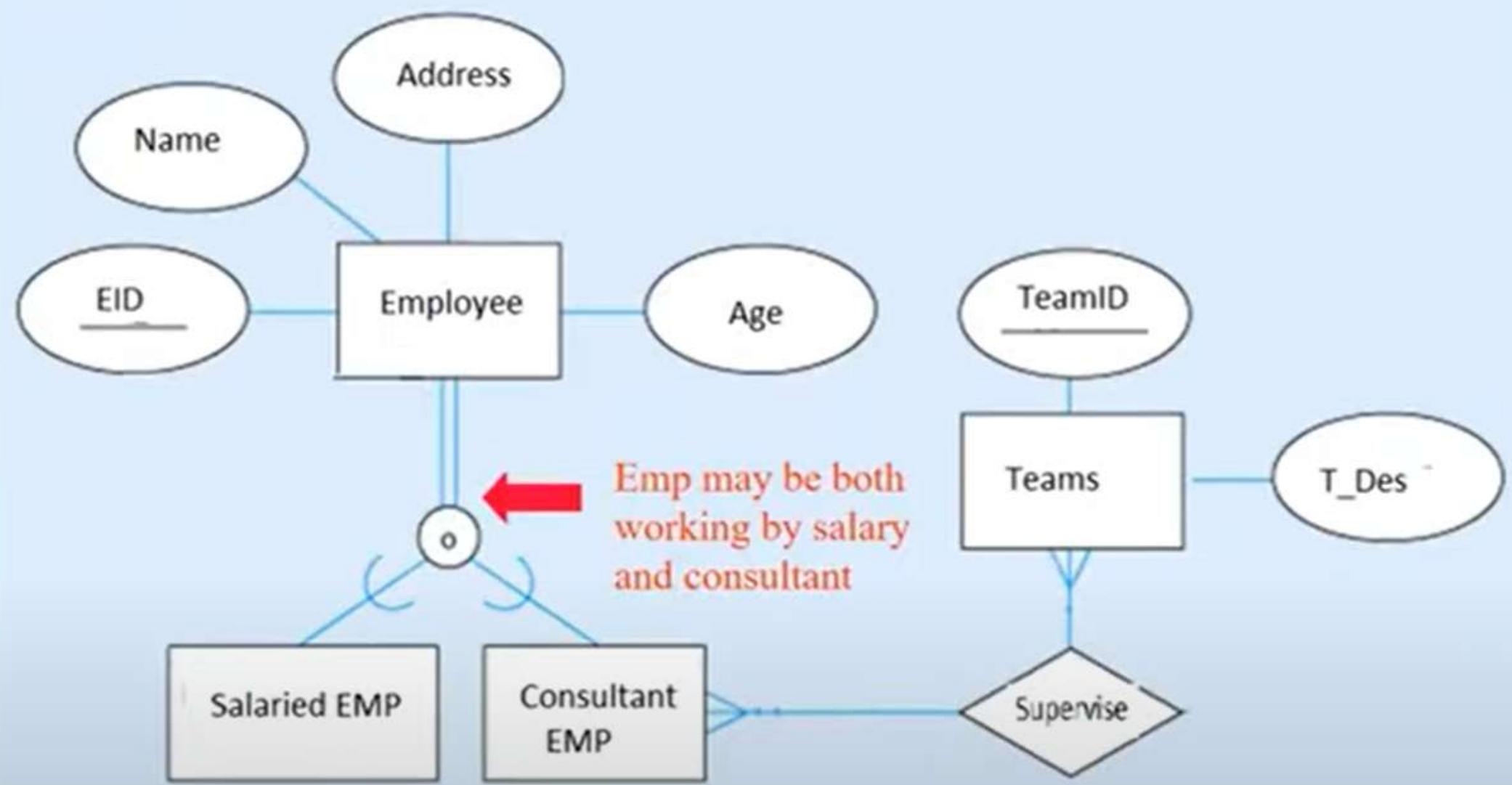
# Constraints in Supertype

## Completeness Constraints:

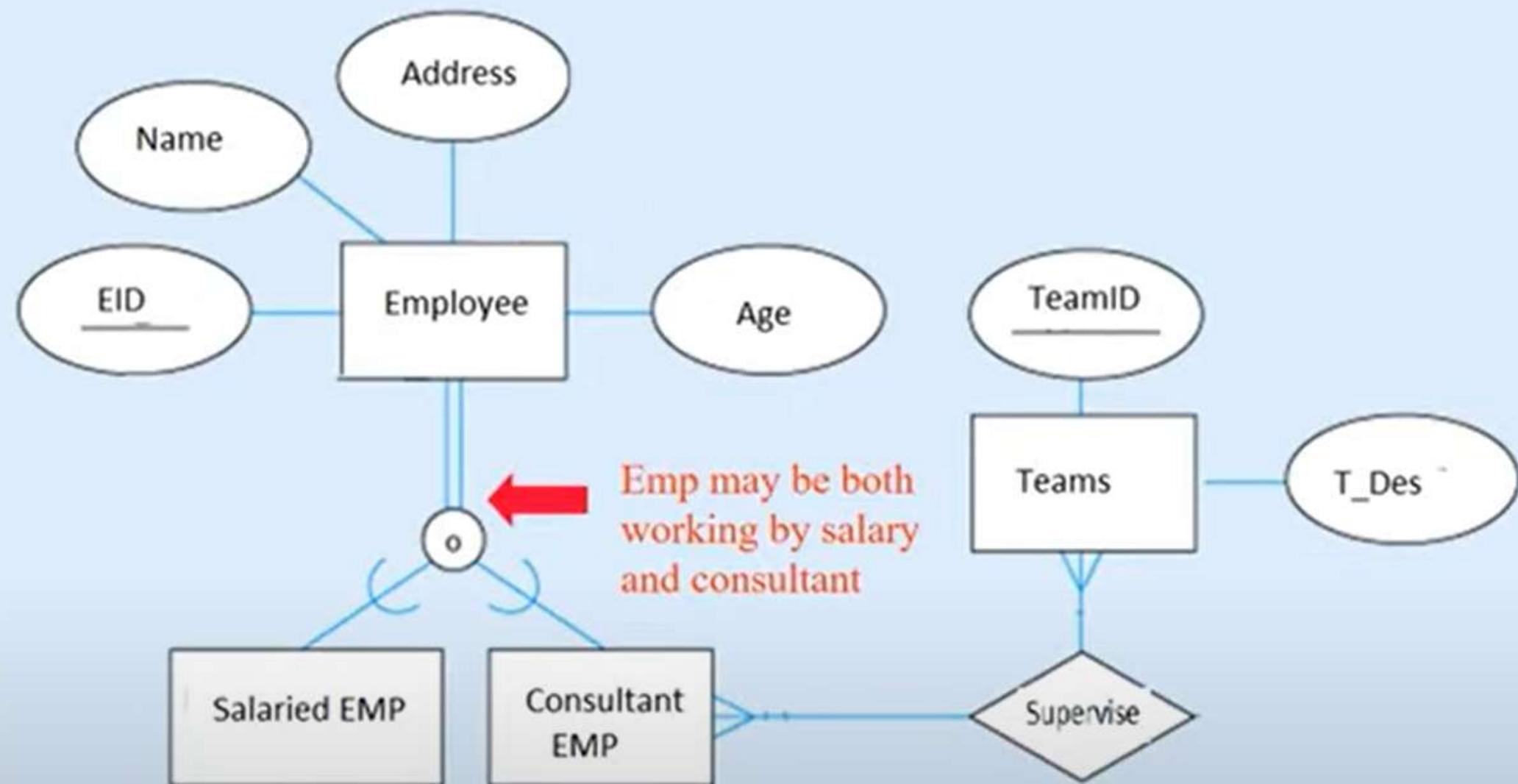
- Total Specialization Rule: Yes (double line)
- Partial Specialization Rule: No (single line)

## Disjointness Constraints:

- Total Disjoint (d)
- Overlap Rule (o)

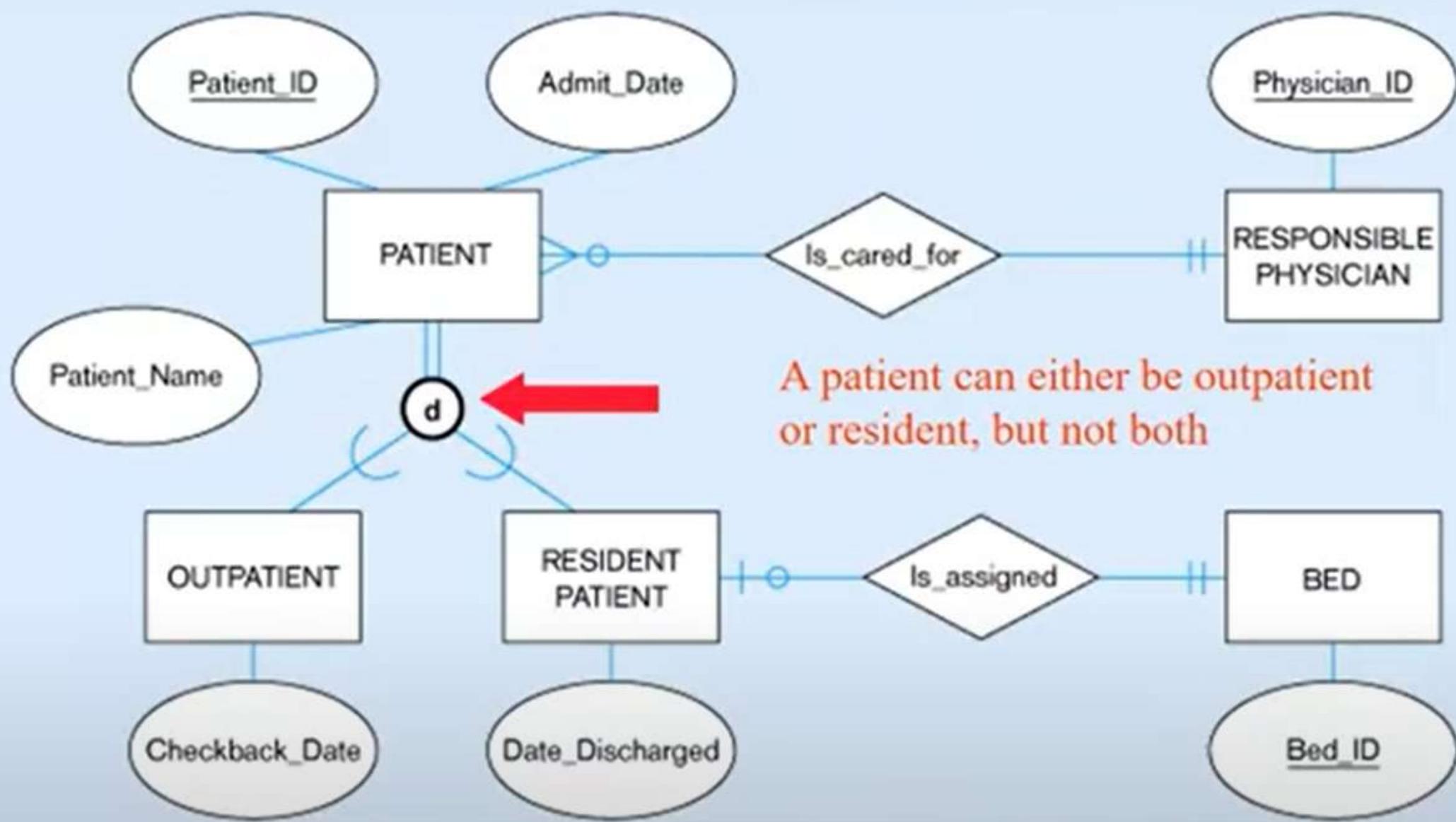


Overlap rule

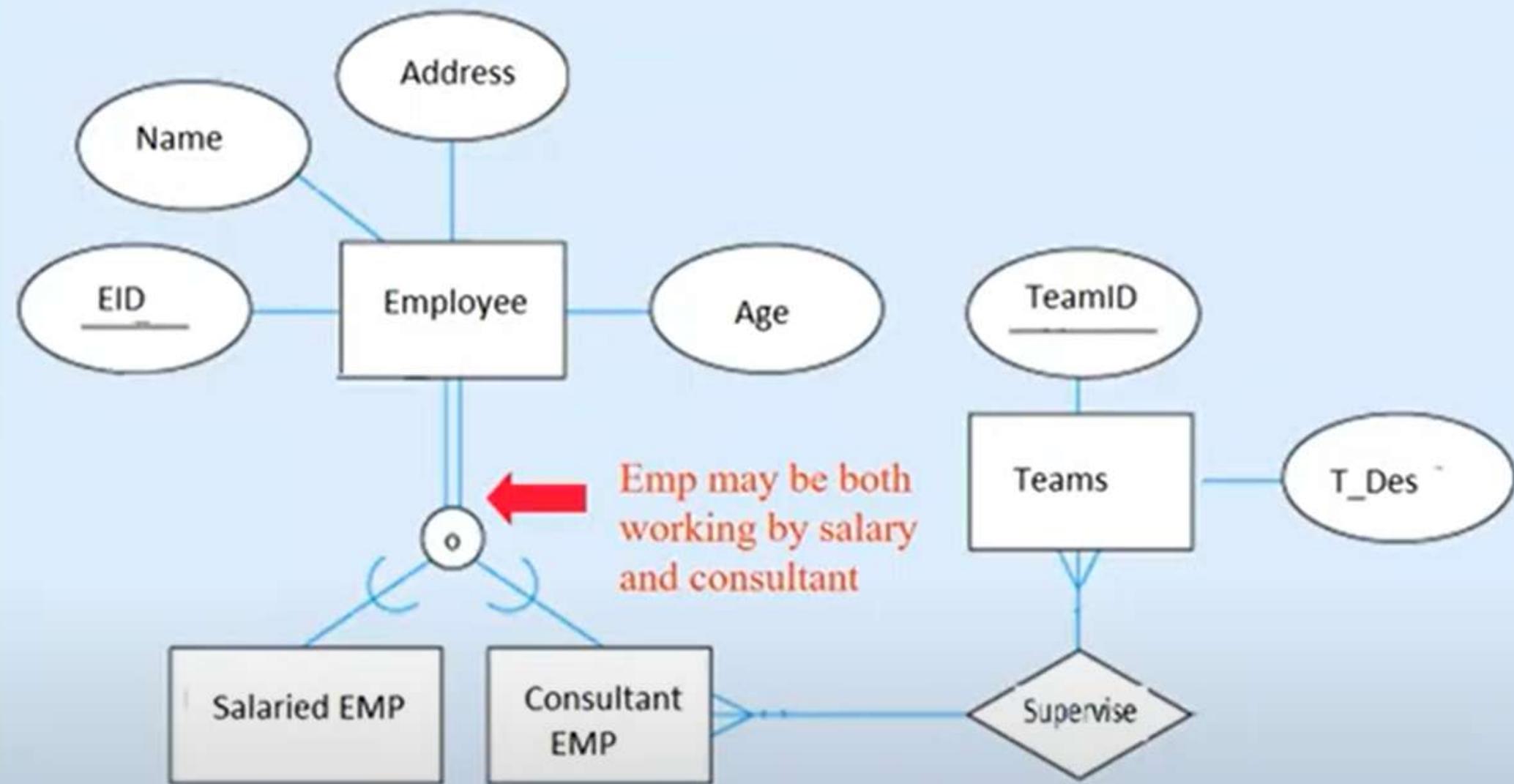


Overlap rule

## Disjoint rule



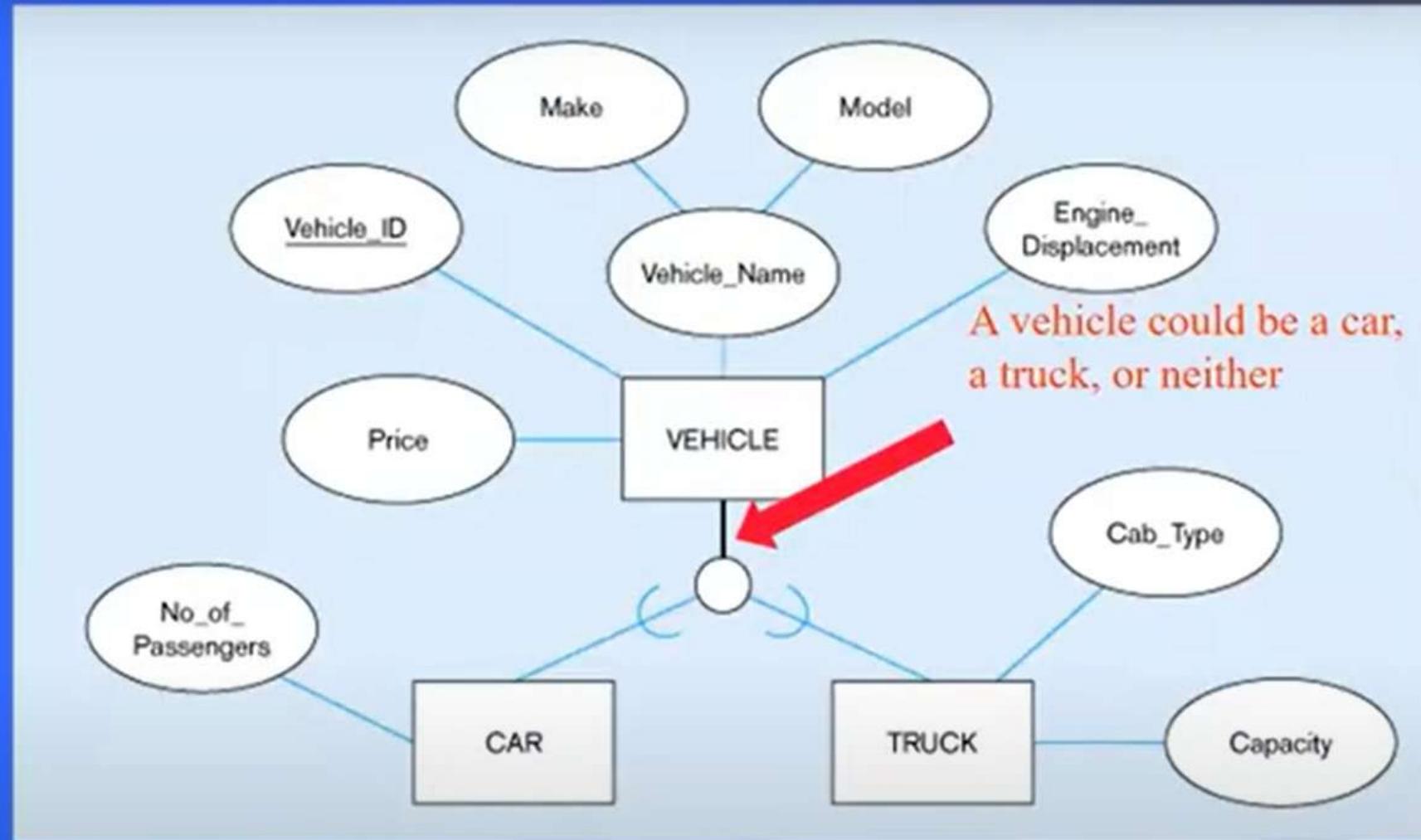
Activat



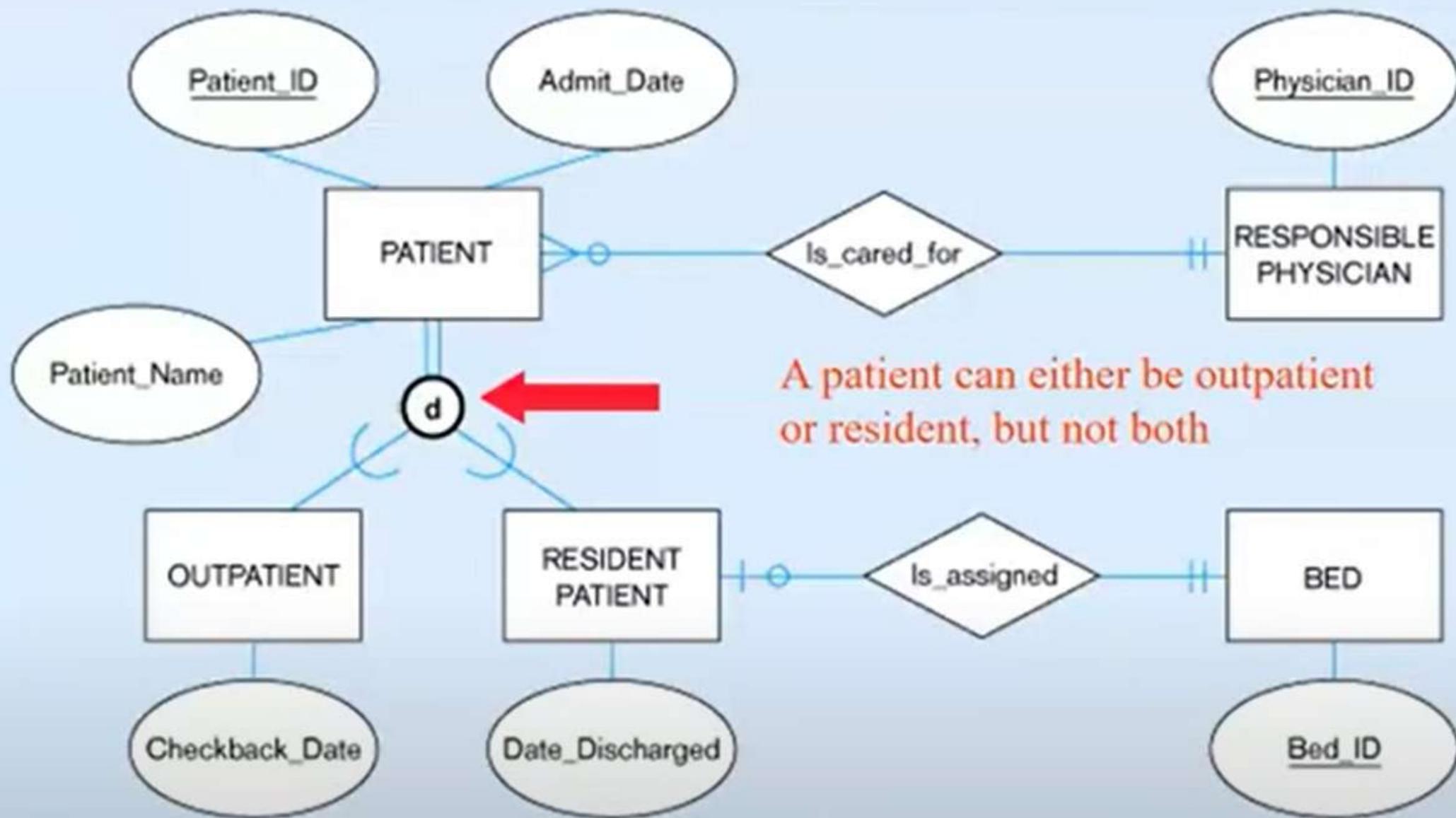
Overlap rule

## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

### Partial specialization rule



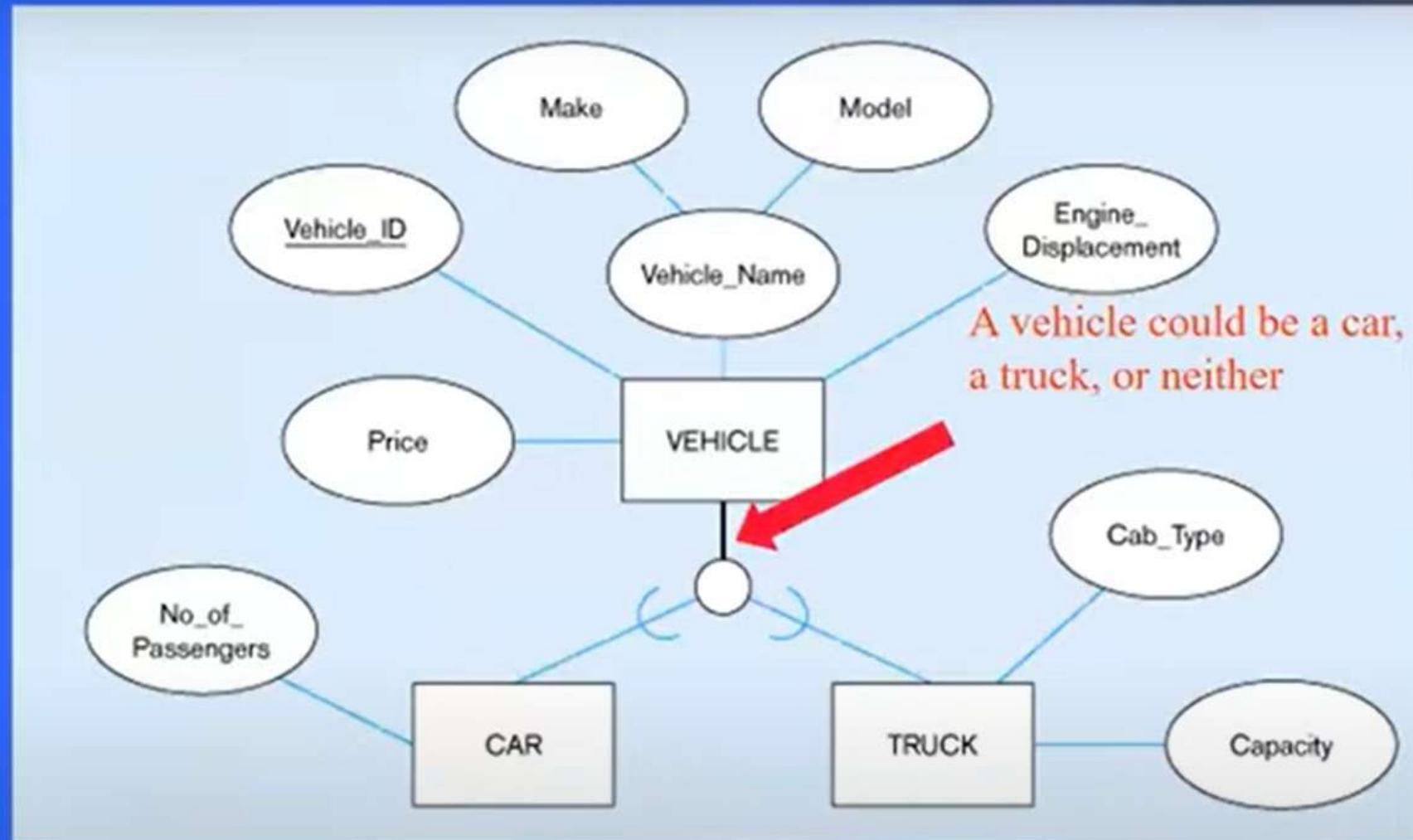
## Disjoint rule



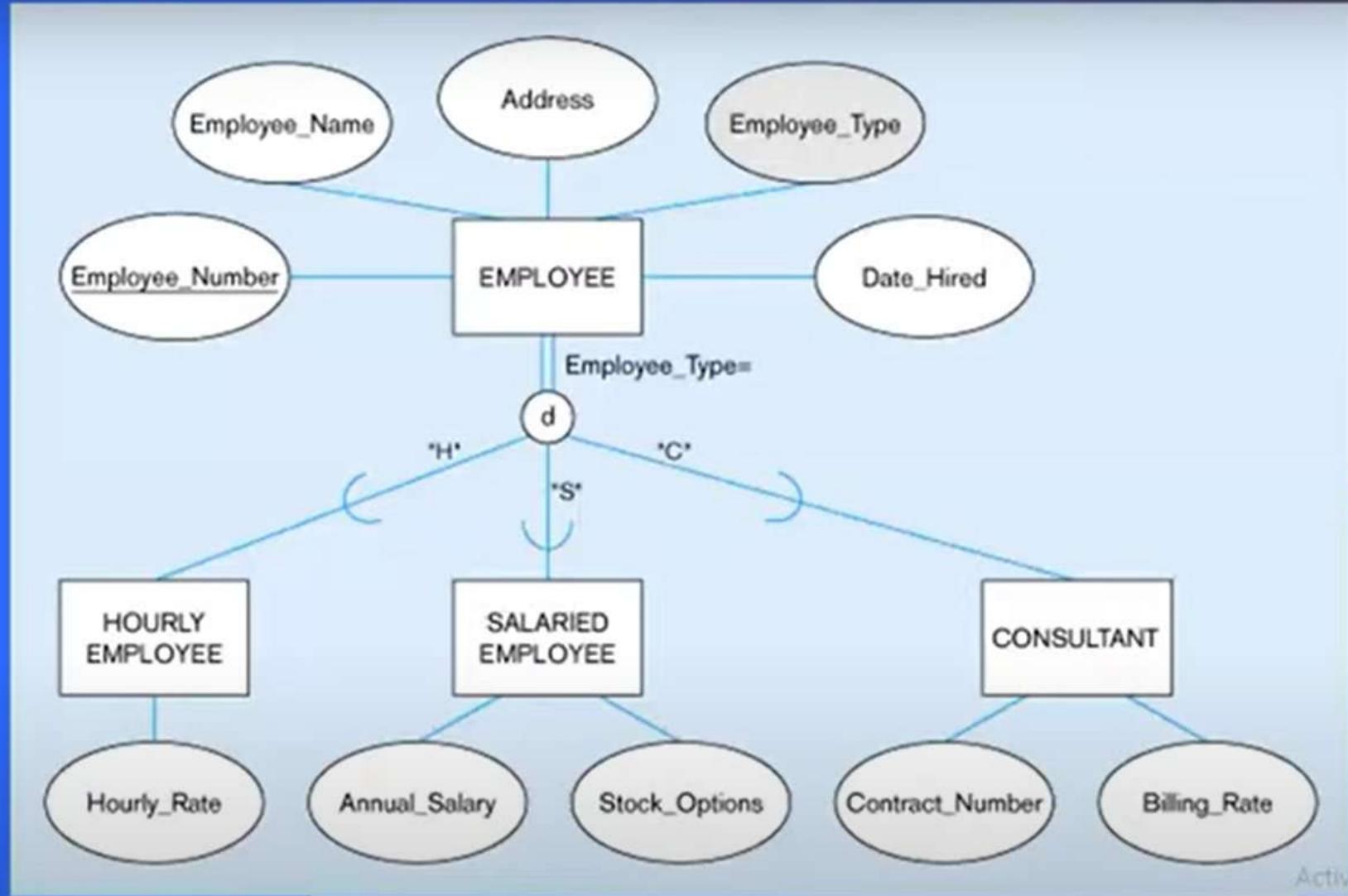
Activati

## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

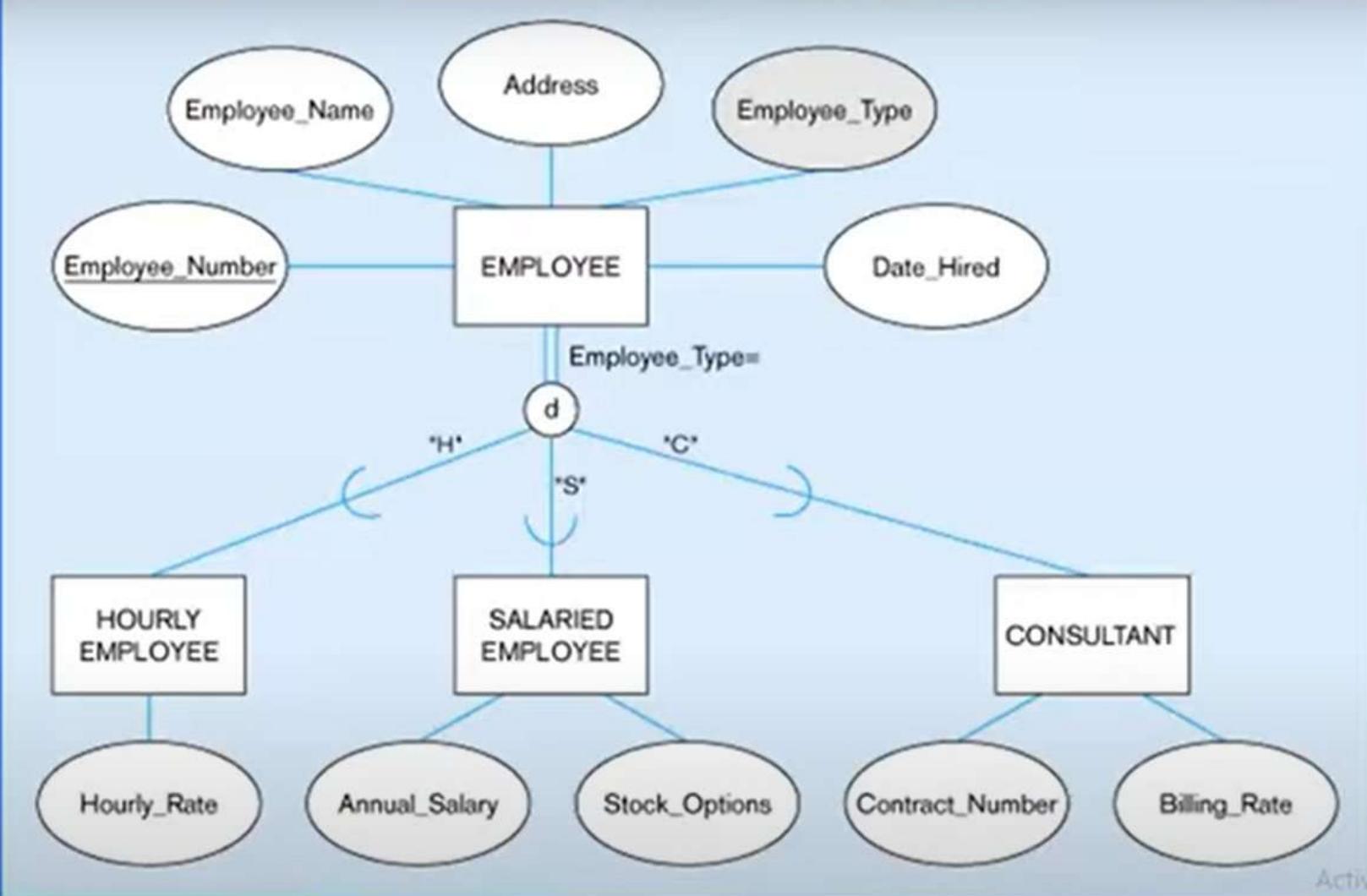
### Partial specialization rule



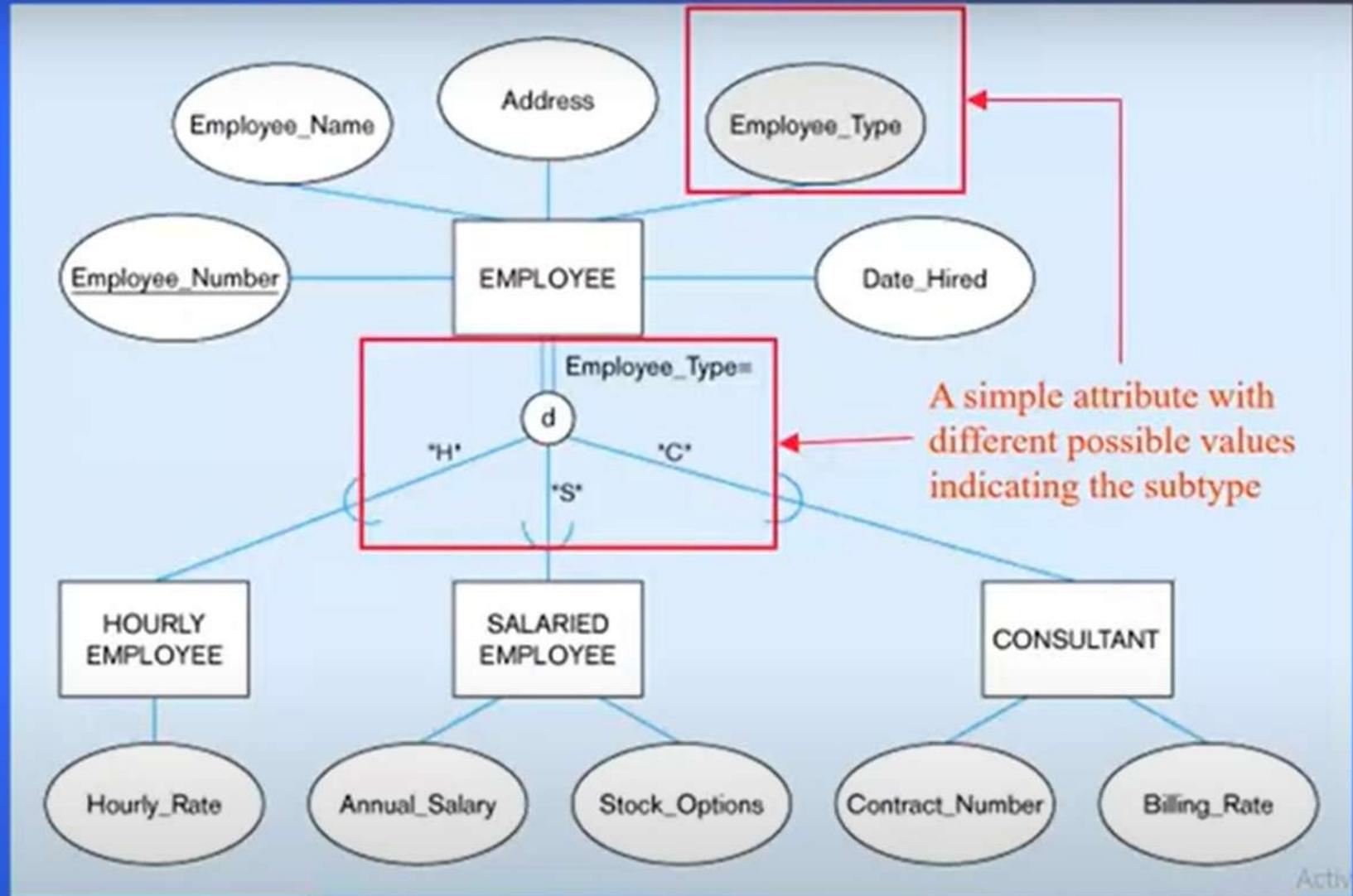
- 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD



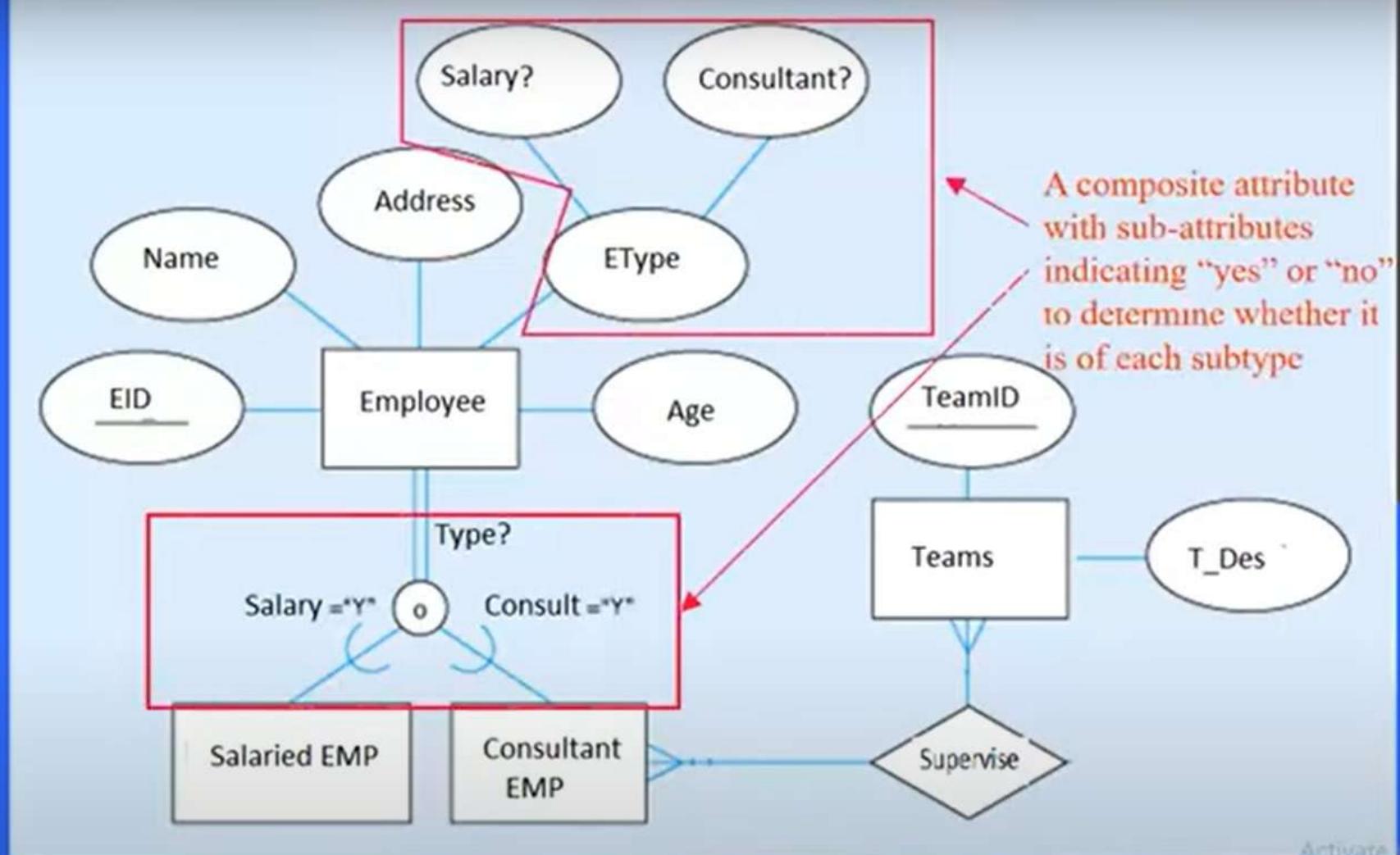
## → 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD



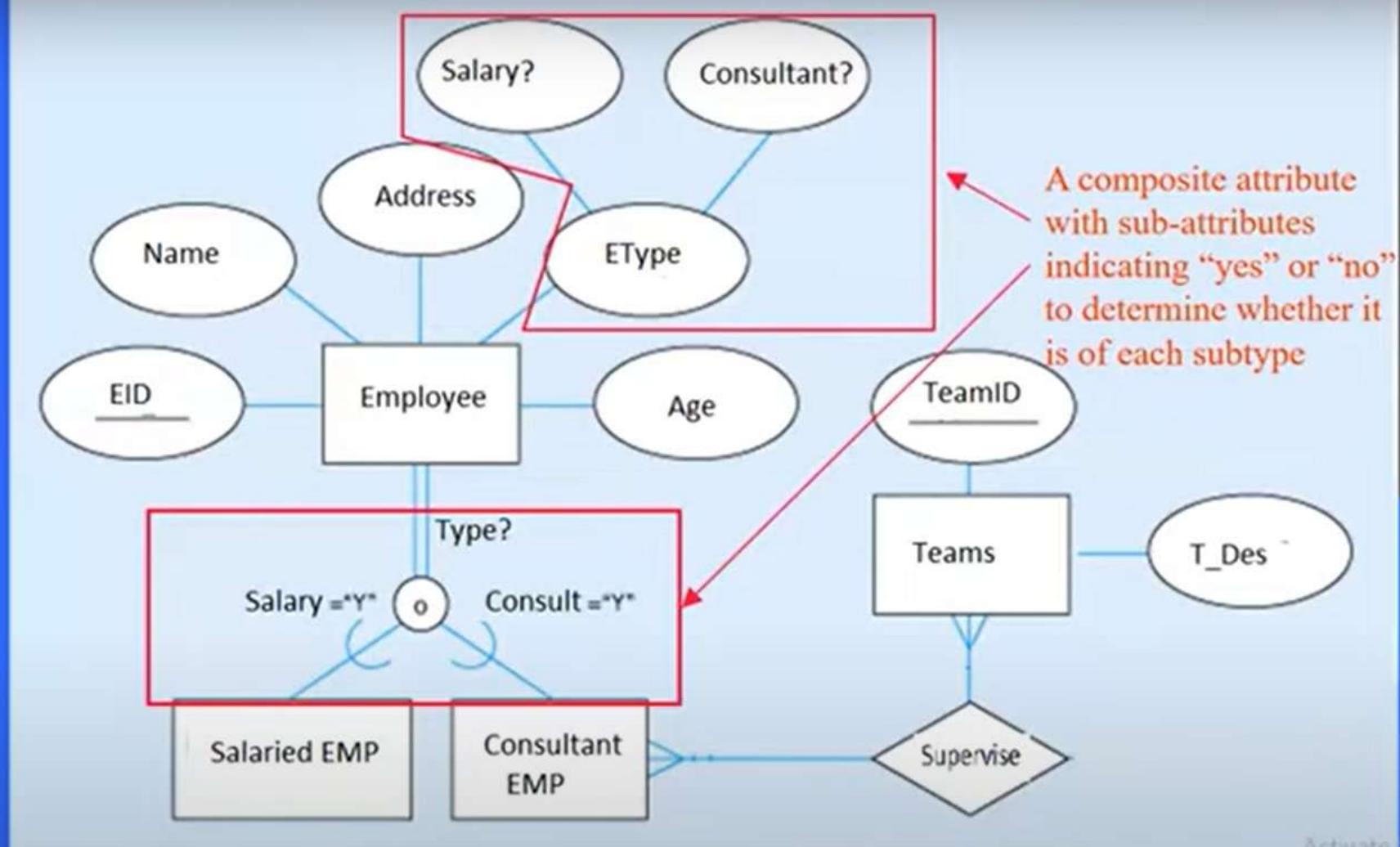
- 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD



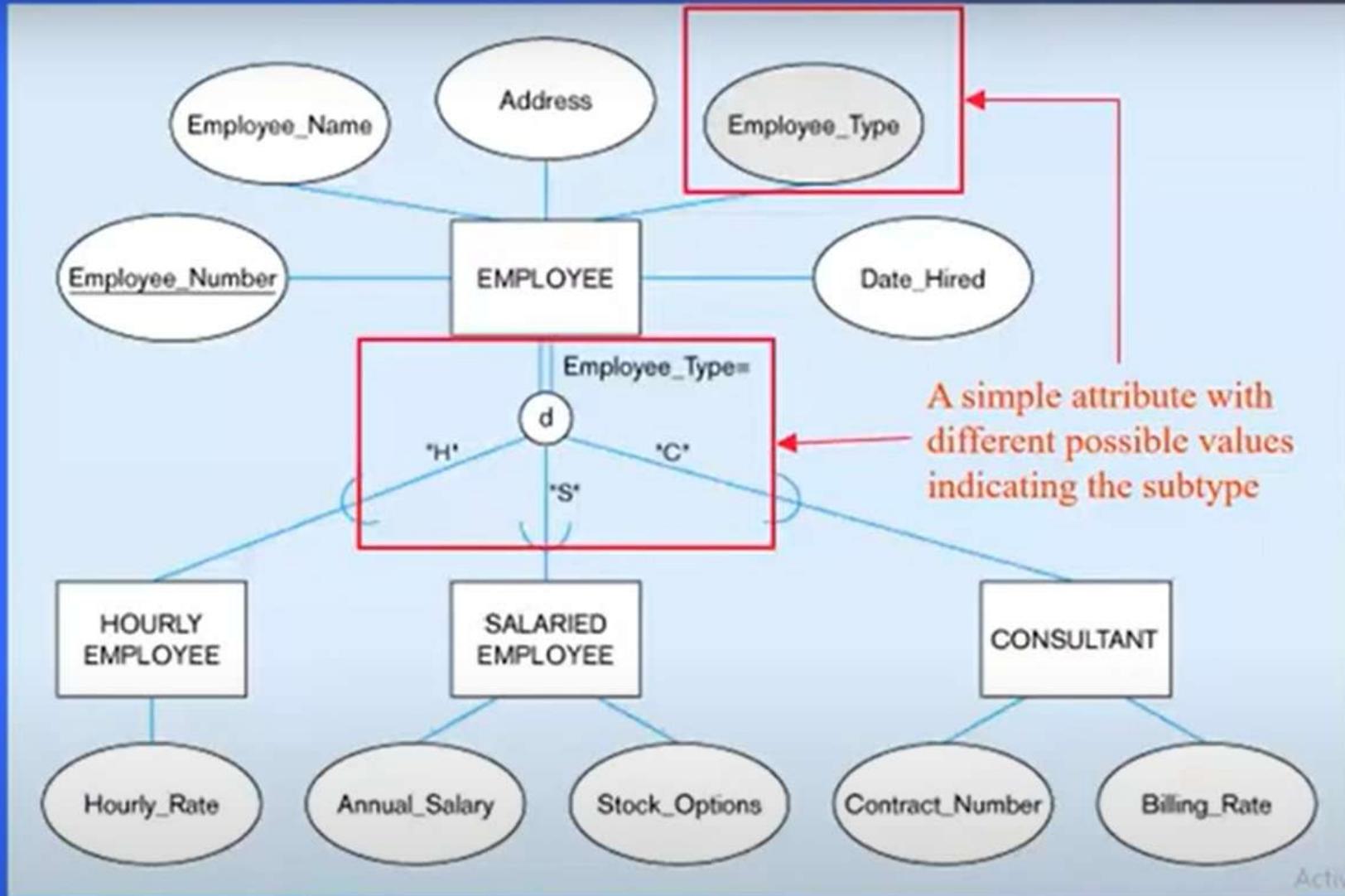
- ⇒ 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD



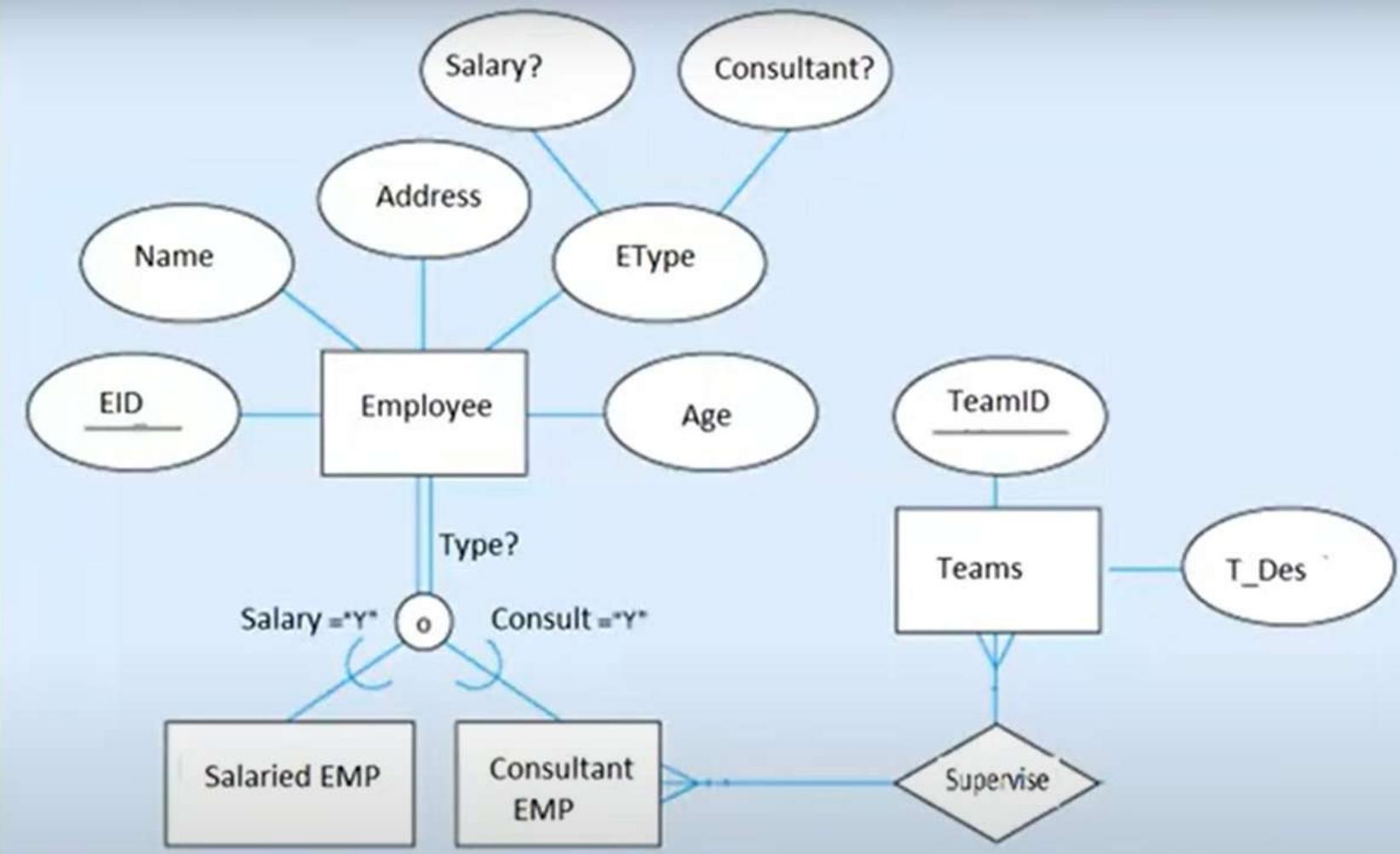
- ⇒ 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD



## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

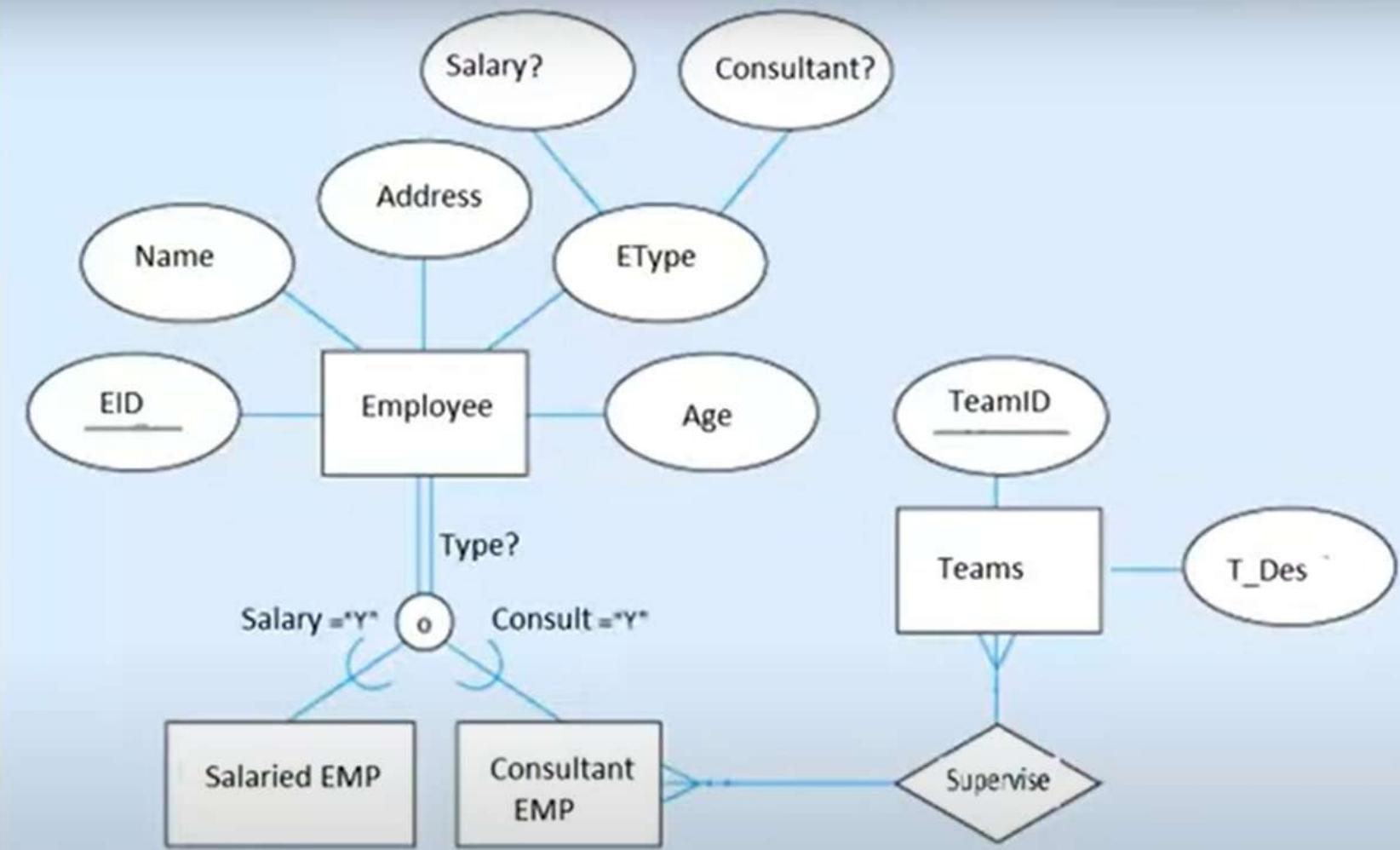


⇒ 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

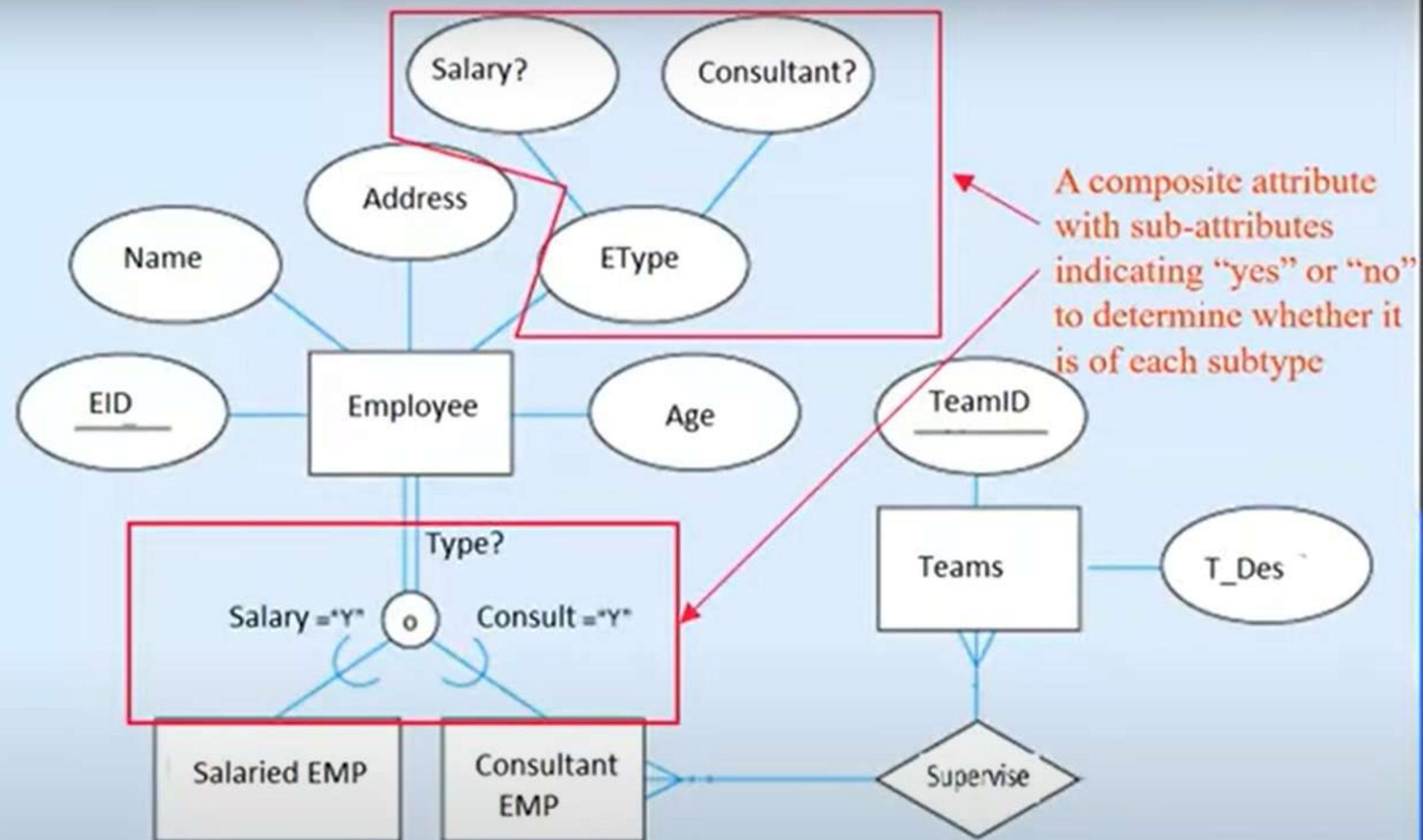


Activate

⇒ 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

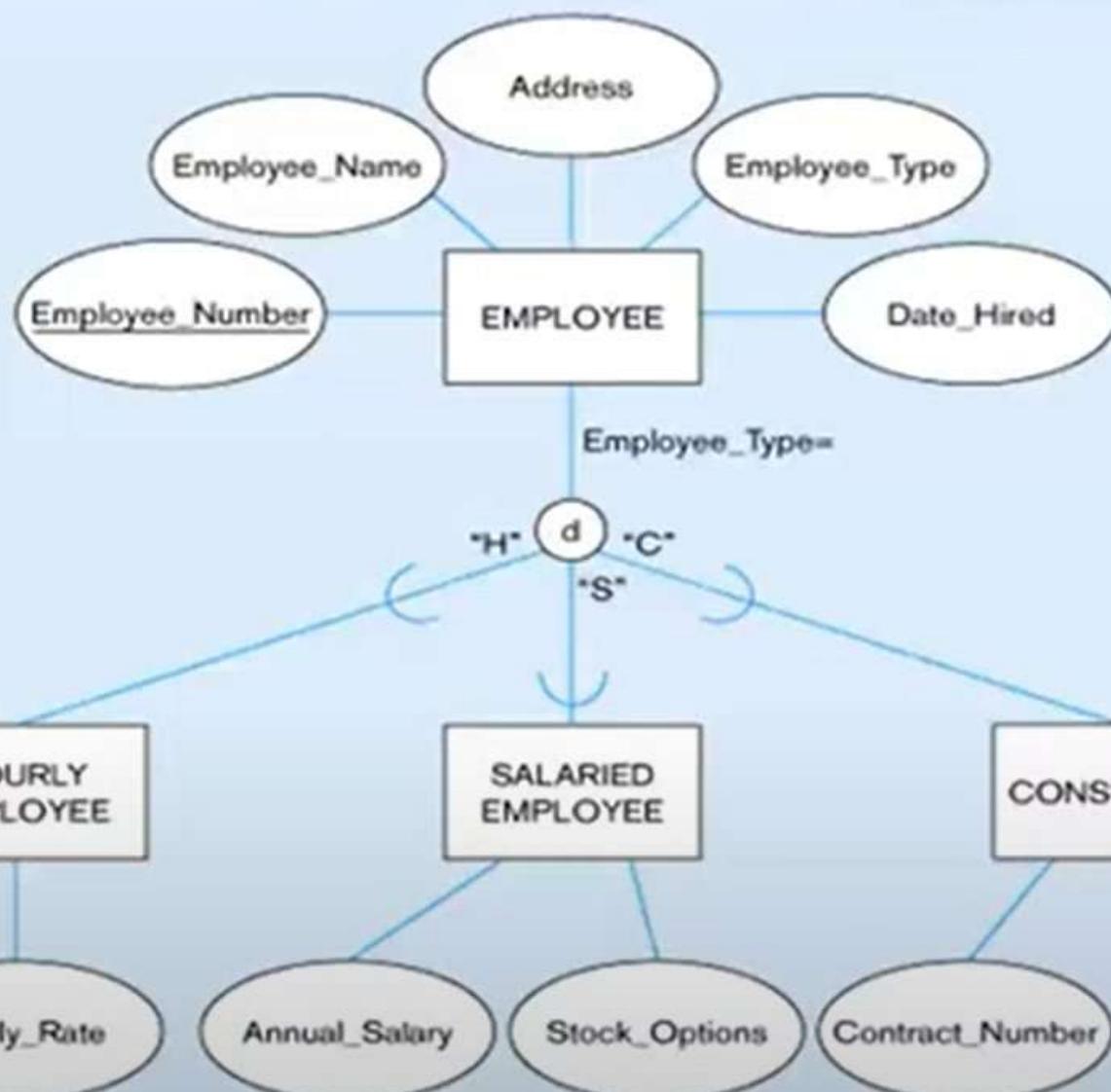


## e Function, Grouping, Union, Subqueries, EERD



unction, Grouping, Union, Subqueries, EERD

# Transforming EER Diagrams into Relations



Join, Grouping, Union, Subqueries, EERD

## Mapping Supertype/subtype relationships to relations

### EMPLOYEE

<u>Employee_Number</u>	Employee_Name	Address	Employee_Type	Date_Hired
------------------------	---------------	---------	---------------	------------

### HOURLY\_EMPLOYEE

<u>H_Employee_Number</u>	Hourly_Rate
--------------------------	-------------

### SALARIED\_EMPLOYEE

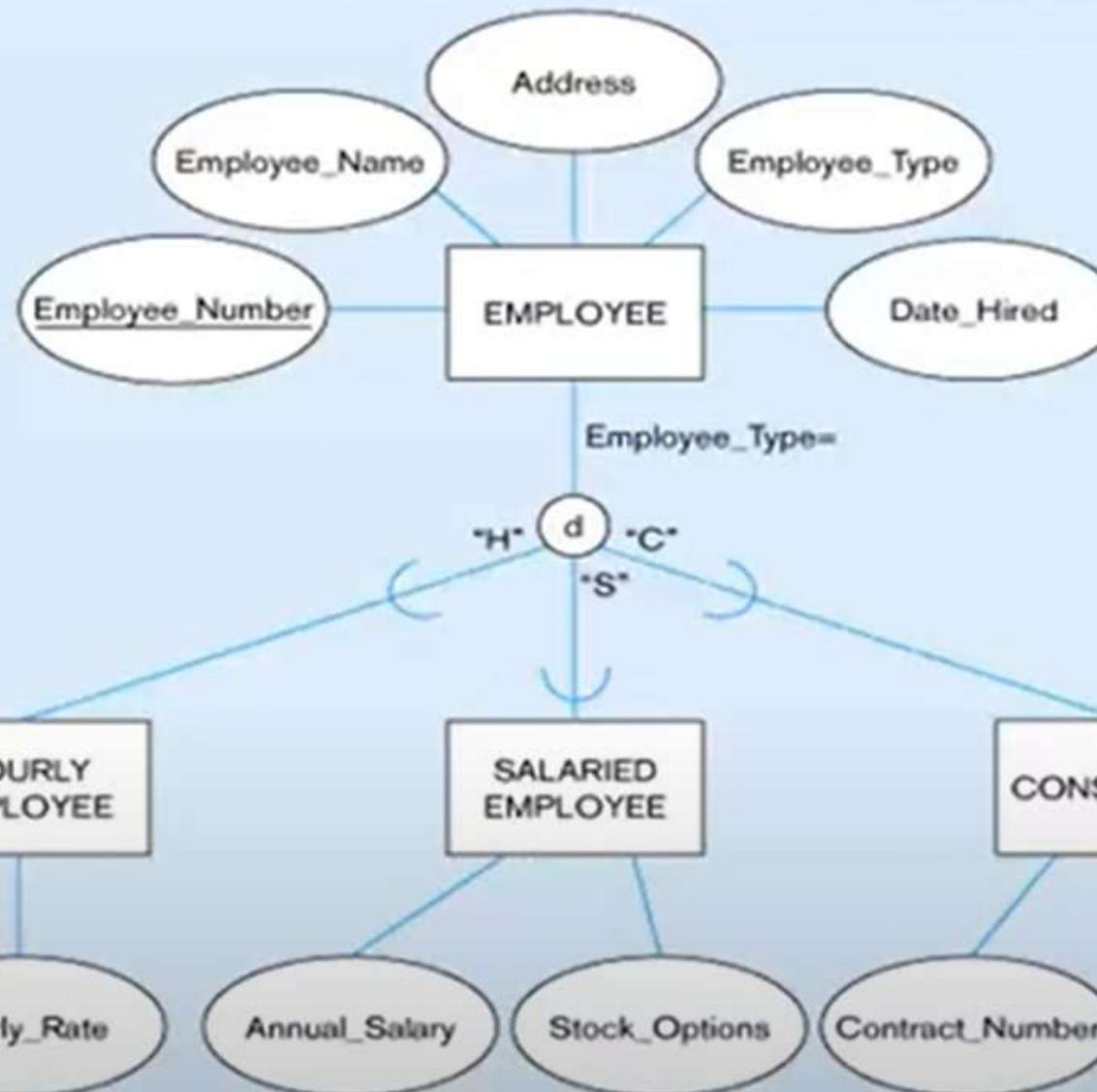
<u>S_Employee_Number</u>	Annual_Salary	Stock_Options
--------------------------	---------------	---------------

### CONSULTANT

<u>C_Employee_Number</u>	Contract_Number	Billing_Rate
--------------------------	-----------------	--------------

Junction, Grouping, Union, Subqueries, EERD

# Transforming EER Diagrams into Relations



ion, Grouping, Union, Subqueries, EERD

## Mapping Supertype/subtype relationships to relations

### EMPLOYEE

<u>Employee_Number</u>	Employee_Name	Address	Employee_Type	Date_Hired
------------------------	---------------	---------	---------------	------------

### HOURLY\_EMPLOYEE

<u>H_Employee_Number</u>	Hourly_Rate
--------------------------	-------------

### SALARIED\_EMPLOYEE

<u>S_Employee_Number</u>	Annual_Salary	Stock_Options
--------------------------	---------------	---------------

### CONSULTANT

<u>C_Employee_Number</u>	Contract_Number	Billing_Rate
--------------------------	-----------------	--------------

ion, Grouping, Union, Subqueries, EERD

## Mapping Supertype/subtype relationships to relations

### EMPLOYEE

<u>Employee_Number</u>	Employee_Name	Address	Employee_Type	Date_Hired
------------------------	---------------	---------	---------------	------------

### HOURLY\_EMPLOYEE

<u>H_Employee_Number</u>	Hourly_Rate
--------------------------	-------------

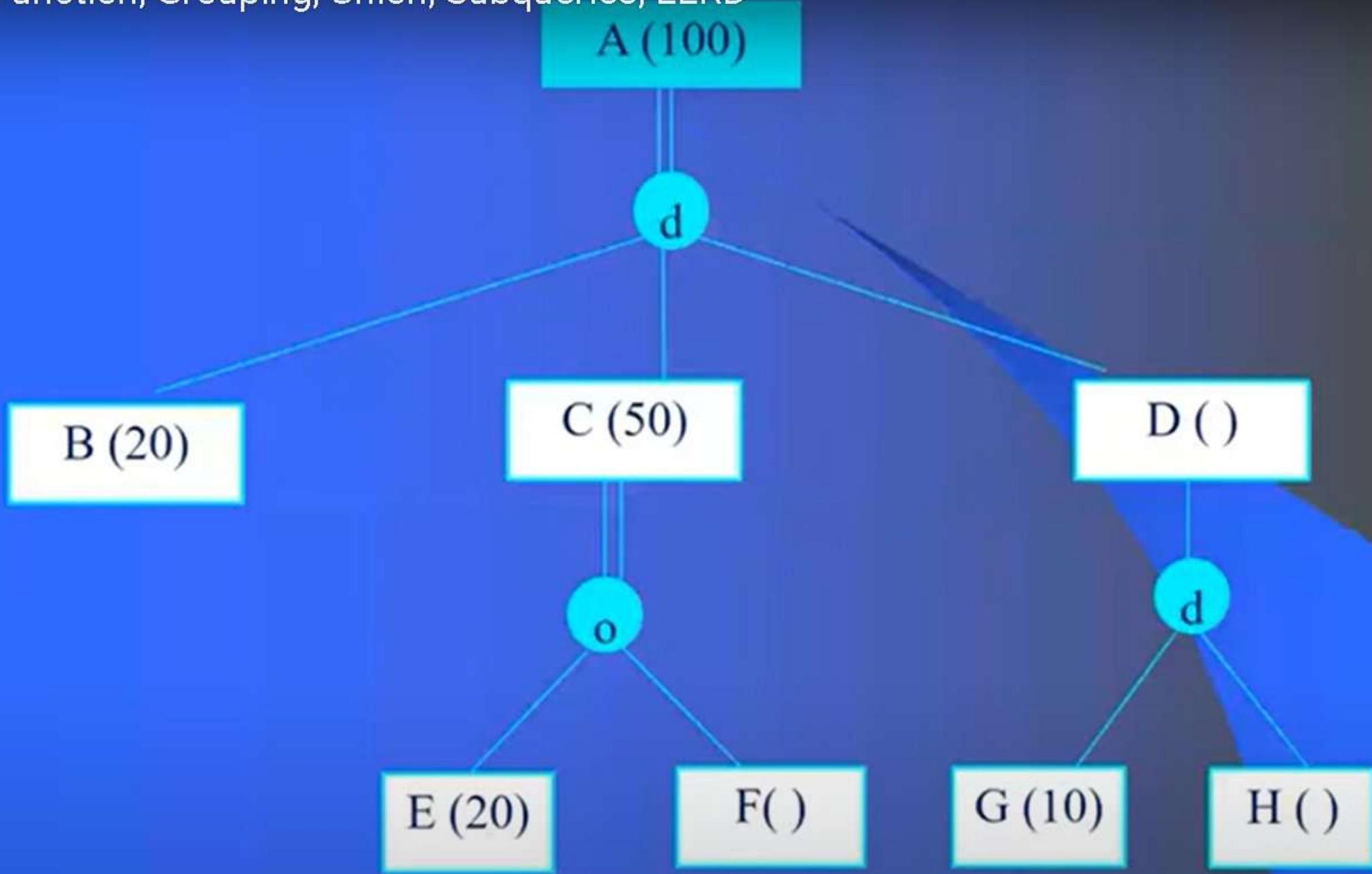
### SALARIED\_EMPLOYEE

<u>S_Employee_Number</u>	Annual_Salary	Stock_Options
--------------------------	---------------	---------------

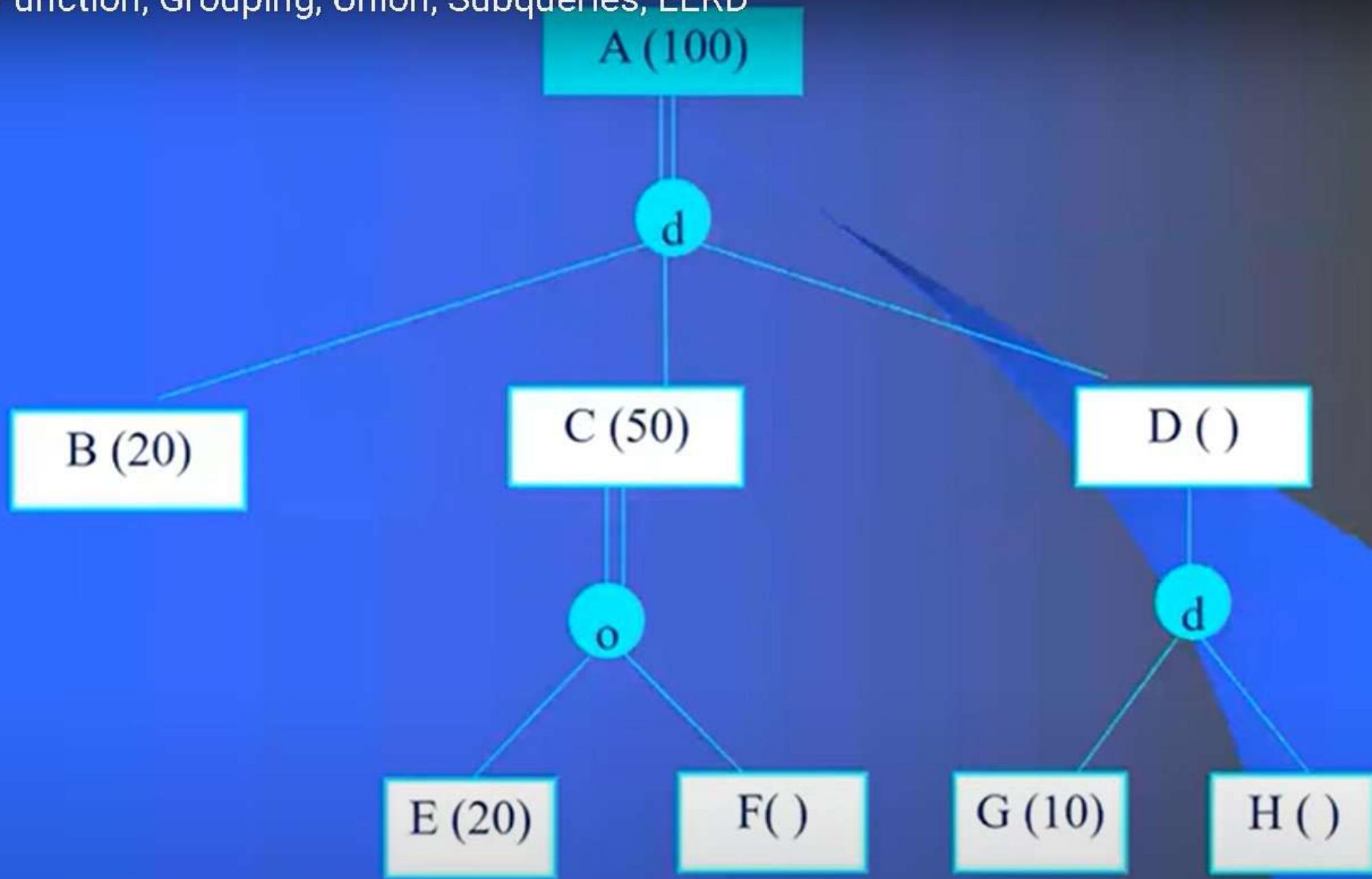
### CONSULTANT

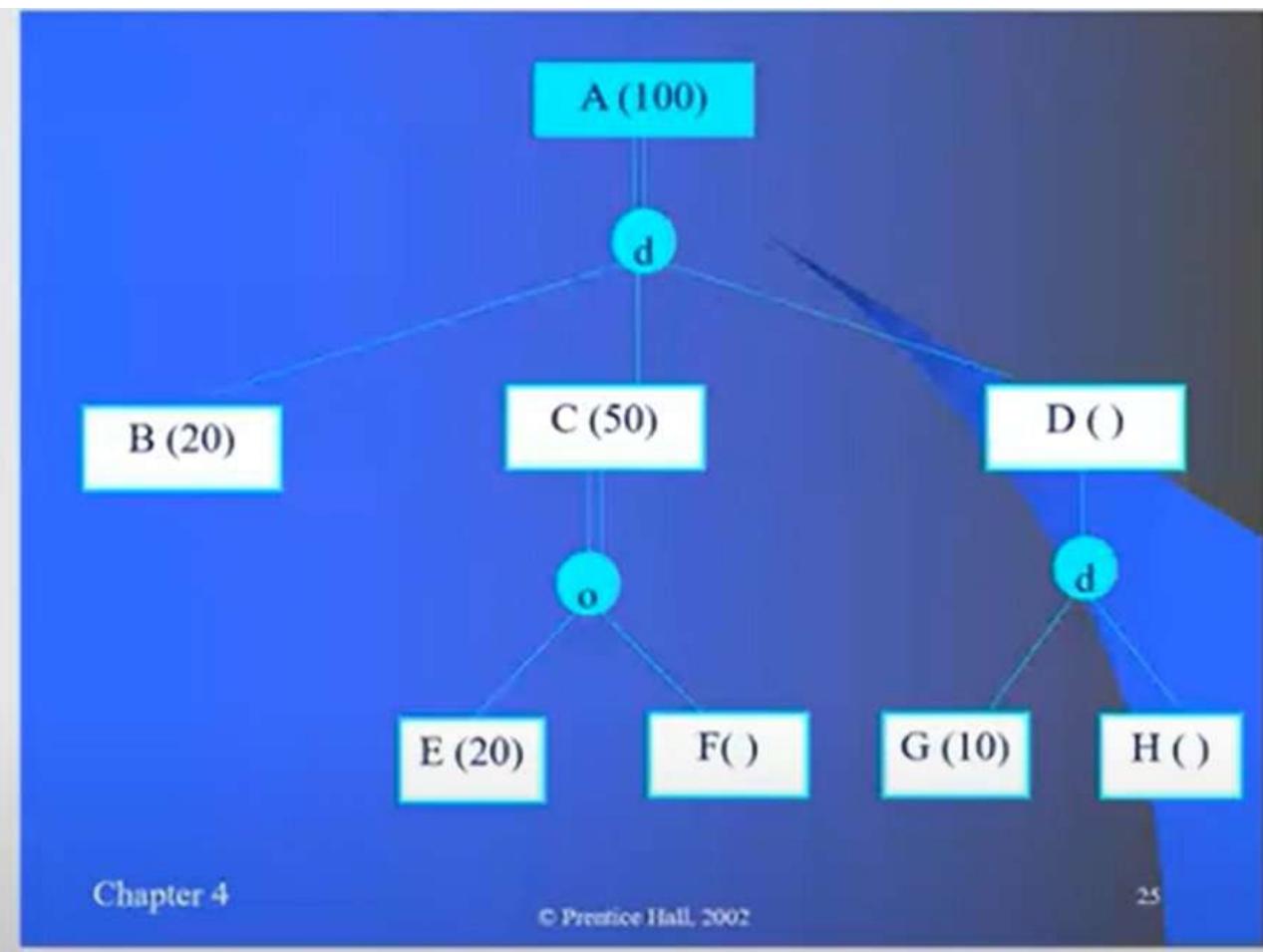
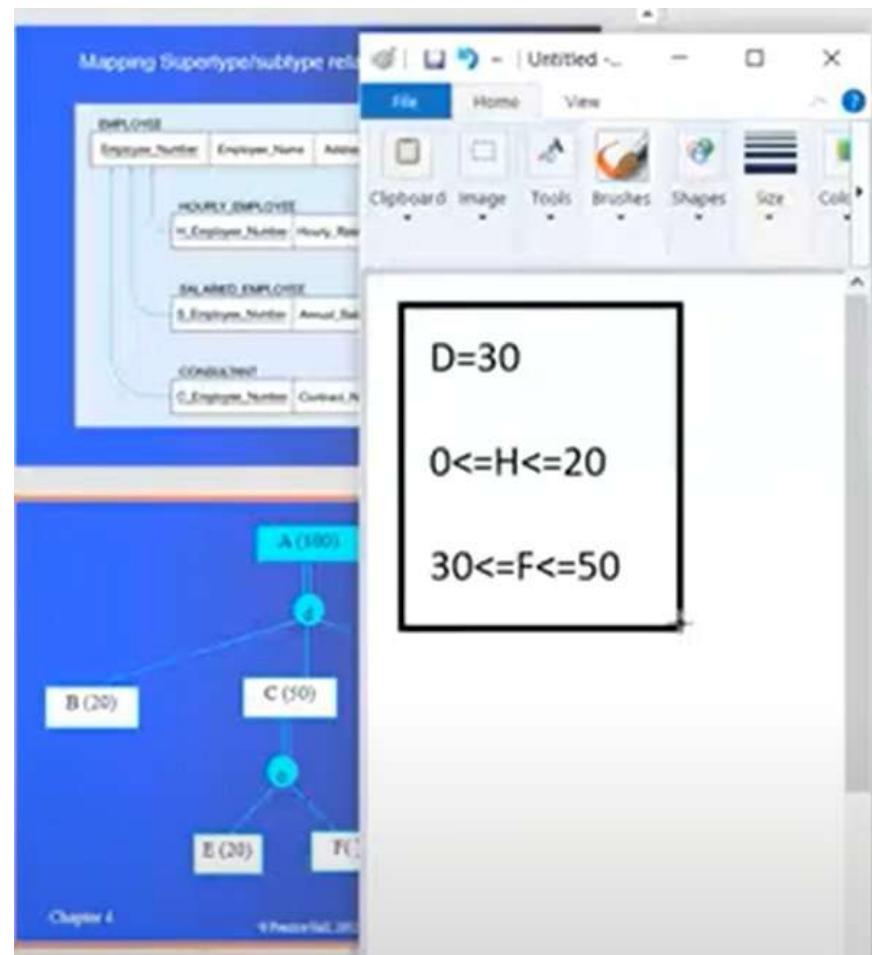
<u>C_Employee_Number</u>	Contract_Number	Billing_Rate
--------------------------	-----------------	--------------

## Aggregate Function, Grouping, Union, Subqueries, EERD

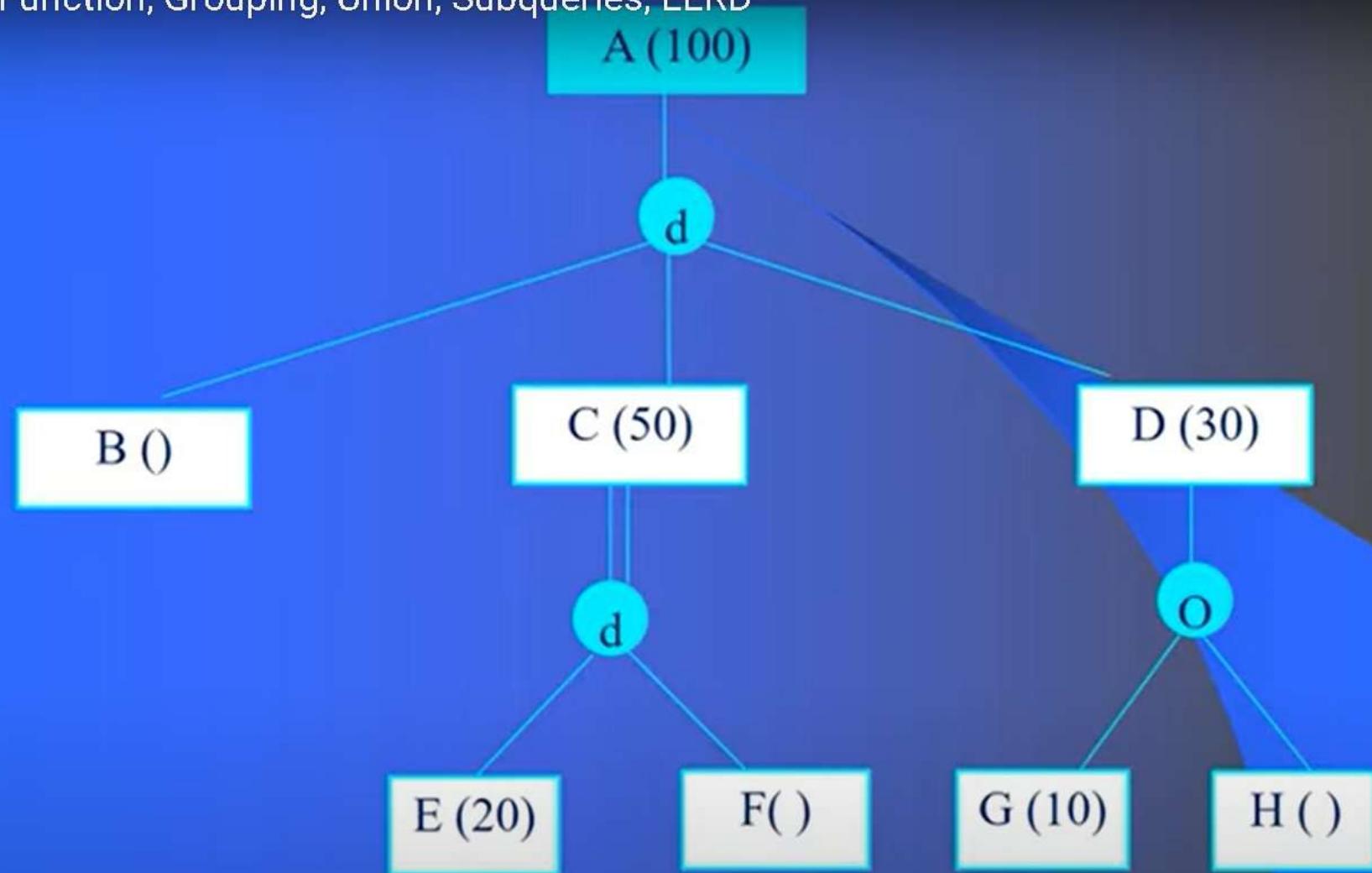


## Aggregate Function, Grouping, Union, Subqueries, EERD

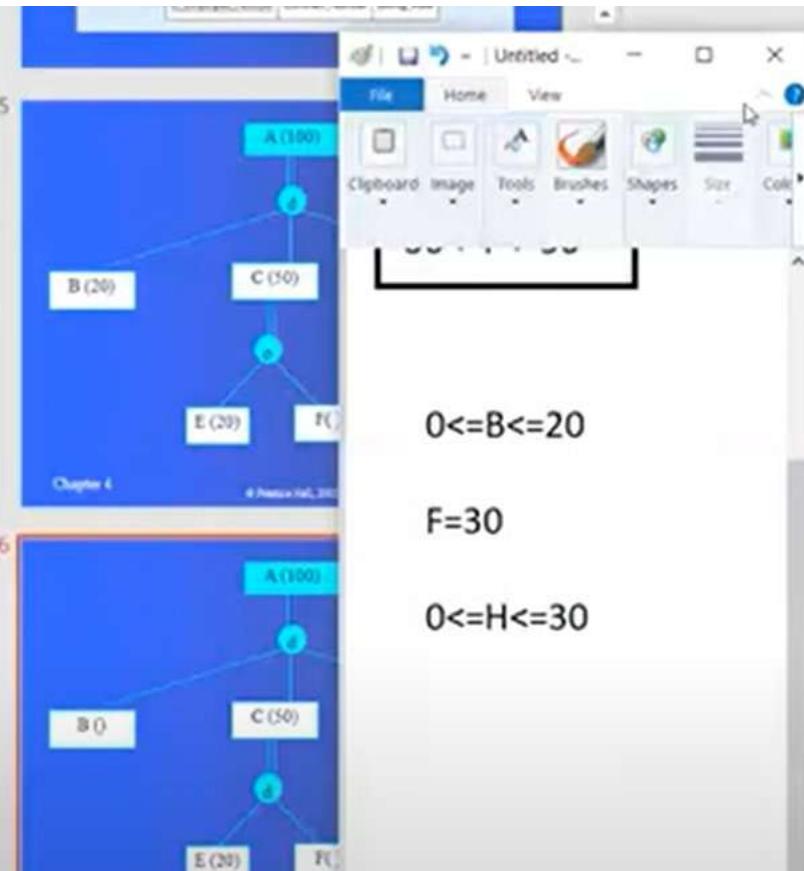




## 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD



25

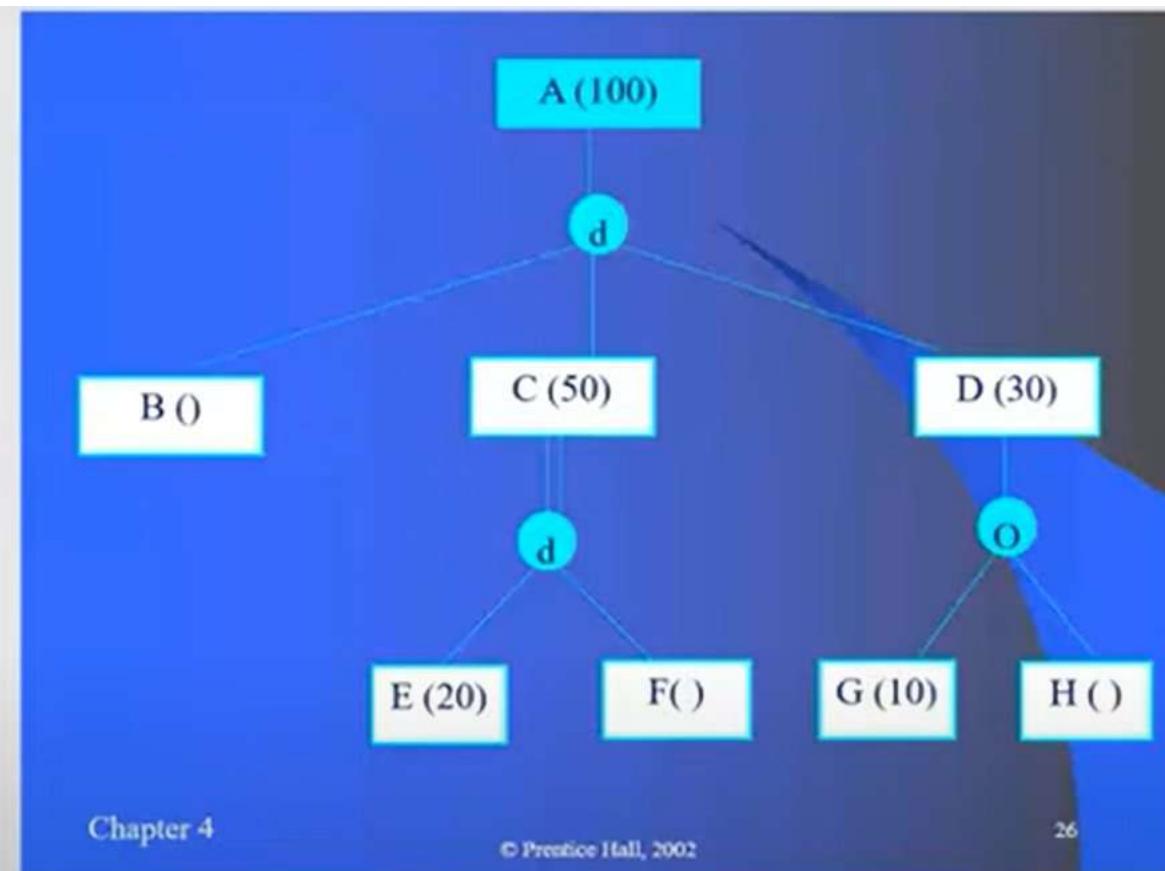


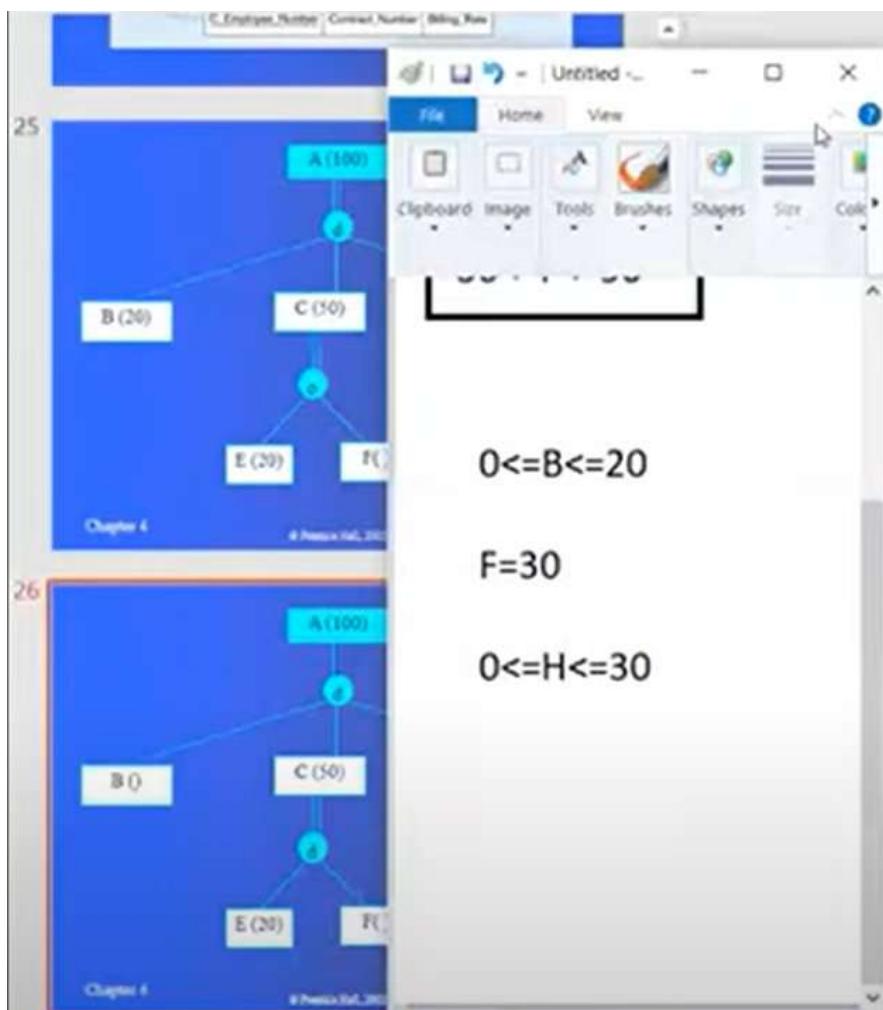
$$0 \leq B \leq 20$$

$$F = 30$$

$$0 \leq H \leq 30$$

26

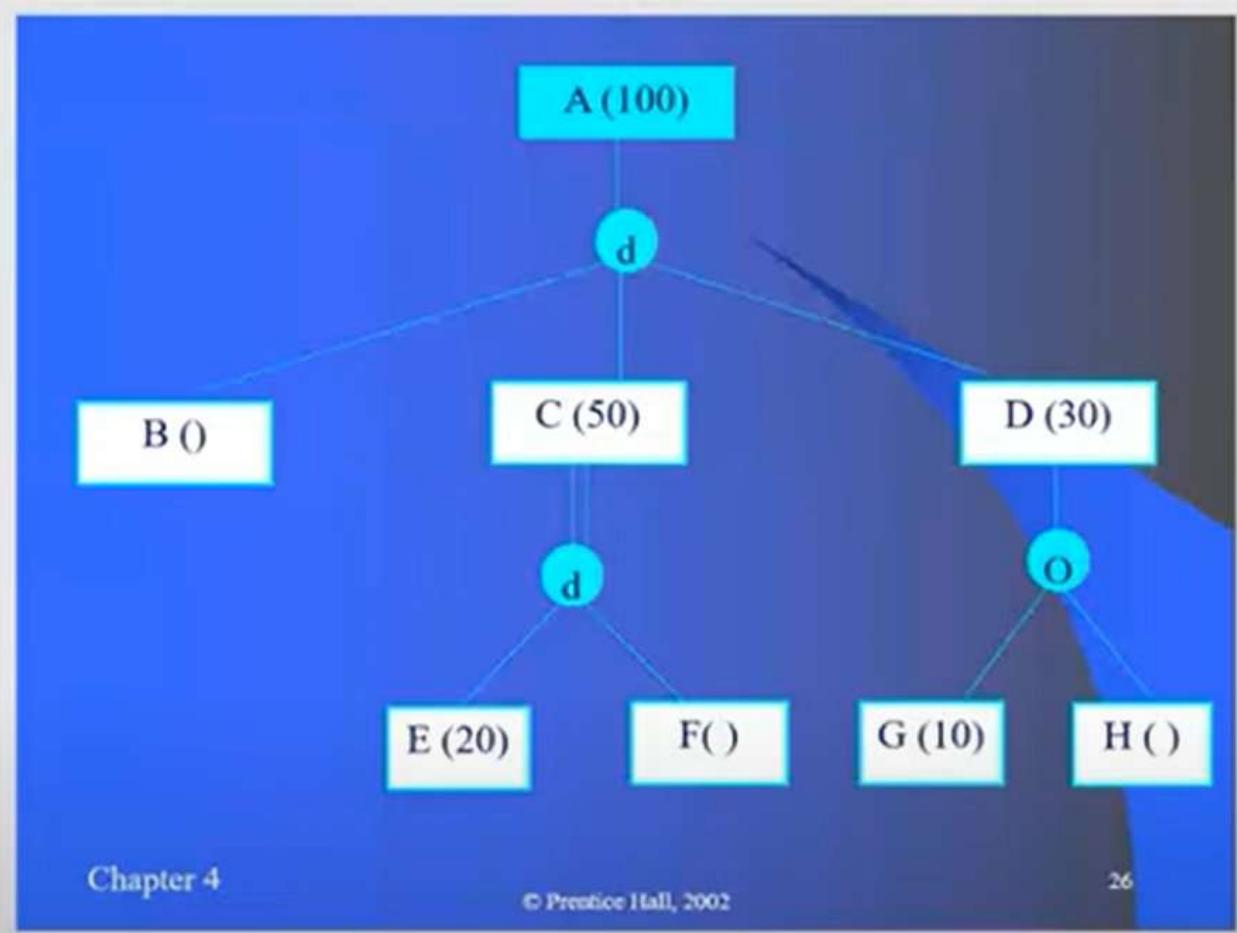




$0 \leq B \leq 20$

$F = 30$

$0 \leq H \leq 30$



Activate Window  
Go To Selection

```
select st_fname,st_age,dept_id  
from Student  
order by st_address
```

```
from Student  
order by st_address
```

```
- select st_fname, st_age, dept_id  
from Student  
order by 1
```

```
from Student  
order by st_address
```

```
select st_fname,st_age,dept_id  
from Student  
order by 2
```

```
from Student  
order by st_address
```

```
select st_fname,st_age,dept_id  
from Student  
order by 2
```

```
select st_fname, st_age, dept_id  
from Student  
order by dept_id, st_age
```

```
select st_fname,st_age,dept_id  
from Student  
order by dept_id asc,st_age desc
```

```
order by 1
```

```
select st_fname,st_age,dept_id  
from Student  
order by dept_id asc,st_age desc
```

I

```
delete from |
```

```
order by dept_id asc,st_age desc
```

```
delete from Department where dept_id=20
```

I

```
update Department set dept_id=4000 where dept_id=20
```

```
delete from Department where dept_id=20
```

```
update Department set dept_id=4000 where dept_id=20
```

order by many columns

update delete ??

Data Types

built in functions      db\_name      year

template Explorer

import and export

--builtin functions

--Agg Functions

getdate()    isnull    coalesce

concat    Convert

```
select year(getdate())
```

```
select year(getdate())
```

built-in function YEAR(expression datetime) RETURNS int

concat Convert

```
select year(getdate())
```

```
select month(getdate())
```

```
E select substring(st_fname,1,3)  
from Student
```

```
select substring(st_fname,1,3)  
from Student
```

```
select db_name()
```

```
select suser_name()
```

```
select year(getdate())
```

```
select month(getdate())
```

```
select substring(st_fname,1,3)  
from Student
```

```
select db_name()
```

```
select suser_name()
```

```
select month(getdate())
```

```
select substring(st_fname,1,3)  
from Student
```

```
select db_name()
```

```
select suser_name()
```

## Template Browser



Statistics

Stored Procedure

Synonym

Table

Add Column

Add Constraint

Add FileStream Column

Add Key

Add Memory Optimized Check Constraint

Add Memory Optimized Column

Add Memory Optimized Constraint

Add Memory Optimized Foreign Key

Add Memory Optimized Hash Index

Add Memory Optimized Key

Add Memory Optimized Non Clustered Index

Add Memory Optimized Trigger

Add MEMORY\_OPTIMIZED\_DATA Filegroup

Add System-Versioning

Create External Table

Create FileTable

Create Memory Optimized System-Versioned

Create Memory Optimized Table

Create System-Versioned Table

Create Table with FileStream Column

Create Table

Drop Column

Drop Constraint

Drop External Table

Drop Key

Drop Table

Modify Memory Optimized Column

Trigger

Object Explorer

Connect X

(SQL Server 13.0.1742.0 - DESKTOP-VF50P25\Karmi (33)) X DESKTOP-VF50P25\ITI - dbo.Student DESKTOP-VF50P25\ITI - ITI Day4.sqi - (local)...P\VF50P25\Karmi (56)

```
-- =====
-- Create table template
-- =====

USE <database, sysname, AdventureWorks>
GO
Incorrect syntax near '<'. Expecting ID, or QUOTED_ID.

IF OBJECT_ID('<schema_name, sysname, dbo>.<table_name, sysname, sample>', 'U') IS NOT NULL
    DROP TABLE <schema_name, sysname, dbo>.<table_name, sysname, sample>
GO

CREATE TABLE <schema_name, sysname, dbo>.<table_name, sysname, sample>
(
<columns_in_primary_key, , c1> <column1_datatype, , int> <column1_name, sysname, c1>
<column2_name, sysname, c2> <column2_datatype, , char(10)> <column2_name, sysname, c2>
<column3_name, sysname, c3> <column3_datatype, , datetime> <column3_name, sysname, c3>
CONSTRAINT <constraint_name, sysname, PK_sample_table> PRIMARY KEY
)
GO
```

200 % 4 Connected. (1/1) (local) (13.0 RTM) DESKTOP-VF50P25\Karmi (33) ITI 00:00:00 0 rows

Object Explorer

Connect •

(SQL Server 13.0.1742.0 - DESKTOP-VF50P25\Ram (53)) X DESKTOP-VF50P25\ITI - dbo.Student DESKTOP-VF50P25\ITI - ITM Day4.sq... (local)... P-VF50P25\Ram (58)

```
-- =====
-- Create table template
-- =====

USE <database, sysname, AdventureWorks>
GO

IF OBJECT_ID('<schema_name, sysname, dbo>.<table_name, sysname, sample_table>', 'U') IS NOT NULL
    DROP TABLE <schema_name, sysname, dbo>.<table_name, sysname, sample_table>
GO

CREATE TABLE <schema_name, sysname, dbo>.<table_name, sysname, sample_table> (
    <columns_in_primary_key, , c1> <column1_datatype, , int> <column1_name, sysname, c1>
    <column2_name, sysname, c2> <column2_datatype, , char(10)> <column2_name, sysname, c2>
    <column3_name, sysname, c3> <column3_datatype, , datetime> <column3_name, sysname, c3>
)
CONSTRAINT <constraint_name, sysname, PK_sample_table> PRIMARY KEY
GO
```

200 % 4 Connected. (1/1) (local) (13.0 RTM) DESKTOP-VF50P25\Ram (53) ITI 00:00:00 0 rows

→ 4 SQL, Aggregate Function, Grouping, Union, Subqueries, EERD

>>View Template explorer<<

>>Enhanced Entity-Relationship Diagram<<

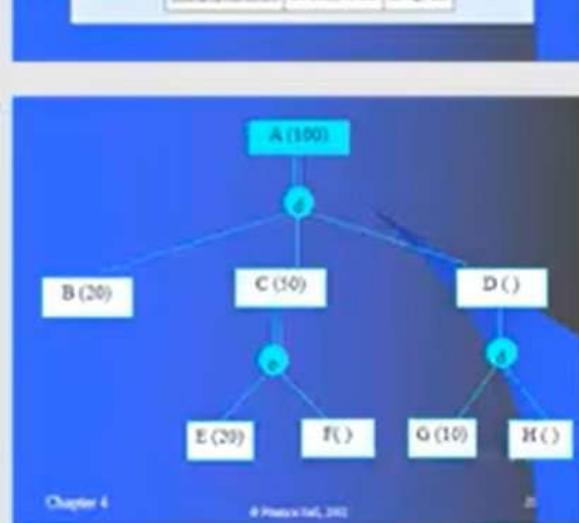
```
select month(getdate())
```

```
select substring(st_fname,1,3)  
from Student
```

```
select db_name()
```

```
select suser_name()
```

25

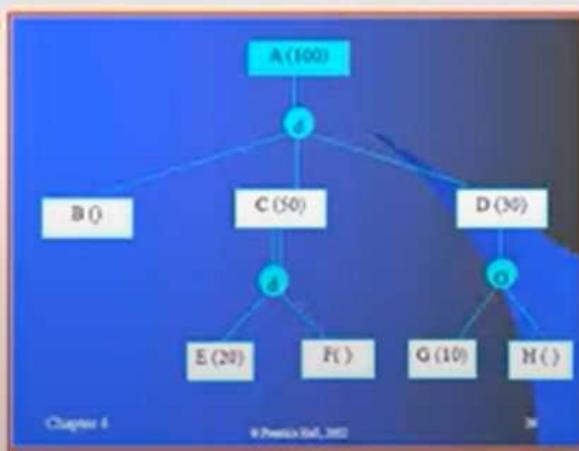


Chapter 4

© Prentice Hall, 2002

25

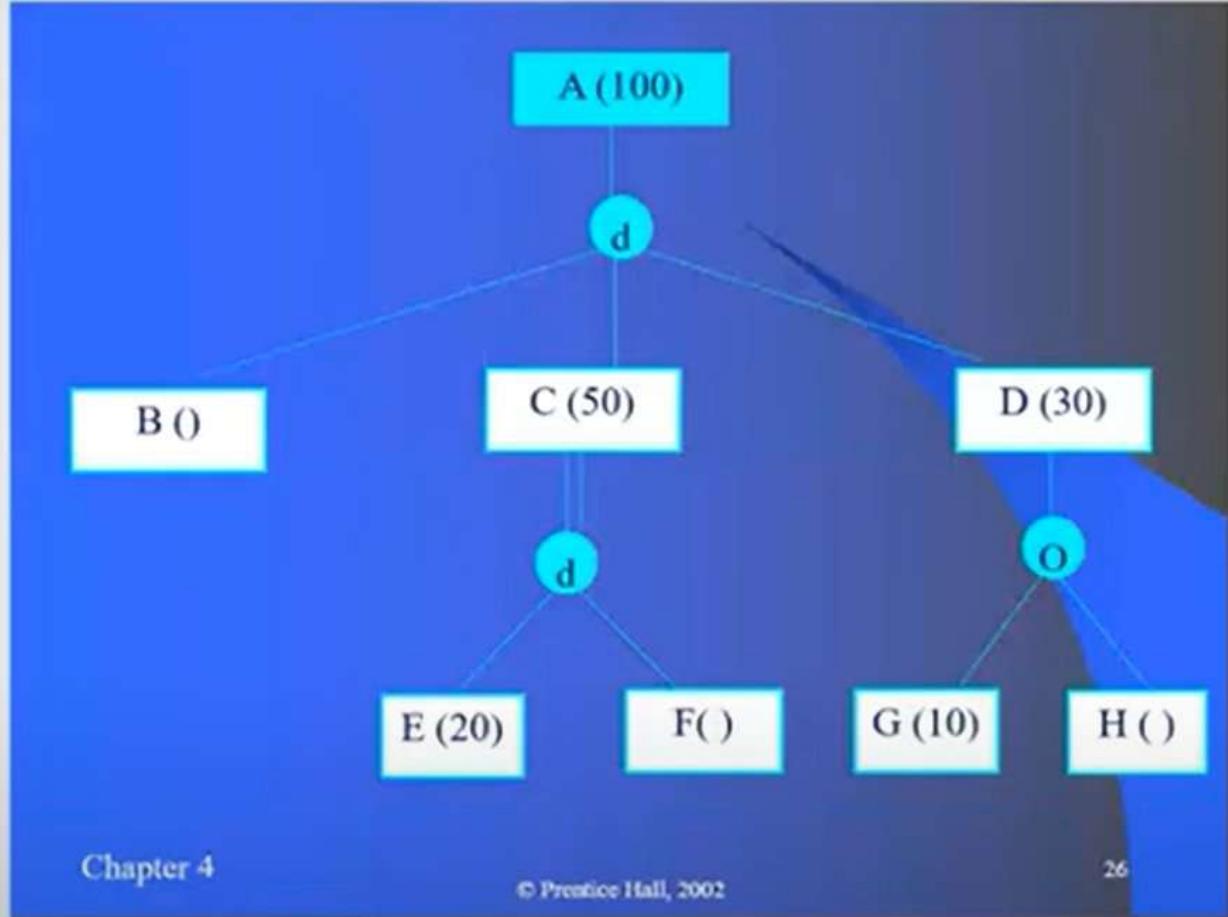
26



Chapter 4

© Prentice Hall, 2002

26



Chapter 4

© Prentice Hall, 2002

26

Click to add notes

1

Activate Window  
Review Software

order by many columns

update delete ??

Data Types

built in functions      db\_name      year

template Explorer

import and export



# Course Outline

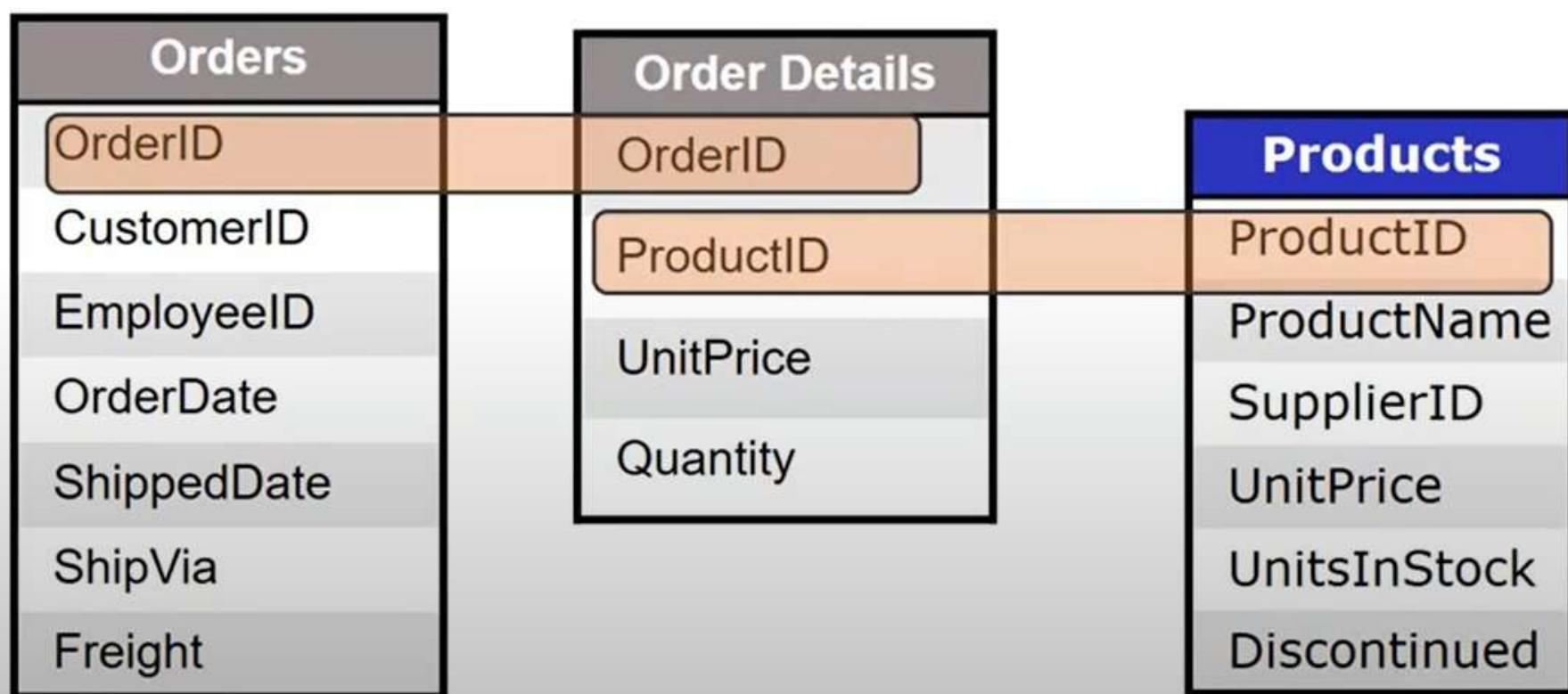


## Installation

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

# Overview of Relational Databases

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



Activate Wi  
Go to Settings t

Activ  
Go to

# SQL Server Versions History

1 <sup>st</sup> 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

2 <sup>nd</sup> 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

# 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

# SQL Server Components

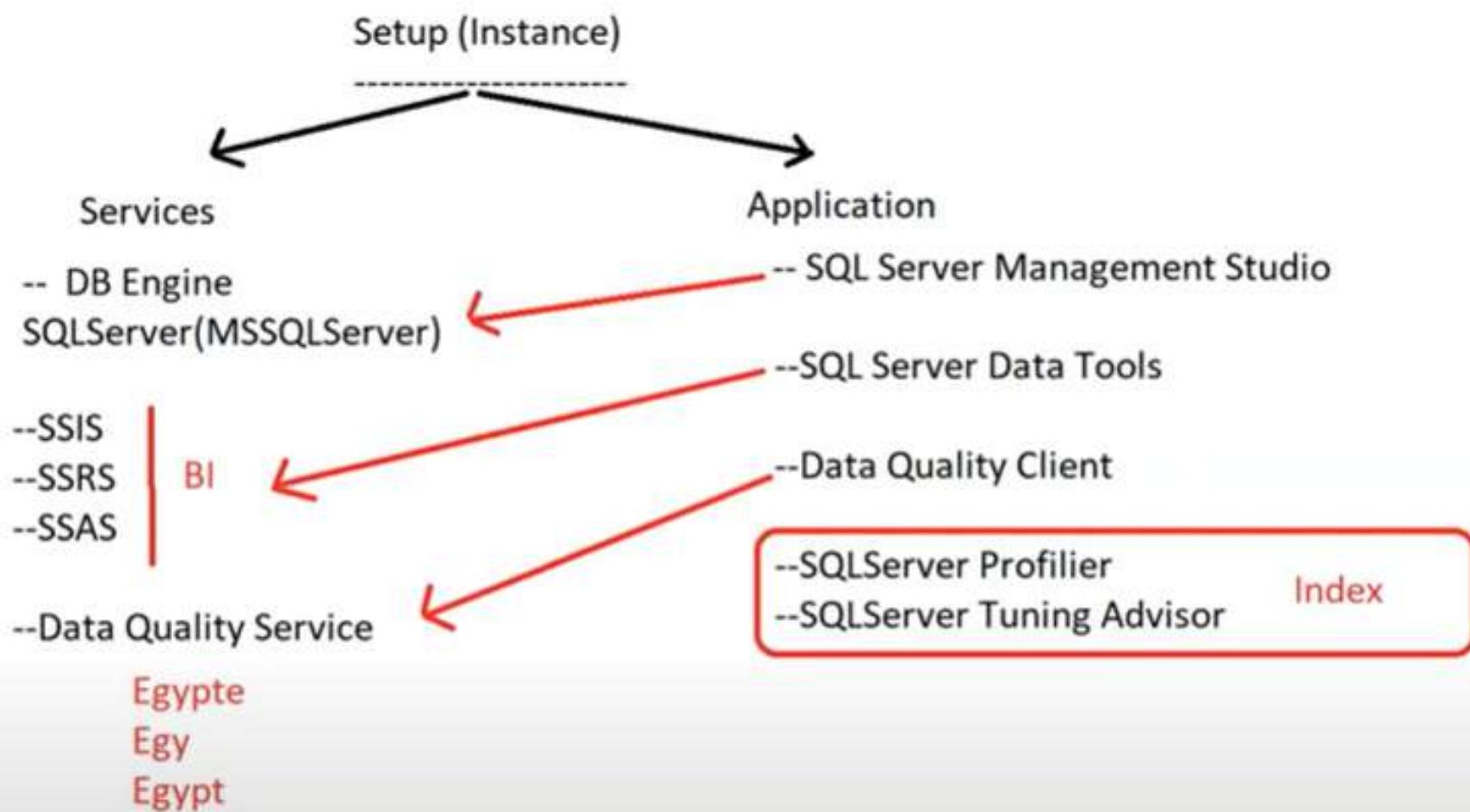
Server Component	Description
<i>SQL Server Database Engine</i>	Core service for storing and processing data
<i>Analysis Services</i>	Tools for creating and managing analytical processing
<i>Reporting Services</i>	Components for creating and deploying reports
<i>Integration Services</i>	Tools for moving,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

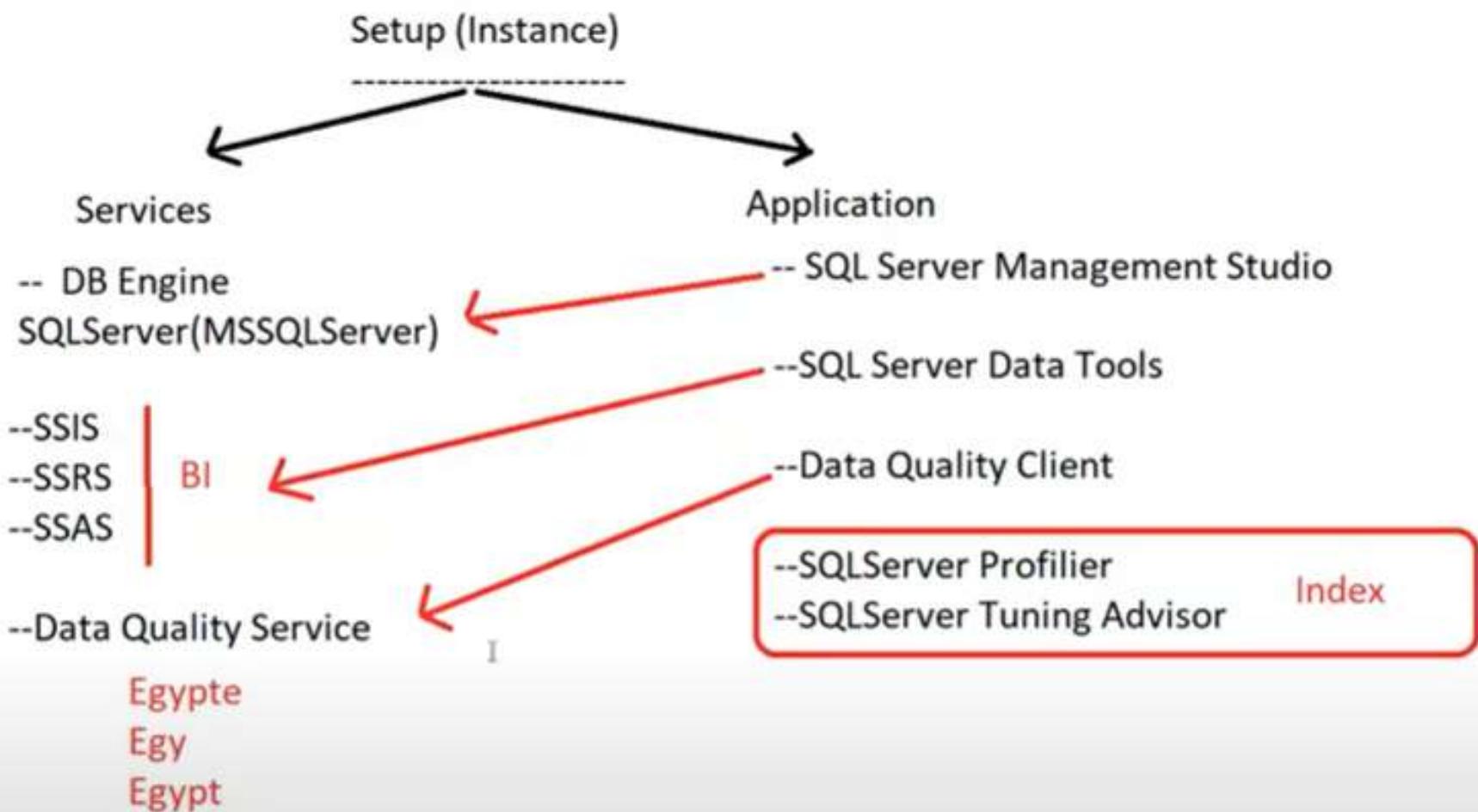
*Data Quality Service & Master Data Service*

# Course Outline

## Installation

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





## authorizarion (Permissions)

---

### authentication (UserName & Password)

---->Windows authentication

Win Admin ===> SQL Admin

---->SQLServer authentication

Dev\_ahmed 123

Dev\_ali 444

Microsoft SQL Server Management Studio

File Edit View Debug Tools Window Help

New Query

Object Explorer

Logins

- ##MS\_PolicyEventProcessingLogin##
- ##MS\_PolicyTsqlExecutionLogin##
- Ahmed
- Ali\_Manager
- Ali
- Ayman
- DESKTOP-VF50P25\Rami
- Dev\_Ali
- EB\_Stud
- emad
- khalid
- log1
- M\_Ali
- mahmoud
- MALI
- mohamed
- mohmed
- MyNew\_Dev
- mynewuser
- Nada
- NewProg
- NT AUTHORITY\SYSTEM
- NT Service\MSSQLSERVER
- NT SERVICE\ReportServer
- NT SERVICE\SQLSERVERAGENT
- NT SERVICE\SQLTELEMETRY
- NT SERVICE\SQLWriter
- NT SERVICE\Winmgmt
- Omar
- Omar1
- Omar2
- sa
- SD\_Stud
- SDLogin

Login Properties - sa

Select a page:

- General
- Server Roles
- User Mapping
- Status

Script Help

Login name: sa

Windows authentication

SQL Server authentication

Password: \*\*\*

Confirm password: \*\*\*

Specify old password

Old password: [redacted]

Enforce password policy

Enforce password expiration

User must change password at next login

Mapped to certificate: [redacted]

Mapped to asymmetric key: [redacted]

Map to Credential: [redacted] Add

Mapped Credentials:

Credential	Provider
master	

Connection:

Server: DESKTOP-VF50P25\Rami

[View connection properties](#)

Progress:

Ready

Default database: master

Default language: English

OK Cancel

Activate Windows  
Go to Settings to activate Windows.

The screenshot displays the Microsoft SQL Server Management Studio (SSMS) interface. On the left, the Object Explorer shows a list of logins, including system logins like 'sa' and 'sa' (from '##MS\_PolicyTsqlExecutionLogin##'). The main window shows the 'Login Properties' dialog for the 'sa' login. The 'General' tab is selected. The 'Login name' is set to 'sa'. The 'SQL Server authentication' radio button is selected. The 'Password' field contains '\*\*\*'. The 'Confirm password' field also contains '\*\*\*'. The 'Enforce password policy' checkbox is checked. The 'Default database' is set to 'master' and the 'Default language' is set to 'English'. The 'Connection' section shows the server as 'DESKTOP-VF50P25\Rami'. A progress bar indicates the task is 'Ready'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons. A watermark at the bottom right of the screen says 'Activate Windows Go to Settings to activate Windows.'

Book1 - Excel (Product Activation Failed)

Table Tools

File Home Insert Page Layout Formulas Data Review View Add-ins Team Design Tell me what you want to do...

Table Name: Table\_\_ITI\_Cour Summarize with PivotTable Properties Header Row First Column Filter Button  
Table\_\_ITI\_Cour Remove Duplicates Insert Slicer Export Refresh Open in Browser Total Row Last Column  
Resize Table Convert to Range External Table Data Banded Rows Banded Columns

A1

	A	B	C	D	E	F	G
1	Crs_Id	Crs_Name	Crs_Duration	Top_Id			
2	100	HTML	20	3			
3	200	C Programming	60	1			
4	300	OOP	80	1			
5	400	Unix	50	4			
6	500	SQL Server	60	2			
7	600	C#	80	1			
8	700	Web Service	20	3			
9	800	Java	60	1			
10	900	Oracle	50	2			
11	1000	ASP.Net	60	3			
12	1100	Win_XP	20	4			
13	1200	Photoshop	30	5			
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							

Microsoft Windows [Version 10.0.17763.1577]  
(c) 2018 Microsoft Corporation. All rights reserved.  
C:\Users\Rami>sqlcmd -s ".  
1> use ITI  
2> go  
Changed database context to 'ITI'.  
1> create table mytesting (id int , name varchar(20))  
2> go  
1>

Microsoft SQL Server Management Studio

Activate Windows  
Go to Settings to activate Windows.

Sheet1

Ready

100%

myITIResult.txt - Notepad

File Edit Format View Help

**Top\_Id**      **.Top\_Name**

<b>Top_Id</b>	<b>.Top_Name</b>
1	1 . Programming
2	2 . DB
3	3 . Web
4	4 . Operating System
5	5 . Design

(5 rows affected)

```
select top(3) *
from Student

select top(3) st_fname
from Student

select top(3)
from Student
where st_address='alex'
```

```
select top(3) salary  
from Instructor
```

```
select max(salary)  
from Instructor
```

```
select top(2) salary  
from Instructor  
order by salary|
```

```
from Instructor
```

```
select max(salary)  
from Instructor
```

```
select top(2) salary  
from Instructor  
order by salary desc
```

```
select top(2) salary  
from Instructor  
order by salary desc
```

```
select top(2) distinct salary  
from Instructor  
order by salary desc
```

```
select top(2) distinct salary  
from Instructor  
order by salary desc
```

```
select top(4) with ties  
from Student  
order by st_age
```

```
select top(3) with ties
from Student
order by st_age
```

```
select top(7) with ties
from Student
order by st_age
```

```
■ select top(7) with ties  
from Student  
order by st_age
```

```
select newid() --GUID
```

```
■ select *, newid()  
from Student
```

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

```
select st_fname+' '+st_lname as fullname  
from Student  
where fullname='ahmed ali'
```

```
--execution order  
--from  
--join    i  
--on  
--where  
--group  
--having  
--select
```

```
--from  
--join  
--on  
--where I  
--group  
--having  
--select  
--order by  
--top
```

```
--join  
--on  
--where  
--group by  
--having  
--select  
--order by  
--top
```

--join

--on

--where

--group|  
--having [agg]

--select [distinct +agg]

--order by

--top

```
select st_fname+' '+st_lname as fullname  
from Student  
where st_fname+' '+st_lname='ahmed ali'
```

```
select *  
from (select st_fname+' '+st_lname as fullname  
      from Student) as Newtable  
where fullname='ahmed ali'
```

```
-| select *  
  from (select st_fname+' '+st_lname as fullname  
        from Student) as Newtable  
where fullname='ahmed ali'
```

-| -----execution order

--from

--join

```
--DB Objects [table    view      function   SP     Rule]
--[ServerName].[DBName].[schemaName].[objectName]
```

```
[ -] select *
  from Student
```

I

```
select *
  from Student
```

```
- select *
```

```
from Student
```

```
- select *
```

```
from [DESKTOP-VF50P25].iti.dbo.Student
```

```
select |
```

```
select *  
from Company_SD.dbo.Project
```

```
select dname  
from Company_SD.dbo.Departments  
union all  
select dept_name  
from Department
```

--DDL

select \* into table2  
from Student

select \* from table2

select \* into table2  
from Student

--DDL

- select \* into table2  
from Student

select \* from table2

- select \* into table2  
from Student

```
[-] select * into company_sd.dbo.student  
      from Student
```

```
[-] select st_id,st_fname into tab3  
      from Student  
      where st_address='alex'
```

```
- select * into tab4  
from Student  
where 1=2
```

```
- select * into tab4  
from Student  
where st_age<-1000
```

```
select * into tab4  
from Student  
where 1=2
```

```
select * into tab4  
from Student  
where st_age<-1000
```

```
insert into tab3  
values(66, 'ali')
```

```
--insert based in select
```

```
insert into tab3  
select st_id, st_fname from student
```

```
insert into tab3  
values(66,'ali')
```

--insert based in select

```
insert into tab3  
select st_id,st_fname from student
```

```
[-]insert into tab3  
      select st_id,st_fname from student
```

```
[-]select  
      from  
      having
```

```
insert into tab3  
select st_id,st_fname from student  
  
select sum(salary)  
from Instructor  
having count(ins_id)>100
```

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

```
 select Sum(salary)
from Instructor
```

```
- select Sum(salary)
  from Instructor
 having count(ins_id)<100
```

```
- select ins_name
  from Instructor
 having count(ins_id)<100
```

```
- select Sum(salary)
  from Instructor
 having count(ins_id)<100
```

```
- select Sum(salary)
  from Instructor
 where count(ins_id)>100      ---error
```

## Ranking Functions

-----  
Row\_Number()

Dense\_rank()

NTiles(Group)

Rank()

Select \*  
From (  
    Select \*, Row\_Number() over(order by esal desc) as RN  
                , Dense\_rank() over(order by esal desc) as DR  
                , NTile(3) over(order by esal desc) as G  
    From employee ) as Newtable

where RN=1

DR=1

RN=3

DR<=2

RN<=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

Activate Windows

## Ranking Functions

-----  
Row\_Number()

Dense\_rank()

NTiles(Group)

Rank()

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

DR=1

RN=3

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

Activate Windows

## Ranking Functions

-----  
Row\_Number()

Dense\_rank()

NTiles(Group)

Rank()

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

DR=1

RN=3

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

Activate Windows

Get a 1-year Microsoft Office 365 Home Premium for \$100

## Ranking Functions

-----  
Row\_Number()

Dense\_rank()

NTiles(Group)

Rank()

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

DR=1

RN=3

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	12
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

Activate Windows

Go to [www.microsoft.com/activatenow](#)

```
select Sum(salary)  
from Instructor  
where count(ins_id)>100      ---error
```

```
[-] select *,Row_number() over(order by st_Age desc) as RN  
      from Student
```

```
[-] select *,Dense_rank() over(order by st_Age desc) as DR  
      from Student
```

```
- select *,Row_number() over(order by st_Age desc) as RN  
from Student
```

```
- select *,Dense_rank() over(order by st_Age desc) as DR  
from Student  I
```

```
select *,Row_number() over(order by st_Age desc) as RN  
from Student  
where RN=1
```

```
select *,Dense_rank() over(order by st_Age desc) as DR  
from Student  
where DR=1
```

```
    Select *  
    from (  
        select *,Row_number() over(order by st_Age desc) as RN  
        from Student) as Newtable  
    where RN=1
```

```
    select *
```

```
- Select *
from (
    select *,Row_number() over(order by st_Age desc) as RN
    from Student) as Newtable
where RN=1

- select *
from (
    select *,Dense_rank() over(order by st_Age desc) as DR
    from Student) AS NEWTABLE
where DR=1
```

```
    where RN=1

select *
from (
    select *,Dense_rank() over(partition by dept_id order by st_Age desc)
        from Student) AS NEWTABLE
```

```
select *  
from (  
    select *,Dense_rank() over(partition by dept_id order by st_Age desc)  
        from Student) AS NEWTABLE  
where DR=1
```

```
select *  
from  
  (select *,Ntile(4) over(order by st_age desc) as G  
   from Student) as Newtable
```

```
select *
from
  (select *,Ntile(4) over(order by st_age desc) as G
   from Student) as Newtable
where G=1
```

- - - - - Data types
  - - - - - Numeric DT
  - - - - - Decimal DT
  - - - - - Char DT
  - - - - - DateTime
  - - - - - binary DT

-----Numeric DT

- bit** --bool 0:1 true:false
- tinyint** 1 Byte -128:+127 unsigned 0:255
- smallint** 2B -32768:+32767 unsigned 0:65555
- int** 4B
- bigint** 8B

-----Decimal DT

- smallmoney**
- money**



### -----Char DT

- char(10) [fixed length character] ahmed 10 ali 10 على ???
- varchar(10) [variable length character] ahmed 5 ali 3
- nchar(10) unicode على على
- nvarchar(10)
- nvarchar(max) up to 2GB

### -----DateTime

- Date
- Time
- time(7)
- sma

-----DateTime

Date MM/DD/yyyy

Time hh:mm:12.765

time(7) hh:mm:12.7659876

smalldatetime MM/DD/yyyy hh:mm:00

datetime MM/DD/yyyy hh:mm:ss.987

datetime2(7) MM/DD/yyyy hh:mm:ss.9879876

datetimeoffset 11/24/2020 10:30 +2:00 Timezone

-----binary DT

binary 0111100 11111100

image

**datetimeoffset** 11/24/2020 10:30 +2:00 Timezone

-----binary DT

**binary** 0111100 11111100

**image**

-----others

**XML**

**unique\_identifier**

**sql\_variant**



--DB Engine

## sql\_variant

--DB Engine

--types of instances

types odf authentication

--Top select into

--Ranking

--datatype

```
select ins_name, salary  
from Instructor
```

```
- select ins_name, salary,  
      case  
        when salary >= 3000 then 'high sal'  
        when salary < 3000 then 'low'  
        else 'No value'  
      end as Newsal  
from Instructor
```

```
when salary>=3000 then 'high sal'  
when salary<3000 then 'low'  
else 'No value'  
end as Newsal
```

```
from Instructor
```

```
update Instructor  
set salary=salary*1.20
```

```
select ins_name,salary,
       case
           when salary>=3000 then 'high sal'
           when salary<3000 then 'low'
           else 'No value'
       end as Newsal
from Instructor
```

```
select ins_name,iif(salary>=3000,'high','low')
from Instructor
```

```
case  
when salary >= 3000 then salary * 1.10  
else salary * 1.20  
end
```

---

```
select convert(varchar(20), getdate())
```

```
select cast(getdate() as varchar(20))
```

```
select convert(varchar(20),getdate())
```

```
select cast(getdate() as varchar(20))
```

```
select convert(varchar(20),getdate(),102)
```

```
select convert(varchar(20),getdate(),103)
```

```
select convert(varchar(20),getdate(),104)
```

```
select convert(varchar(20),getdate(),105)
```

```
select format(getdate(), '')
```

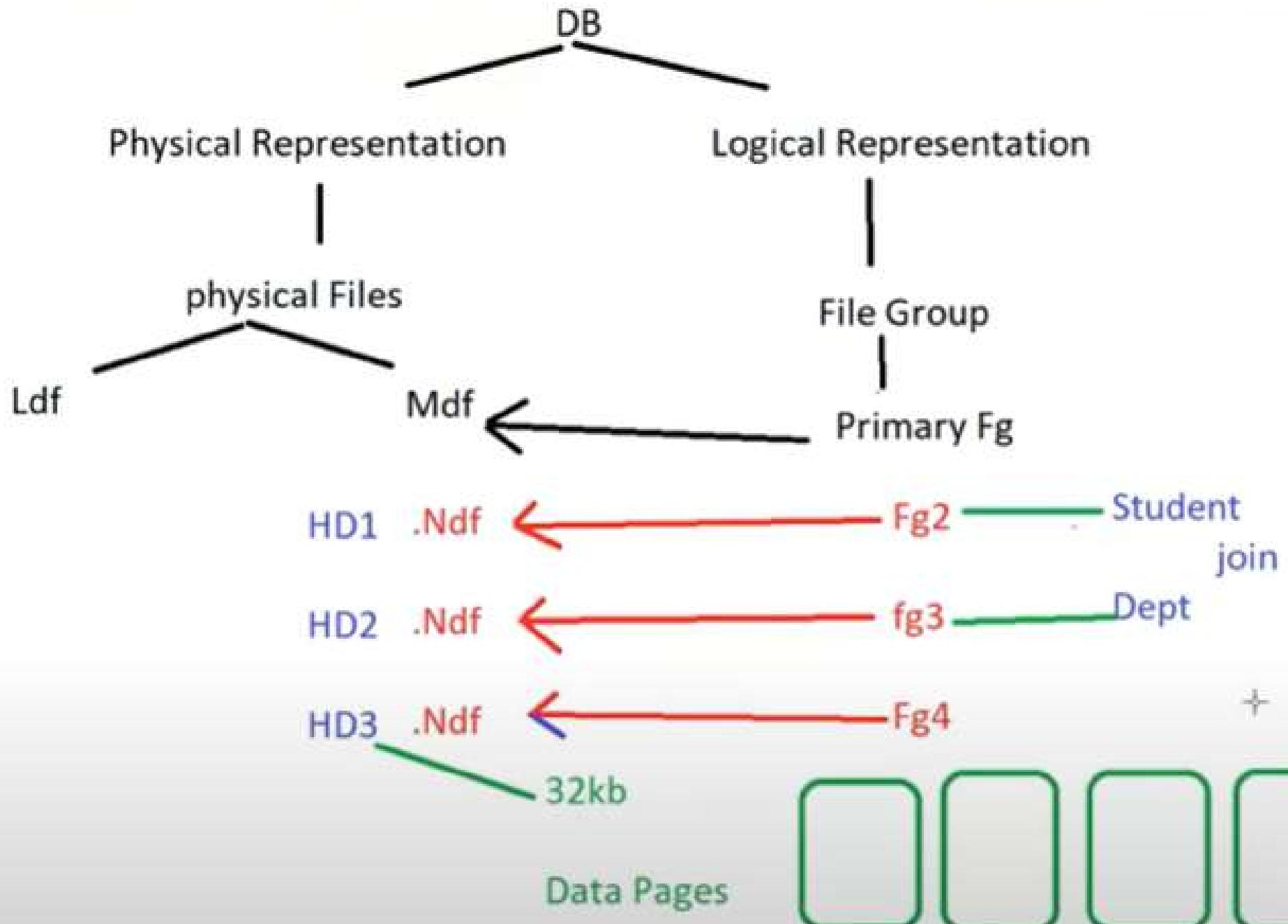
```
select format(getdate(), 'dd-MM-yyyy')
select format(getdate(), 'dddd MMMM yyyy')
select format(getdate(), 'ddd MMM yy')
select format(getdate(), 'ddddd')
select format(getdate(), 'MMMM')
select format(getdate(), 'hh:mm:ss')
select format(getdate(), 'HH')
select format(getdate(), 'hh tt')
select format(getdate(), 'dd-MM-yyyy hh:mm:ss tt')
```

```
select format(getdate(), 'dd')  
select day(getdate())
```

```
I  
select eomonth(getdate())
```

```
select format(eomonth(getdate()), 'dddd')
```

```
| select format(eomonth(getdate()), 'dd' )
```



Connect - (SQL Server 13.0.1742.0 - DESKTOP-VF50P25)\Ran

- Databases
  - System Databases
  - Database Snapshots
  - AdventureWorks2012
  - AdventureWorksDW2012
  - Company\_SD
  - ITI
  - MenofyaDB
  - MyNewDB
  - New2
  - NewDB
  - ReportServer
  - ReportServerTempDB
  - sales
  - SD
  - sadasdas
  - Sohag2
  - SohagDB
  - test2
  - DB1
- Database Diagrams
- Tables
  - System Tables
  - FileTables
  - External Tables
- Views
- External Resources
- Synonyms
- Programmability
- Service Broker
- Storage
- Security

Output

Column Name	Data Type	Allow Nulls
<b>id</b>	int	<input type="checkbox"/>
name	varchar(50)	<input type="checkbox"/>
<b>age</b>	int	<input checked="" type="checkbox"/>
email	varchar(50)	<input checked="" type="checkbox"/>
hiredate	date	<input checked="" type="checkbox"/>
sal	int	<input checked="" type="checkbox"/>
overtime	int	<input checked="" type="checkbox"/>
Netsal		<input type="checkbox"/>
Dnum	int	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Properties

DTS-published	No
Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	No
Indexable	Yes
Is Columnset	No
<b>Is Sparse</b>	<b>Yes</b>
Merge-published	No
Not For Replication	No
Replicated	No
RowGuid	No
Size	4
<b>Is Sparse</b>	

[Tbl] dbo.Employee

(Name)	Employee
Database Name	DB1
Description	
Schema	dbo
Server Name	desktop-vf50p25

Table Designer

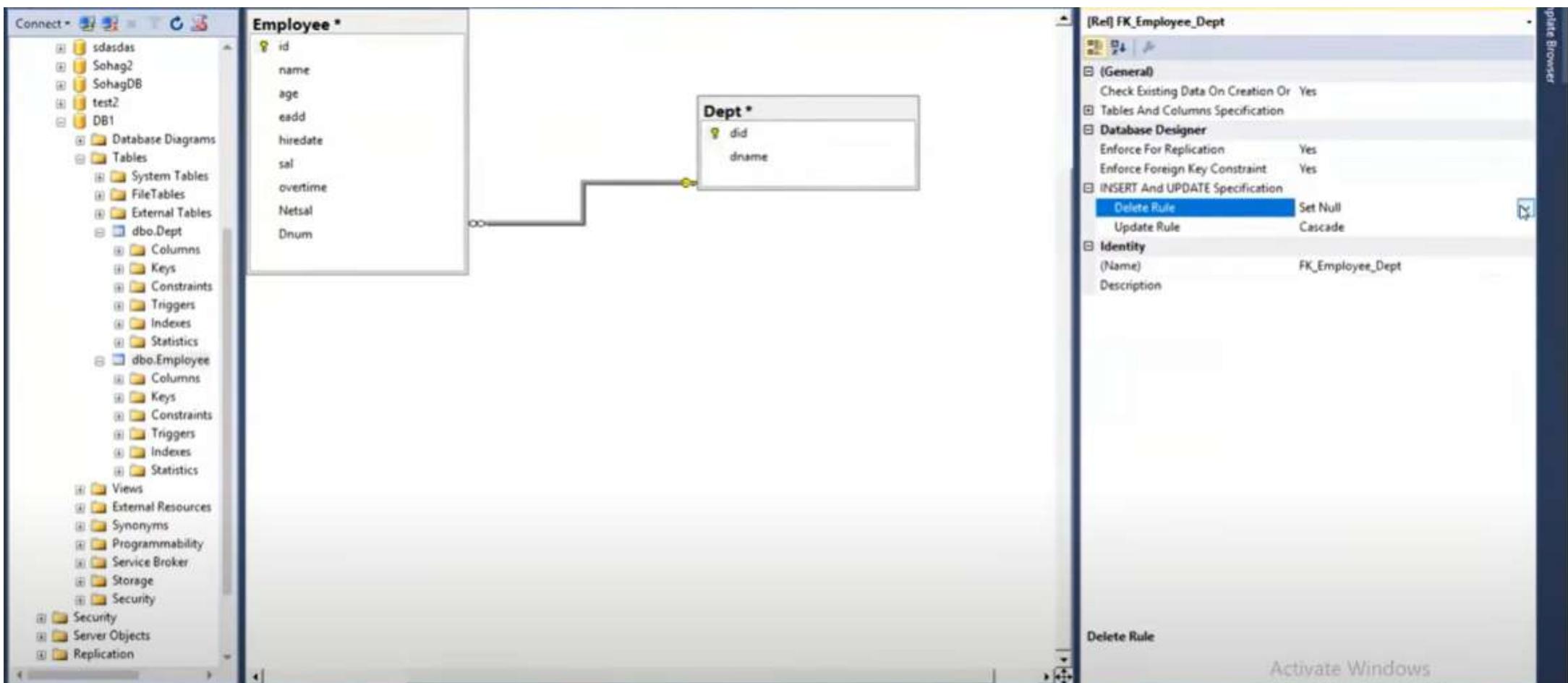
Identity Column	<b>id</b>
Indexable	Yes
Lock Escalation	Table

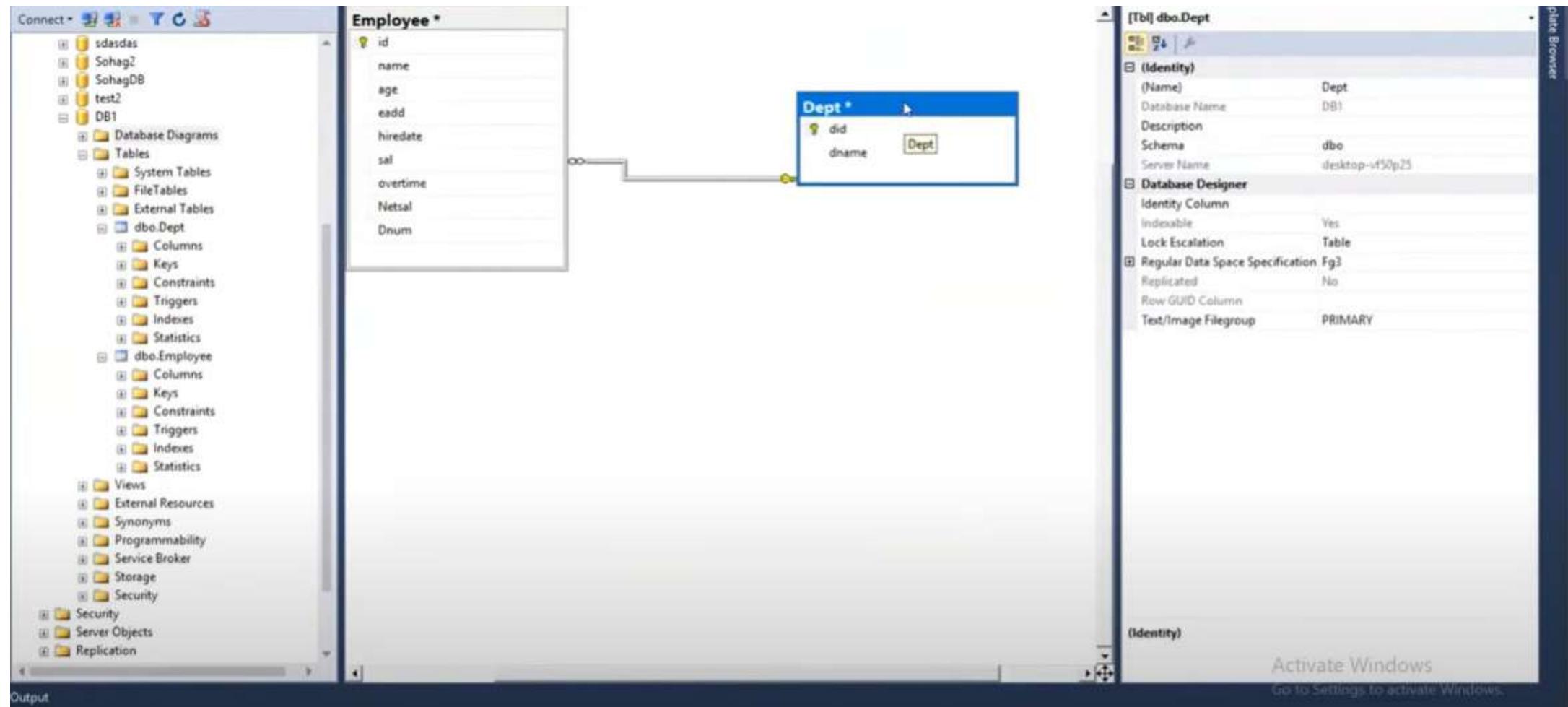
Regular Data Space Specification Fg2

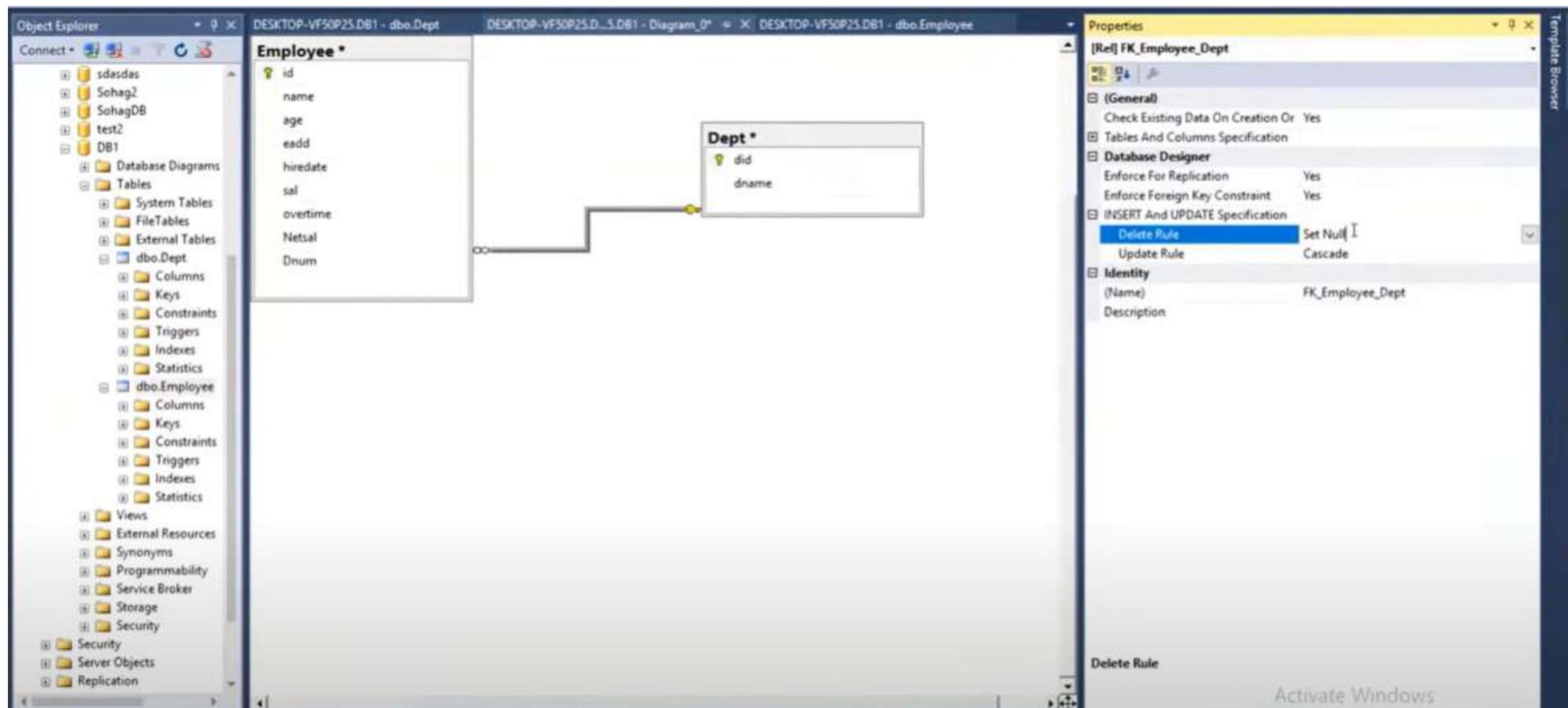
(Data Space Type)	Filegroup
Filegroup or Partition Scheme	Fg2
Partition Column List	
Replicated	No
Row GUID Column	
Text/Image Filegroup	PRIMARY

Filegroup or Partition Scheme Name

Activate Windows  
Go to Settings to activate Windows.







Connect +

sdasdas

Sohag2

SohagDB

test2

DB1

- Database Diagrams
- Tables
  - System Tables
  - FileTables
  - External Tables
  - dbo.Dept
    - Columns
    - Keys
    - Constraints
    - Triggers
    - Indexes
    - Statistics
  - dbo.Employee
    - Columns
    - Keys
    - Constraints
    - Triggers
    - Indexes
    - Statistics
- Views
- External Resources
- Synonyms
- Programmability
- Service Broker
- Storage
- Security
- Security
- Server Objects
- Replication

Column Name      Data Type      Allow Nulls

Column Name	Data Type	Allow Nulls
<b>Dnum</b>	int	<input checked="" type="checkbox"/>
id	int	<input type="checkbox"/>
name	varchar(50)	<input type="checkbox"/>
age	int	<input checked="" type="checkbox"/>
eadd	varchar(50)	<input checked="" type="checkbox"/>
hiredate	date	<input checked="" type="checkbox"/>
sal	int	<input checked="" type="checkbox"/>
overtime	int	<input checked="" type="checkbox"/>
Netsal		<input checked="" type="checkbox"/>

Column Properties

(General)

(Name)	Dnum
Allow Nulls	Yes
Data Type	int
Default Value or Binding	<input type="text"/>

Table Designer

Identity Column	<b>Dnum</b>
Indexable	Yes
Lock Escalation	Table
Regular Data Space Specification	Fg2
Replicated	No
Row GUID Column	
Text/Image Filegroup	PRIMARY

[Tbl] dbo.Employee

(Identity)

Name	Employee
Database Name	DB1
Description	
Schema	dbo
Server Name	desktop-vf50p25

(Table Designer)

Identity Column	<b>Dnum</b>
Indexable	Yes
Lock Escalation	Table
Regular Data Space Specification	Fg2
Replicated	No
Row GUID Column	
Text/Image Filegroup	PRIMARY

Activate Windows

SchemaName.ObjectName

select \*  
from dbo.Student

create schema HR

```
select *  
from dbo.Student
```

```
create schema HR
```

```
create schema sales
```

```
alter schema HR transfer Student
```

```
alter schema HR transfer Instructor
```

```
alter schema sales transfer department
```

```
create table student
```

```
(
```

```
)
```

```
alter schema sales transfer department
```

```
create table student
```

```
(  
    id int,  
    name varchar(20)
```

```
P  
create table sales.student  
(  
    id int,  
    name varchar(20)  
)  
I  
select * from in
```

```
select * from Instructor
```

```
select * from hr.Instructor
```

```
select * from Student
```

```
select * from Hr.Student
```

```
select * from Instructor
```

```
select * from hr.Instructor
```

```
select * from Student
```

```
select * from Hr.Student
```

```
select * from Instructor
```

```
select * from hr.Instructor
```

```
select * from Student
```

```
select * from Hr.Student
```

**create schema HR**

**create schema sales**

**alter schema HR transfer Student**

**alter schema HR transfer Instructor**

**alter schema sales transfer department**

**create schema HR**

**create schema sales**

**alter schema HR transfer Student**

**alter schema HR transfer Instructor**

**alter schema sales transfer department**

```
- select * from hr.Student
```

```
- insert into hr.Student(st_id,st_fname)  
values(666,'ali')
```

```
- update hr.Student  
    set st_age+=1
```

**create schema HR**

**create schema sales**

**alter schema HR transfer Student**

**alter schema HR transfer Instructor**

**alter schema sales transfer department**

```
select * from Student
```

```
select * from Hr.Student
```

```
use AdventureWorks2012
```

```
select * from HumanResources.EmployeeDepartmentHistory
```

```
create synonym HE  
for HumanResources.EmployeeDepartmentHistory
```

```
select * from HE
```

```
drop table course
```

**drop table course**

---data & Metadata

**delete from course**

----data

**truncate table course**

----data

I

```
create table test1  
(  
    id int identity,  
    name varchar(20)  
)  
  
insert into test1 values('ali')  
  
select * from |
```

```
name varchar(20)
)

insert into test1 values('omar')

select * from test1

delete from test1
```

```
insert into test1 values('hassan')
```

```
select * from test1
```

```
delete from test1
```

```
truncate table test1
```

```
select * from test1
```

```
delete from test1
```

```
truncate table test1
```

DB Integrity				
	Column	Domain Integrity Range of Values	Entity Integrity Uniqueness (Rows)	Refrential Integrity Relationship
DB Constraints	-- Data type -- Default 'cairo' --Allow NULL Not NULL --Check Constraint	--PK Constraint -Unique Constraint Allow One Null Value	-- Foreign key Constraint	
DB Objects	Rules Triggers	Index Triggers		Custom Constraint SP - Trigger Triggers

Activate Windows  
Go to Settings to activate Windows

DB Integrity			
	Column		
	Domain Integrity	Entity Integrity	Refrential Integrity
	Range of Values	Uniqueness (Rows)	Relationship
DB Constraints	-- Data type -- Default 'cairo' --Allow NULL Not NULL --Check Constraint	--PK Constraint  -Unique Constraint Allow One Null Value	-- Foreign key Constraint
DB Objects	Rules  Triggers	Index  Triggers	Custom Constraint SP - Trigger  Triggers

Activate Windows

```
- create table Dept  
(  
    Dept_id int primary key,  
    dname varchar(20)  
)
```

```
create table emp
```

(

```
create table emp  
(  
    eid int ,  
    ename varchar(20),  
    eadd varchar(20),  
    hiredate date,  
    sal int,  
    overtime int,  
    netsal int,  
    BD data,  
    ah|
```

```
(  
    eid int ,  
    ename varchar(20) ,  
    eadd varchar(20) ,  
    hiredate date ,  
    sal int ,  
    overtime int ,  
    netsal int ,  
    BD data ,  
    age int ,  
    gender int ,
```

```
ename varchar(20),  
eadd varchar(20),  
hiredate date,  
sal int,  
overtime int,  
netsal int,  
BD date,  
age int,  
gender int,  
hour_rate int,  
did int
```

```
create table emp
(
    eid int primary key identity(1,1),
    ename varchar(20),
    eadd varchar(20),
    hiredate date,
    sal int,
    overtime int,
    netsal int,
    BD date,
    age int,
    gender int,
    hour_rate int,
    did int
```

```
\ eid int primary key identity(1,1),
ename varchar(20),
eadd varchar(20) default 'alex',
hiredate date default getdate(),
sal int,
overtime int,
netsal as(isnull(sal,0)+isnull(overtime,0)) persisted,
BD date,
age as(year(getdate())-year(BD)),
gender int,
hour_rate int not null,
did int
```

```
create table emp
(
    eid int primary key identity(1,1),
    ename varchar(20),
    eadd varchar(20) default 'alex',
    hiredate date default getdate(),
    sal int,
    overtime int,
    netsal as(isnull(sal,0)+isnull(overtime,0)) persisted,
    BD date,
    age as(year(getdate())-year(BD)),
    gender int,
    hour_rate int not null,
    did int
```

```
cau voi chia (20) default area ,  
hiredate date default getdate(),  
sal int,  
overtime int,  
netsal as(isnull(sal,0)+isnull(overtime,0)) persisted,  
BD date,  
age as(year(getdate())-year(BD)),  
gender int,  
hour_rate int not null,  
did int  
)
```

```
eadd varchar(20) default 'alex',
hiredate date default getdate(),
sal int,
overtime int,
netsal as(isnull(sal,0)+isnull(overtime,0)) persisted,
BD date,
age as(year(getdate())-year(BD)),
gender int,
hour_rate int not null,
did int,
constraint c1 primary key(eid,ename),
)
[
```

```
overtime int,  
netsal as(isnull(sal,0)+isnull(overtime,0)) persisted,  
BD date,  
age as(year(getdate())-year(BD)),  
gender int,  
hour_rate int not null,  
did int,  
constraint c1 primary key(eid,ename),  
constraint c2 unique(sal),  
constraint c3 unique(overtime),  
constraint c4 check(sal>1000),  
constraint c5 check(eadd in ('cairo','mansoura','alex'))|  
)
```

```
BD date,  
age as(year(getdate())-year(BD)),  
gender varchar(1),  
hour_rate int not null,  
did int,  
constraint c1 primary key(eid,ename),  
constraint c2 unique(sal),  
constraint c3 unique(overtime),  
constraint c4 check(sal>1000),  
constraint c5 check(eadd in ('cairo','mansoura','alex')),  
constraint c6 check(gender='F' or gender='M'),  
constraint c7 check(overtime between 100 and 500),  
constraint c8 |  
)
```

```
age as(year(getdate())-year(BD)),  
gender varchar(1),  
hour_rate int not null,  
did int,  
constraint c1 primary key(eid,ename),  
constraint c2 unique(sal),  
constraint c3 unique(overtime),  
constraint c4 check(sal>1000),  
constraint c5 check(eadd in ('cairo','mansoura','alex')),  
constraint c6 check(gender='F' or gender='M'),  
constraint c7 check(overtime between 100 and 500),  
constraint c8 foreign key(did) references dept(dept_id)
```

```
create table emp
(
    eid int identity(1,1),
    ename varchar(20),
    eadd varchar(20) default 'alex',
    hiredate date default getdate(),
    sal int,
    overtime int,
    netsal as(isnull(sal,0)+isnull(overtime,0)) persisted,
    BD date,
    age as(year(getdate())-year(BD)),
    gender varchar(1),
    hour_rate int not null,
    did int,
```

Object Explorer

Connect ▾

- dbo.Demo
- dbo
- New Table...
- Design
- Select Top 1000 Rows
- Edit Top 200 Rows**
- Script Table as
- View Dependencies
- Memory Optimization Advisor
- Encrypt Columns...
- Full-Text index
- Storage
- Stretch
- Policies
- Facets
- Start PowerShell
- Reports
- Rename
- Delete
- Refresh
- Properties
- Triggers
- Indexes
- Statistics

Day6.sql - (local)...P-VF50P25\Ram (55)\*

```
create table emp
(
    eid int identity(1,1),
    ename varchar(20),
    eadd varchar(20) default 'alex',
    hiredate date default getdate(),
    sal int,
    overtime int,
    netsal as(isnull(sal,0)+isnull(overtime,0)) persisted,
    BD date,
    age as(year(getdate())-year(BD)),
    gender varchar(1),
    hour_rate int not null,
    did int,
    . . .
)
```

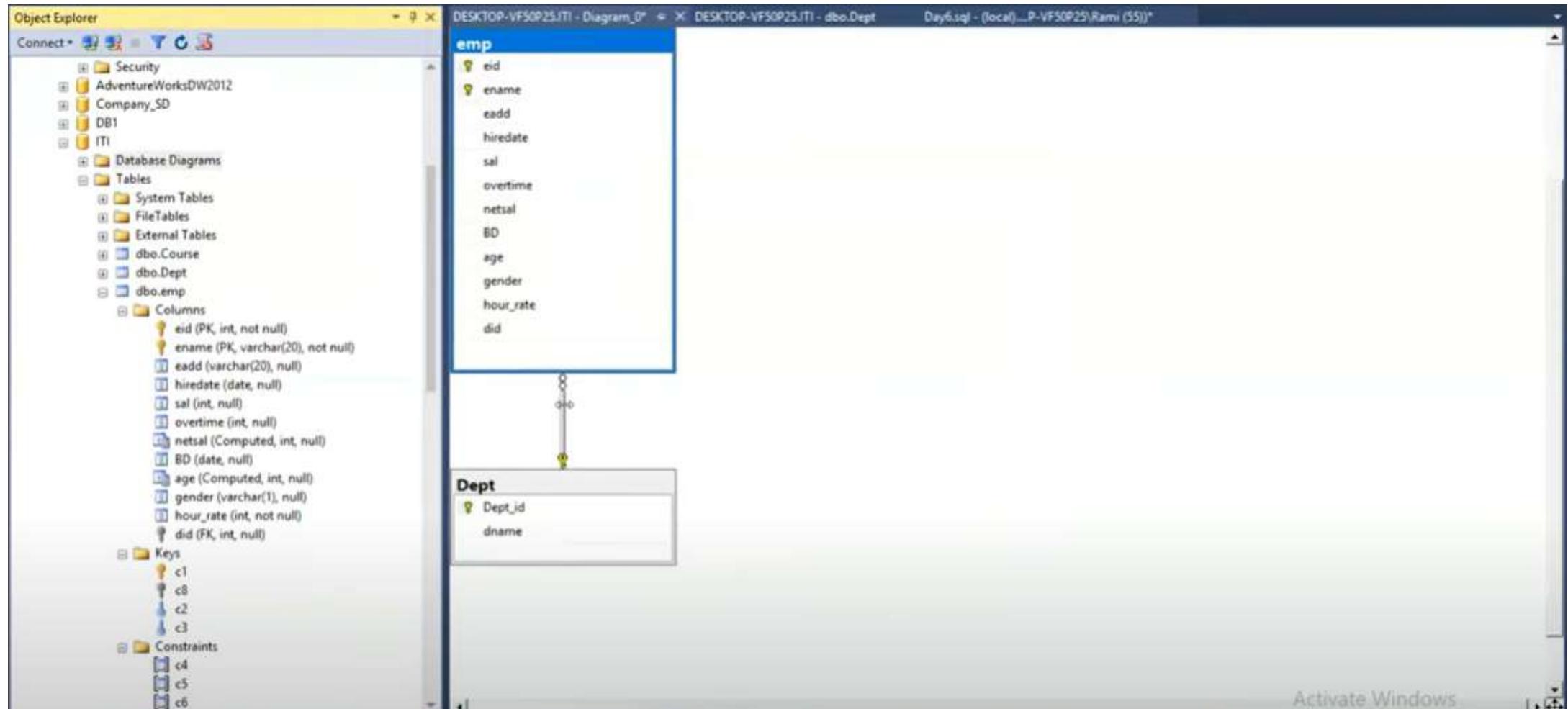
Messages

Command(s) completed successfully.

200 %

Query executed successfully.

(local) (13.0 RTM) DESKTOP-VF50P25\Ram (55) ITI 00:00:00 0 rows



Activate Windows

Day6.sql - (local)\ITI (DESKTOP-VF50P25\Ram (55))\* - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Generic Debugger

Object Explorer

Day6.sql - (local)\...P-VF50P25\Ram (55)\*

```
hour_rate int not null,  
did int,  
constraint c1 primary key(eid,ename),  
constraint c2 unique(sal),  
constraint c3 unique(overtime),  
constraint c4 check(sal>1000),  
constraint c5 check(eadd in ('cairo','mansoura','alex')),  
constraint c6 check(gender='F' or gender='M'),  
constraint c7 check(overtime between 100 and 500),  
constraint c8 foreign key(did) references dept(dept_id)  
    on delete set NULL on update cascade  
)  
  
alter table emp add constraint c100 check(hour_rate )
```

Messages

Command(s) completed successfully.

Query executed successfully.

(local) (13.0 RTM) DESKTOP-VF50P25\Ram (55) ITI 00:00:00 0 rows

```
constraint c4 check(sal>1000),
constraint c5 check(eadd in ('cairo','mansoura','alex')),
constraint c6 check(gender='F' or gender='M'),
constraint c7 check(overtime between 100 and 500),
constraint c8 foreign key(did) references dept(dept_id)
    on delete set NULL on update cascade
)
```

```
alter table emp add constraint c100 check(hour_rate>100)
```

```
alter table emp add constraint c100 check(hour_rate>100)
```

```
alter table emp drop constraint c3
```

--Constraint ---> New Data

--Constraint --->shared

--Datatype              Constraint      Default

--Rule

I

```
alter table emp drop constraint c3
```

--Constraint ---> New Data

--Constraint --->shared

--Datatype Constraint Default

```
alter table instructor add constraint c200 check(salary>1000)
```

--Rule

create rule r1 as @x>1000

--Rule

create rule r1 as @x>1000

sp\_bindrule r1, 'instructor.salary'

sp\_bindrule r1, 'emp.overtime'

sp\_unbindrule 'instructor.salary'

sp\_unbindrule 'emp.overtime'

drop rule r1

```
create default def1 as 5000
```

```
- sp_bindefault def1,'instructor.salary'
```

```
sp_unbindefault 'instructor.salary'
```

```
drop rule r1
```

```
create default def1 as 5000
sp_bindefault def1,'instructor.salary'
sp_unbindefault 'instructor.salary'
drop default def1
--Create Datatype      ComplexDT  (int      >1000      default 5000)
```

```
--Create Datatype    ComplexDT  (int      >1000      default 5000)
create rule r1 as @x>1000
create default def1 as 5000

sp_addtype ComplexDT,'int'

sp_bindrule r1,ComplexDT
I

sp_bindefault def1,ComplexDT

create table |
```

```
└─ sp addtype ComplexDT, 'int'
```

```
sp_bindrule r1,ComplexDT
```

```
sp_bindefault def1,ComplexDT
```

```
└─ create table test3
```

```
(  
    id int,  
    name varchar(20),  
    salary ComplexDT
```

```
--Create Datatype      ComplexDT  (int      >1000      default 5000)
create rule r1 as @x>1000
create default def1 as 5000
```

```
- sp_addtype ComplexDT,'int'
```

```
    sp_bindrule r1,ComplexDT
```

```
    sp_bindefault def1,ComplexDT
```

```
- create table test3
```

```
  (
    id int,
```

```
did int,  
| constraint c1 primary key(eid,ename),  
constraint c2 unique(sal),  
| constraint c3 unique(overtime),  
constraint c4 check(sal>1000),  
constraint c5 check(eadd in ('cairo','mansoura','alex')),  
constraint c6 check(gender='F' or gender='M'),  
constraint c7 check(overtime between 100 and 500),  
constraint c8 foreign key(did) references dept(dept_id)  
    on delete set NULL on update cascade  
)
```

```
alter table emp add constraint c100 check(hour_rate>100)
```

I

```
alter table emp drop constraint c3
```

--XXXXX	Constraint	---> New Data
--XXXXX	Constraint	--->shared
--XXXXX	Datatype	Constraint Default

```
alter table instructor add constraint c200 check(salary>1000)
```

```
--Create Datatype      ComplexDT  (int      >1000      default 5000)
create rule r1 as @x>1000
create default def1 as 5000
sp_addtype ComplexDT,'int'
sp_bindrule r1,ComplexDT
sp_bindefault def1,ComplexDT
create table test3
(
```

```
--Create New Datatype ----ComplexDT (int >1000 default 5000)
create rule r1 as @x>1000
create default def1 as 5000
```

```
-sp_addtype ComplexDT,'int'
```

```
sp_bindrule r1,ComplexDT
```

```
sp_bindefault def1,ComplexDT
```

```
-create table test3
```

```
(  
    id int,
```

```
--Create New Datatype ----ComplexDT (int      >1000      default 5000)
create rule r1 as @x>1000
create default def1 as 5000

[-]sp_addtype ComplexDT,'int'

sp_bindrule r1,ComplexDT

sp_bindefault def1,ComplexDT

[-]create table test3
(
    id int,
```

batch

Function

Stored Procedure

### Variables

#### Local Var

← Declare @x int

--Set @x=10

← --Select @x=100

--Select @x=age from student where id=1

--update Student set fname='omar', @x=age  
where id=9

← Select @x

#### Global Var

can't declare Global Var

can't assign Global Var

@@

Select @@servername

Select @@rowcount

Select @@version

Select @@error

Select @@identity

Select @x=@@rowcount

Select @@rowcount=100 XXXXXXXX

Go to Settings to activate Windows

```
declare @x int  
set @x=10  
select @x
```

```
declare @x int  
setlect @x=10  
select @x
```

```
declare @x int=100  
select @x
```

```
declare @x int=(select avg(st_age) from student)  
select @x
```

```
declare @y int  
select st_age from student where st_id=6
```

```
declare @x int=100  
select @x
```

```
declare @x int=(select avg(st_age) from student)  
select @x
```

```
declare @y int  
select @y=st_age from student where st_id=6
```

```
declare @x int=(select avg(st_age) from student)
select @x
```

```
declare @y int=100
select @y=st_age from student where st_id=9
select @y
```

```
declare @y int=100
select @y=st_age from student where st_address='alex'
select @y
```

```
select @y

declare @y int,@name varchar(20)
select @y=st_age,@name=st_fname from student where st_id=7
select @y,@name

declare @z int
update student set st_fname='ali',@z=dept_id
where st_id=7
select @z
```

```
declare @z int  
update student set st_fname='ali',@z=dept_id  
where st_id=7  
select @z
```

```
declare @t table(x int)  
insert into @t  
select st_id from student where st_address='alex'  
select count(*) from @t
```

```
select count(*) from @t

declare @t table(x int)
insert into @t
select st_id from student where st_address='alex'
select * from @t

declare @t table(x int,y varchar(20))
insert into @t
select st_id,st_fname from student where st_address='alex'
select * from @t
```

```
select st_id,st_fname from student where st_address='alex'  
select * from @t  
  
declare @x int=5  
select top(@x)*  
from student  
  
declare @col varchar(20)= '*',@tab varchar(20)='student'  
select @col from @tab  
  
select * from student I
```

```
select st_id,st_fname from student where st_address='alex'  
select * from @t  
  
declare @x int=5  
select top(@x)*  
from student  
  
declare @col varchar(20)= '*',@tab varchar(20)='student'  
select @col from @tab  
  
select * from student
```

```
declare @x int=5
select top(@x)*
from student

declare @col varchar(20)='ins_name',@tab varchar(20)='instructor'
execute('select '+@col+' from '+@tab)

select 'select * from student'

execute('select * from student')
```

```
select 'select * from student'
```

```
execute('select * from student')
```

- - - - -

--Global Var

```
select @@SERVERNAME
```

```
Select @@version
```

--Global Var

```
select @@SERVERNAME
```

```
Select @@version
```

```
L  
update student  
set st_age+=1
```

--Global Var

```
select @@SERVERNAME
```

```
Select @@version
```

```
update student
```

```
    set st_age+=1
```

```
Select @@ROWCOUNT
```

Select @@version

- update student

set st\_age+=1

Select @@ROWCOUNT

Select @@ROWCOUNT

update student

    set st\_age+=1

Select @@ROWCOUNT

Select @@ROWCOUNT

```
[select * from student
```

```
go
```

Select @@error

```
Select @@ROWCOUNT
```

```
Select @@ROWCOUNT
```

```
select * from stud
```

```
go
```

```
- Select @@error
```

```
select @@IDENTITY
```

--Control of flow statement

--if

--begin

--end

--if exists if not exists

--while

--constinue

--break

```
--end  
--if exists if not exists  
--while  
--constinue  
--break  
--case  
--if  
--waitfor  
--cho|
```

--if|

--begin

--end

--if exists if not exists

--while

--continue

--break

--case

--if

... ± ⌂ ...

```
--if
--begin
--end
--if exists if not exists
--while
--constinue
--break
--case
--iif
--waitfor
--choose
```

```
--Control of flow statement
```

```
--if
```

```
declare @x int
```

```
update student
```

```
    set st_age+=1
```

```
Select @x=@@ROWCOUNT
```

```
if @x>0
```

```
    select 'multi rows Affected'
```

```
else
```

```
    select 'No Rows Affected'
```

```
--begin
```

```
declare @x int  
update student  
    set st_age+=1  
Select @x=@@ROWCOUNT  
if @x>0  
begin  
    select 'multi rows Affected'  
end  
else  
begin  
    select 'No Rows Affected'  
end
```

```
declare @x int  
update student  
    set st_age+=1  
Select @x=@@ROWCOUNT  
if @x>0  
begin  
select 'multi rows Affected'  
end  
else  
begin  
select 'No Rows Affected'  
end
```

```
begin  
    select 'No Rows Affected'  
end
```

```
--begin
```

```
--end
```

```
--if exists if not exists
```

```
create table student
```

```
(  
    id int,
```

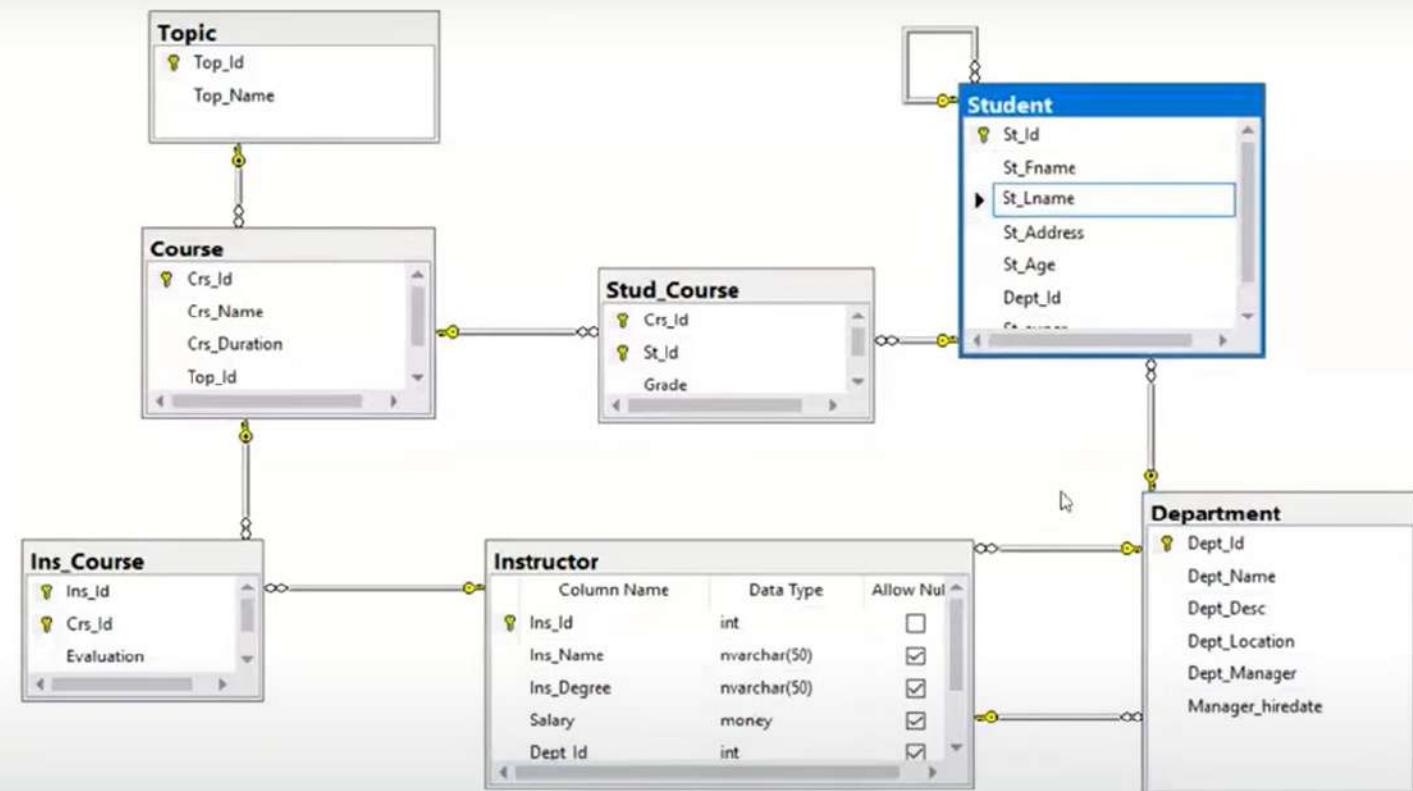
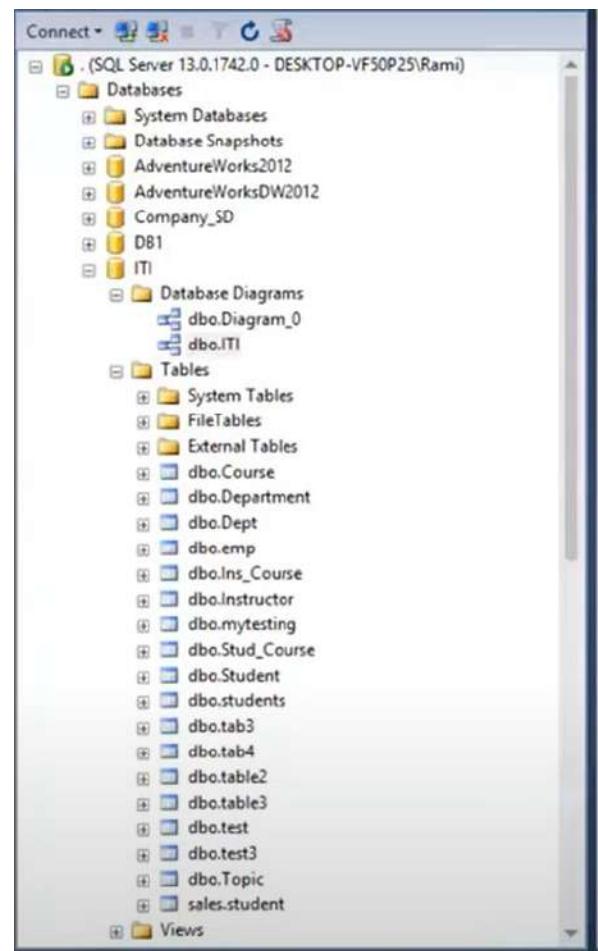
```
    name varchar(20)
```

```
- if exists(select name from sys.tables where name='student')
    select 'table is existed'
else
- create table student
(
    id int,
    name varchar(20)
)
```

```
- if exists(select name from sys.tables where name='students')
    select 'table is existed'
else
-   create table students
(
    id int,
    name varchar(20)
)
```

```
    name varchar(20)
)
delete from department where dept_id=20
```

- --while
- --constinue
- --break



Activate Windows

plate Browser Properties

```
    name varchar(20)
)
if not exists(select dept_id from student where dept_id=20)
    and not exists( select dept_id from Instructor where Dept_Id=20)
        delete from department where dept_id=20
else
    select 'table has relationship'
```

```
if not exists(select dept_id from student where dept_id=20)
    and not exists( select dept_id from Instructor where Dept_Id=20)
        delete from department where dept_id=20
else
    select 'table has relationship'

delete from department where dept_id=20
```

```
else
    select 'table has relationship'

begin try
    delete from department where dept_id=20
end try
begin catch
    select 'error'
end catch
```

```
      select 1/0
      delete from department where dept_id=20
begin try
end try
begin catch
    select 'error'
    select ERROR_LINE(),ERROR_NUMBER(),ERROR_MESSAGE()
end catch
```

```
--while
declare @x int=10
while @x<=20
begin
    Set @x+=1
    if @x=14
        continue
    if @x=16
        break
    select @x
end
```

```
declare @x int=10
while @x<=20
begin
    Set @x+=1
    if @x=14
        continue
    if @x=16
        break
    select @x
end
```

```
break
select @x
end
--continue
--break
--case
--iif
--waitfor
--choose
-----
--batch transac
```

insert  
update  
delete

create table

go

- drop table

crea|

Connect ▾

.(SQL Server 13.0.1742.0 - DESKTOP-1)

Databases

- System Databases
- Database Snapshots
- AdventureWorks2012
- AdventureWorksDW2012
- Company\_SD
- DB1

New Database...  
New Query  
Script Database as  
Tasks  
Policies  
Facets  
Start PowerShell  
Reports  
Rename  
Delete  
Refresh  
Properties

dbo.students  
dbo.tab3  
dbo.tab4  
dbo.table2  
dbo.table3  
dbo.test  
dbo.test3  
dbo.Topic

1 12  
(No)  
1 13  
(No)  
1 15

Query

Output

delete

create table

Detach...  
Take Offline  
Bring Online  
Stretch  
Encrypt Columns...  
Shrink  
Back Up...  
Restore  
Mirror...  
Launch Database Mirroring Monitor...  
Ship Transaction Logs...  
Generate Scripts...  
Generate In-Memory OLTP Migration Checklists  
Extract Data-tier Application...  
Deploy Database to Microsoft Azure SQL Database...  
Deploy Database to a Microsoft Azure VM...  
Export Data-tier Application...  
Register as Data-tier Application...  
Upgrade Data-tier Application...  
Delete Data-tier Application...  
Import Data...  
Export Data...  
Copy Database...  
Manage Database Encryption...

Connect ▾

.(SQL Server 13.0.1742.0 - DESKTOP-1)

Databases

- System Databases
- Database Snapshots
- AdventureWorks2012
- AdventureWorksDW2012
- Company\_SD
- DB1
- ITI
- Database Diagrams
  - dbo.Diagram\_0
  - dbo.ITI
- Tables
  - System Tables
  - FileTables
  - External Tables
  - dbo.Course
  - dbo.Department
  - dbo.Dept
  - dbo.emp
  - dbo.Ins\_Course
  - dbo.Instructor
  - dbo.mytesting
  - dbo.Stud\_Course
  - dbo.Student
  - dbo.students
  - dbo.tab3
  - dbo.tab4
  - dbo.table2
  - dbo.table3
  - dbo.test
  - dbo.test3
  - dbo.Topic

USE [master]  
GO  
\*\*\*\*\* Object: Database [ITI] Script Date: 25/11/2020 12:14:02 \*\*\*\*\*  
CREATE DATABASE [ITI]  
CONTAINMENT = NONE  
ON PRIMARY  
( NAME = N'ITI', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQL13\DATA\ITI.mdf' )  
LOG ON  
( NAME = N'ITI\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQL13\LOG\ITI\_log.ldf' )  
GO  
ALTER DATABASE [ITI] SET COMPATIBILITY\_LEVEL = 100  
GO  
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))  
begin  
EXEC [ITI].[dbo].[sp\_fulltext\_database] @action = 'enable'  
end  
GO  
ALTER DATABASE [ITI] SET ANST NULL DEFAULT OFF

Connected. (1/1) (local) (13.0 RTM) DESKTOP-VF50P25(Rami (64)) master 00:00:00 0 rows

delete

create table

go

drop table

go

create rule

go

sp\_bindrule

batch  
Function  
Stored Procedure

## Variables

Local Var

Global Var

can't declare Global Var  
can't assign Global Var

← Declare @x int

@@

--Set @x=10

Select @@servername

← --Select @x=100

Select @@rowcount

--Select @x=age from student where id=1

Select @@version

--update Student set fname='omar',@x=age  
where id=9

Select @@error

← Select @x

Select @@identity

Select @x=@@rowcount

Select @@rowcount=100 XXXXXXXX

Update Win

Go to Settings to

Query SSMS CMD

Insert -----> execute (F5)

Delete -----> execute

begin transaction  
insert  
update  
delete  
--Commit --rollback

begin transaction  
insert  
update  
delete  
--Commit --rollback

Ldf

begin transaction  
insert

Commit

begin transaction  
Delete

Rollback

begin transaction

insert  
update

Restart Server .....

automatic Rollback

begin transaction

insert  
update  
delete  
Commit rollback

mdf

>ok

>error

>ok

>ok

Activate Windows

Go to Settings to activate Windows.

Query SSMS CMD

Insert -----> execute (F5)

Delete -----> execute

```
begin transaction  
  insert  
  update  
  delete  
--Commit --rollback
```

```
begin transaction  
  insert  
  update  
  delete  
--Commit --rollback
```

Ldf

begin transaction

insert

Commit

begin transaction

Delete

Rollback

begin transaction

insert

update

Restart Server .....

automatic Rollback

begin transaction

insert

update

delete

Commit rollback

mdf

>ok

>error

>ok

>ok

Activate Windows

Go to Settings to activate Windows

```
create rule
```

```
go
```

```
sp_bindrule
```

--Transaction [set of queries] ----> Single unit of work

```
create table parent(pid int primary key)
```

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

--Transaction [set of queries] ----> Single unit of work

```
create table parent(pid int primary key)
```

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

```
insert into parent values(1)
```

```
insert into parent values(2)
```

```
insert into parent values()
```

```
insert into parent values(1)
```

--Transaction [set of queries] ----> Single unit of work

```
create table parent(pid int primary key)
```

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

```
insert into parent values(1)
insert into parent values(2)
insert into parent values(3)
insert into parent values(4)
```

```
insert into parent values(1)
insert into parent values(2)
insert into parent values(3)
insert into parent values(4)
```

```
insert into child values(1)
insert into child values(5)
insert into child values(3)
```

```
select * from child  
truncate table child
```

```
begin transaction  
    insert into child values(1)  
    insert into child values()  
    insert into child values(3)  
rollback
```

```
select * from child  
truncate table child
```

```
begin transaction  
    insert into child values(1)  
    insert into child values(2)  
    insert into child values(3)  
rollback
```

rollback

begin transaction

    insert into child values(1)

    insert into child values(5)

    insert into child values(3)

commit

rollback

```
begin transaction
    insert into child values(1)
    insert into child values(5)
    insert into child values(3)
commit
```

```
begin try
    begin transaction
        insert into child values(1)
        insert into child values(5)
        insert into child values(3)
    commit
end try
begin catch
    rollback
```

```
begin try
    begin transaction
        insert into child values(1)
        insert into child values(5)
        insert into child values(3)
    commit
end try
begin catch
    rollback
end catch
```

```
begin try
    begin transaction
        insert into child values(1)
        insert into child values(2)
        insert into child values(3)
    commit
end try
begin catch
    rollback
end catch
```

```
truncate table child
```

```
begin transaction
    insert into child values(1)
    insert into child values(2)
    insert into child values(3)
rollback
```

```
begin transaction
    insert into child values(1)
```

## Scalar Function

## Functions

### Built in Functions

--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 Function iif , Choose  
--Windowing   Lead Lag First\_Value last\_value

### User Defined Functions

#### Scalar Function

Return One Value

#### Inline table Function

Return Table ----> body ----> Select ---> view

#### Multi statement table valued Function

Return Table --->body ----> Select ---> if ,declare, while  
insert based on select

## Scalar Function

--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 Function iif , Choose  
--Windowing   Lead Lag First\_Value last\_value

## Functions

### Built in Functions

### User Defined Functions

#### Scalar Function

Return One Value

#### Inline table Function

Return Table ----> body ----> Select ---> view

#### Multi statement table valued Function

Return Table --->body ----> Select ---> if ,declare, while

insert based on select

--Functions

select getdate()

select isnull(st\_fname,'')  
from student

select upper(st\_fname),lower(st\_lname)  
from student

```
- select isnull(st_fname, '')  
  from student
```

```
- select upper(st_fname),lower(st_lname)  
  from student
```

```
- select len(st_fname)  
  from student
```

```
- select upper(st_fname),lower(st_lname)
  from student
```

```
- select len(st_fname),st_fname
  from student
```

```
| select max(st_fname)
|   from student
```

```
- select upper(st_fname),lower(st_lname)
  from student

- select len(st_fname),st_fname
  from student

select max(st_fname)
  from student| I
```

```
- select isnull(st_fname, '')  
  from student
```

```
- select Coalesce(st_fname,st_lname,'')  
  from student
```

```
- select upper(st_fname),lower(st_lname)  
  from student
```

```
select getdate()

[-] select isnull(st_fname, '')  
      from student

[-] select Coalesce(st_fname,st_lname,st_address,'')  
      from student

[-] select upper(st_fname),lower(st_lname)  
      from student
```

```
- select len(st_fname),st_fname  
from student
```

```
- select max(st_fname)  
from student
```

```
- select max(len(st_fname))  
from student
```

```
from student
```

```
- select max(len(st_fname))
```

```
from student
```

```
                                I
```

```
- select top(1) st_fname
```

```
from student
```

```
order by len(st_fname) desc
```

```
from student  
order by len(st_fname) desc
```

```
select power(salary,2)  
from Instructor
```

```
select convert(varchar(20),getdate(),101)
```

```
select format(getdate(), 'dd-MM-yyyy')
```

from Instructor

```
select convert(varchar(20),getdate(),101)
```

```
select format(getdate(),'dd-MM-yyyy')
```

```
select db_name(),suser_name()
```

--Create

```
--Create My Own Functions
--string getsname(int id);
create function getsname(@id int)
returns varchar(20)
begin
    declare @name varchar(20)
    select @name=st_fname from student where st_id=@id
    return @name
end
```

```
--Create My Own Functions
--string getsname(int id);
create function getsname(@id int)
returns varchar(20)
begin
    declare @name varchar(20)
    select @name=st_fname from student where st_id=@id
    return @name
end
```

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

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

select getsname(1)
```

```
--  
begin  
    declare @name varchar(20)  
    select @name=st_fname from student where st_id=@id  
    return @name  
end
```

```
select dbo.getsname(1)
```

```
--string getsname(int id);  
  
create function getsname(@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.getsname(3)  
----- I  
  
create function Getist
```

```
- - - - -  
- [ ] create function Getist(@did int)  
returns table  
as  
return  
(  
    select ins_name,salary*12  
    from Instructor  
    where dept_id=@did
```

```
- create function Getist(@did int)
returns table
as
return
(
    select ins_name, salary*12
    from Instructor
    where dept_id=@did
```

```
- create function Getist(@did int)
returns table
as
return
(
    select ins_name, salary*12 as totalsal
    from Instructor
    where dept_id=@did
)
```

```
as
return
(
    select ins_name,salary*12 as totalsal
    from Instructor
    where dept_id=@did
)
```

```
select * from Getist(10)
```

```
        where dept_id=@did  
    )  
  
select * from Getist(10)  
select ins_name from Getist(10)  
select sum(totalsal) from Getist(10)
```

```
- create function Getist(@did int)
returns table
as
return
(
    select ins_name,salary*12 as totalsal
    from Instructor
    where dept_id=@did
```

```
create function Getist(@did int)
returns table
as
return
(
    select ins_name,salary*12 as totalsal
    from Instructor
    where dept_id=@did
)
```

```
        from Instructor  
        where dept_id=@did  
    )  
  
select * from Getist(10)  
select ins_name from Getist(10)  
select sum(totalsal) from Getist(10)
```

```
--Multistatement
create function getstuds(@format varchar(20))
returns @t table
(
    id int,
    ename varchar(20)
)
as
begin
```

```
        )
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
```

```
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 from student
```

```
create function getstuds(@format varchar(20))
returns @t table
(
    id int,
    ename 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'
```

```
--string getsname(int id);  
- create function getsname(@id int)  
  returns varchar(20)  
  begin  
    declare @name varchar(20)  
    select @name=st_fname from student where st_id=@id  
    return @name  
  end  
  I  
  select dbo.getsname(3)
```

## 7 SQL, Variables, If, While, functions

```
select s.st_id as sid,st_fname as sname,grade,crs_name as Cname into grades  
from Student s,Stud_Course sc,Course c  
where s.St_Id=sc.St_Id and c.Crs_Id=sc.Crs_Id
```

```
select * from grades
```

```
SELECT sname,grade,  
      Prod_prev=LAG(sname) OVER(ORDER BY grade),  
      Prod_Next=LEAD(sname) OVER(ORDER BY grade)  
FROM grades
```

```
SELECT sname,grade,cname,  
      Prod_prev=LAG(grade) OVER(partition by Cname ORDER BY grade),  
      Prod_Next=LEAD(grade) OVER(partition by Cname ORDER BY grade)  
FROM grades
```

```
SELECT sname,grade,  
      First=FIRST_VALUE(grade) OVER(ORDER BY grade),  
      last=LAST_VALUE(grade) OVER(ORDER BY grade Rows BETWEEN unbounded preceding AND unbounded following)  
FROM grades
```

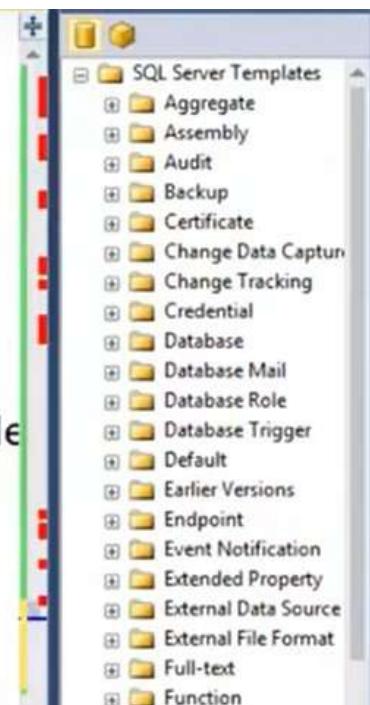
```
SELECT sname,grade,Cname,  
      FIRST_VALUE(grade) OVER(partition by Cname ORDER BY grade),  
      LAST_VALUE(grade) OVER(partition by Cname ORDER BY grade Rows BETWEEN unbounded preceding AND unbounded following)  
FROM grades
```

```
Select * from getstuds('full')

--windowing
--lead lag first_value last_value

select s.st_id as sid,st_fname as sname,grade,crs_name as Cname into grades
from Student s,Stud_Course sc,Course c
where s.St_Id=sc.St_Id and c.Crs_Id=sc.Crs_Id

select * from grades
```



```
select * from grades
```

```
-[SELECT surname, grade,  
     Prod_prev=LAG(surname) OVER(ORDER BY grade),  
     Prod_Next=LEAD(surname) OVER(ORDER BY grade)  
  FROM grades
```

```
-[SELECT surname, grade, cname,
```

200 %

Results Messages

	sid	sname	grade	Cname
1	4	Ahmed	90	HTML
2	5	NULL	120	HTML
3	6	Heba	110	HTML
4	4	Ahmed	100	C Progammimg
5	5	NULL	100	C Progammimg
6	8	Mohamed	80	C Progammimg
7	9	Saly	80	C Progammimg
8	5	NULL	90	OOP
9	6	Heba	90	OOP
10	10	NULL	120	OOP
11	4	Ahmed	110	Unix
12	5	NULL	110	Unix
13	6	Heba	100	SQL Server

Query executed successfully.

(local) (13.0 RTM) DESKTOP-VF50P25\Rami (60) ITI 00:00:00 18 rows

```
select * from grades
```

```
- SELECT sname, grade,  
      X= LAG(grade) OVER(ORDER BY grade),  
      Y= LEAD(grade) OVER(ORDER BY grade)  
   FROM grades
```

200 %

Results Messages

	sname	grade	X	Y
1	Mohamed	80	NULL	80
2	Saly	80	80	80
3	NULL	80	80	90
4	Ahmed	90	80	90
5	Mohamed	90	90	90
6	NULL	90	90	90
7	Heba	90	90	90
8	Ahmed	90	90	100
9	Ahmed	100	90	100
10	NULL	100	100	100
11	Heba	100	100	110
12	Heba	110	100	110
13	Ahmed	110	110	110
14	NULL	110	110	110

## 7 SQL, Variables, If, While, functions

variables.sql - (local)\ITI (DESKTOP-VF50P25\Rami (60)) - Microsoft SQL Server Management Studio

Object Explorer    Connect    New Query    Execute    Debug    Generic Debugger    Properties

DESKTOP-VF50P25\ITI - dbo.Student    variables.sql - (local)\P-VF50P25\Rami (60))\*    SQLQuery1.sql - (local)\VF50P25\Rami (57)

```
select * from grades
```

```
SELECT sname, grade,
       X= LAG(grade) OVER(ORDER BY grade),
       Y= LEAD(grade) OVER(ORDER BY grade)
  FROM grades
```

Results    Messages

	sname	grade	X	Y
1	Mohamed	80	NULL	80
2	Saly	80	80	80
3	NULL	80	80	90
4	Ahmed	90	80	90
5	Mohamed	90	90	90
6	NULL	90	90	90
7	Heba	90	90	90
8	Ahmed	90	90	100
9	Ahmed	100	90	100
10	NULL	100	100	100
11	Heba	100	100	110
12	Heba	110	100	110
13	Ahmed	110	110	110
14	NULL	110	110	110

Query executed successfully.

(local) (13.0 RTM) DESKTOP-VF50P25\Rami (60) ITI 00:00:00 18 rows

Output    Ready

Activate Windows  
Go to Settings to activate Windows.

Activate Windows  
Go to Settings to activate Windows.

+17 H MA A M M A K R Y R

2:46:22 / 3:13:07

```
Y= LEAD(grade) OVER(ORDER BY grade)
FROM grades
```

```
SELECT sname,grade,
       I | X= LAG(sname) OVER(ORDER BY grade),
       Y= LEAD(sname) OVER(ORDER BY grade)
FROM grades
```

200 %

Results Messages

	sname	grade	X	Y
1	Mohamed	80	NULL	80
2	Saly	80	80	80
3	NULL	80	80	90
4	Ahmed	90	80	90
5	Mohamed	90	90	90
6	NULL	90	90	90
7	Heba	90	90	90
8	Ahmed	90	90	100
9	Ahmed	100	90	100
10	NULL	100	100	100
11	Heba	100	100	110
12	Heba	110	100	110
13	Ahmed	110	110	110
14	NULL	110	110	110

```
Y= LEAD(grade) OVER(ORDER BY grade)
FROM grades
```

```
SELECT sname,grade,
X= LAG(sname) OVER(ORDER BY grade),
Y= LEAD(sname) OVER(ORDER BY grade)
FROM grades
```

200 %

Results Messages

	sname	grade	X	Y
1	Mohamed	80	NULL	Saly
2	Saly	80	Mohamed	NULL
3	NULL	80	Saly	Ahmed
4	Ahmed	90	NULL	Mohamed
5	Mohamed	90	Ahmed	NULL
6	NULL	90	Mohamed	Heba
7	Heba	90	NULL	Ahmed
8	Ahmed	90	Heba	Ahmed
9	Ahmed	100	Ahmed	NULL
10	NULL	100	Ahmed	Heba
11	Heba	100	NULL	Heba
12	Heba	110	Heba	Ahmed
13	Ahmed	110	Heba	NULL
14	NULL	110	Ahmed	Saly

```
FROM grades
```

```
-| SELECT sname, grade,  
      X= LAG(sname) OVER(ORDER BY grade),  
      Y= LEAD(sname) OVER(ORDER BY grade)  
  FROM grades
```

200 %

Results Messages

	sname	grade	X	Y
1	Mohamed	80	NULL	Omar
2	Omar	80	Mohamed	Saly
3	Saly	84	Omar	Ahmed
4	Ahmed	90	Saly	Hassan
5	Hassan	91	Ahmed	Heba
6	Heba	92	Hassan	Mohamed
7	Mohamed	93	Heba	Ahmed
8	Ahmed	97	Mohamed	Ahmed
9	Ahmed	100	Ahmed	eman
10	eman	100	Ahmed	Heba
11	Heba	103	eman	Heba
12	Heba	110	Heba	Ahmed
13	Ahmed	110	Heba	Saly
14	Saly	110	Ahmed	Fady

Query executed successfully.

(local) (13.0 RTM) DESKTOP-VF50P25\Ram (60) ITI 00:00:00 18 rows

```
Y= LEAD(grade) OVER(ORDER BY grade)
FROM grades
```

```
SELECT sname, grade,
X= LAG(sname) OVER(ORDER BY grade),
Y= LEAD(sname) OVER(ORDER BY grade)
FROM grades
```

200 %

Results Messages

	sname	grade	X	Y
1	Mohamed	80	NULL	80
2	Omar	80	80	84
3	Saly	84	80	90
4	Ahmed	90	84	91
5	Hassan	91	90	92
6	Heba	92	91	93
7	Mohamed	93	92	97
8	Ahmed	97	93	100
9	Ahmed	100	97	100
10	eman	100	100	103
11	Heba	103	100	110
12	Heba	110	103	110
13	Ahmed	110	110	110
14	Saly	110	110	110

```

FROM grades
|-
[-] SELECT sname,grade,
      X= LAG(sname) OVER(ORDER BY grade),
      Y= LEAD(sname) OVER(ORDER BY grade)
FROM grades

```

200 % 4

Results Messages

	sname	grade	X	Y
1	Mohamed	80	NULL	Omar
2	Omar	80	Mohamed	Saly
3	Saly	84	Omar	Ahmed
4	Ahmed	90	Saly	Hassan
5	Hassan	91	Ahmed	Heba
6	Heba	92	Hassan	Mohamed
7	Mohamed	93	Heba	Ahmed
8	Ahmed	97	Mohamed	Ahmed
9	Ahmed	100	Ahmed	eman
10	eman	100	Ahmed	Heba
11	Heba	103	eman	Heba
12	Heba	110	Heba	Ahmed
13	Ahmed	110	Heba	Saly
14	Saly	110	Ahmed	Fady

Query executed successfully.

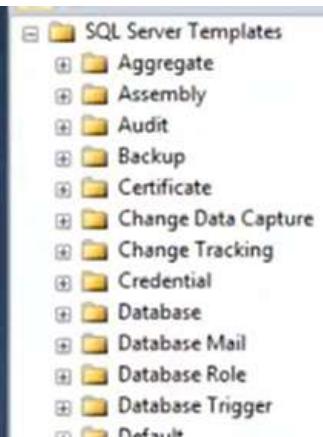
(local) (13.0 RTM) DESKTOP-VF50P25\Ram (60) | ITI 00:00:00 | 18 rows

```
select *
from (
-SELECT sname,grade,
      X= LAG(grade) OVER(ORDER BY grade),
      Y= LEAD(grade) OVER(ORDER BY grade)
FROM grades) as Newtable
where sname='eman'
```

```
- SELECT sname, grade,  
      X= LAG(grade) OVER(ORDER BY grade),  
      Y= LEAD(grade) OVER(ORDER BY grade)  
  from grades
```

```
- select *
```

```
SELECT sname, grade,
       Prod_prev=LAG(grade) OVER(ORDER BY grade),
       Prod_Next=LEAD(grade) OVER(ORDER BY grade),
       First=FIRST_VALUE(grade) OVER(ORDER BY grade),
       last=LAST_VALUE(grade) OVER(ORDER BY grade Rows BETW
FROM grades
```



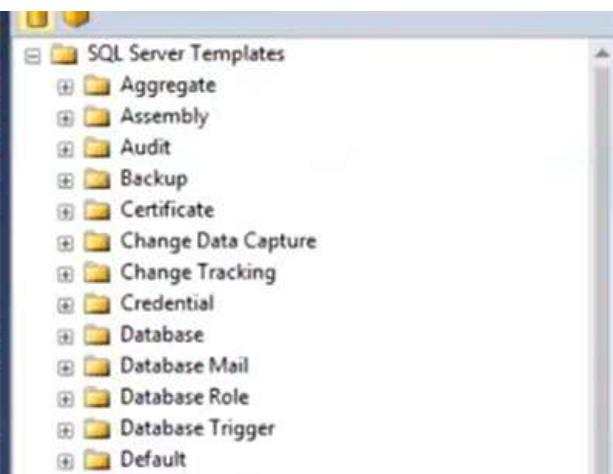
```
SELECT sname, grade,  
    Prod_prev=LAG(grade) OVER(ORDER BY grade),  
    Prod_Next=LEAD(grade) OVER(ORDER BY grade),  
    First=FIRST_VALUE(grade) OVER(ORDER BY grade),  
    last=LAST_VALUE(grade) OVER(ORDER BY grade Rows BETW  
FROM grades
```

- SQL Server Templates
  - Aggregate
  - Assembly
  - Audit
  - Backup
  - Certificate
  - Change Data Capture
  - Change Tracking
  - Credential
  - Database
  - Database Mail
  - Database Role
  - Database Trigger

```
SELECT sname, grade,  
       Prod_prev=LAG(grade) OVER(ORDER BY grade),  
       Prod_Next=LEAD(grade) OVER(ORDER BY grade),  
       First=FIRST_VALUE(grade) OVER(ORDER BY grade),  
       last=LAST_VALUE(grade) OVER(ORDER BY grade Rows BETW  
FROM grades
```

SQL Server Templates
Aggregate
Assembly
Audit
Backup
Certificate
Change Data Capture
Change Tracking
Credential
Database
Database Mail
Database Role
Database Trigger

```
SELECT sname, grade, Cname,
       Prod_prev=LAG(Sname) OVER(partition by Cname ORDER I
       Prod_Next=LEAD(Sname) OVER(partition by Cname ORDER
       First=FIRST_VALUE(Sname) OVER(partition by Cname ORDE
       last=LAST_VALUE(Sname) OVER(partition by Cname ORDE
FROM grades
```



use iti

declare @x int

set @x=10

select @x

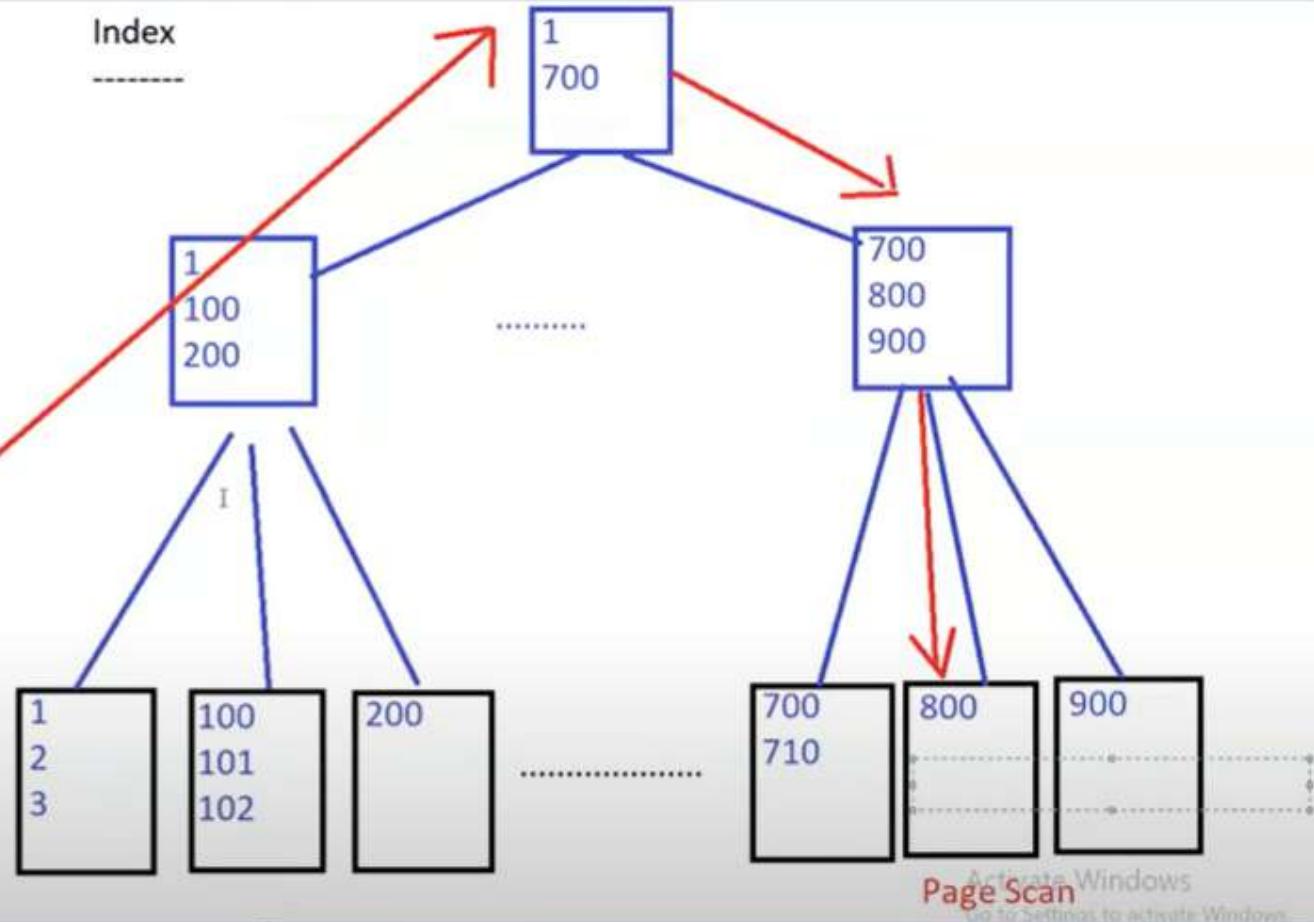
declare @x int

--7--+ 0.. 10

PK  
Sorted  
Clustered Index

Sid	Sname	age
1	ahmed	21
2	ali	23
4	eman	24
7	omar	25

Select \*  
from student  
where sid = 804



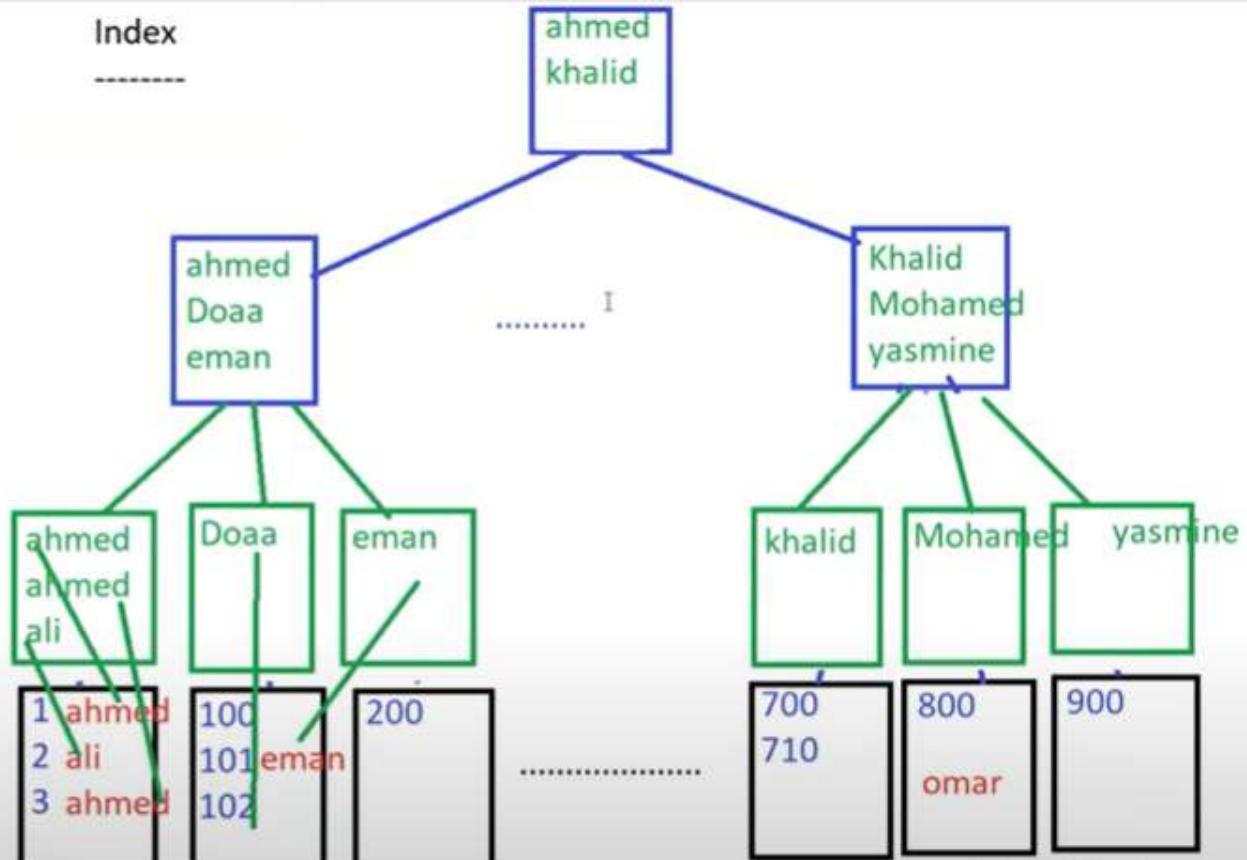
PK  
Sorted  
Clustered Index

NonClustered Index (Sname)

Sid	Sname	age
1	ahmed	21
2	ali	23
4	eman	24
7	omar	25

Select \*  
from student  
where name='omar'

Index



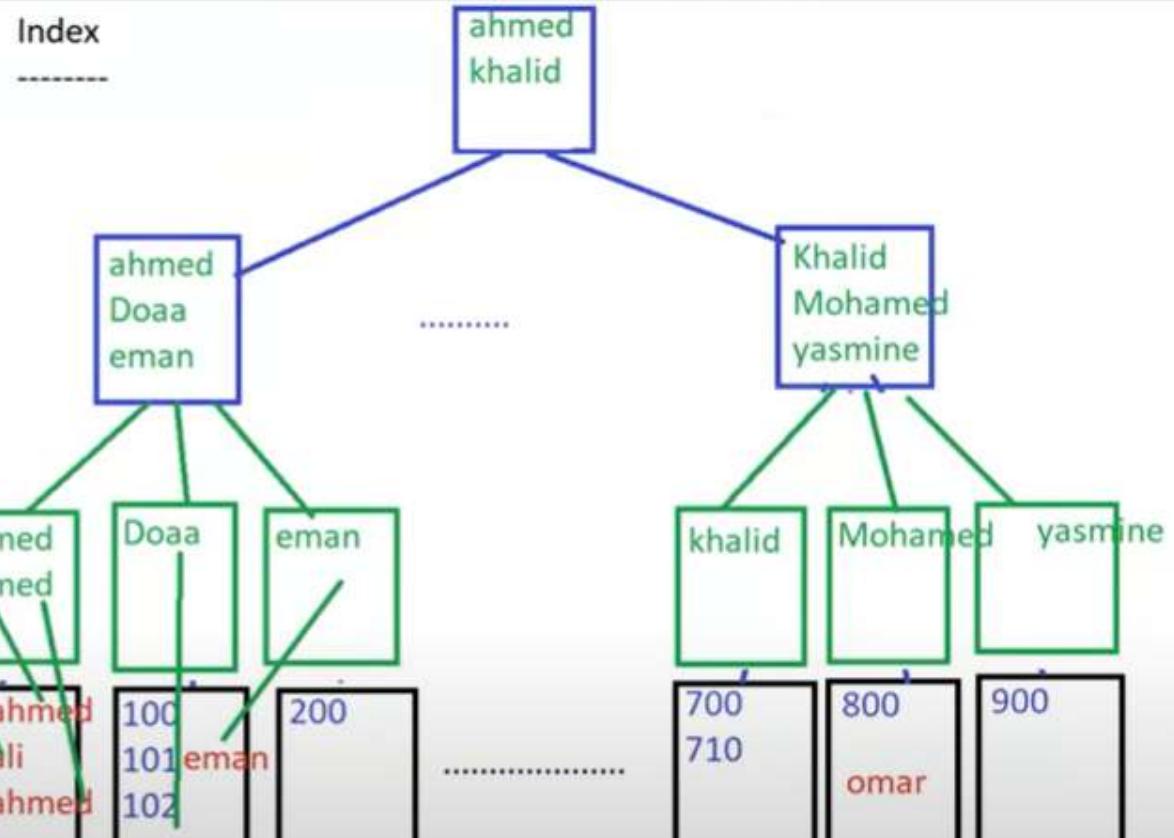
Activate Windows  
Get the latest for Windows 10

PK  
Sorted  
Clustered Index

NonClustered Index (Sname)

Sid	Sname	age
1	ahmed	21
2	ali	23
4	eman	24
7	omar	25

Select \*  
from student  
where name='omar'



Activate Windows  
Go to Settings to activate Windows

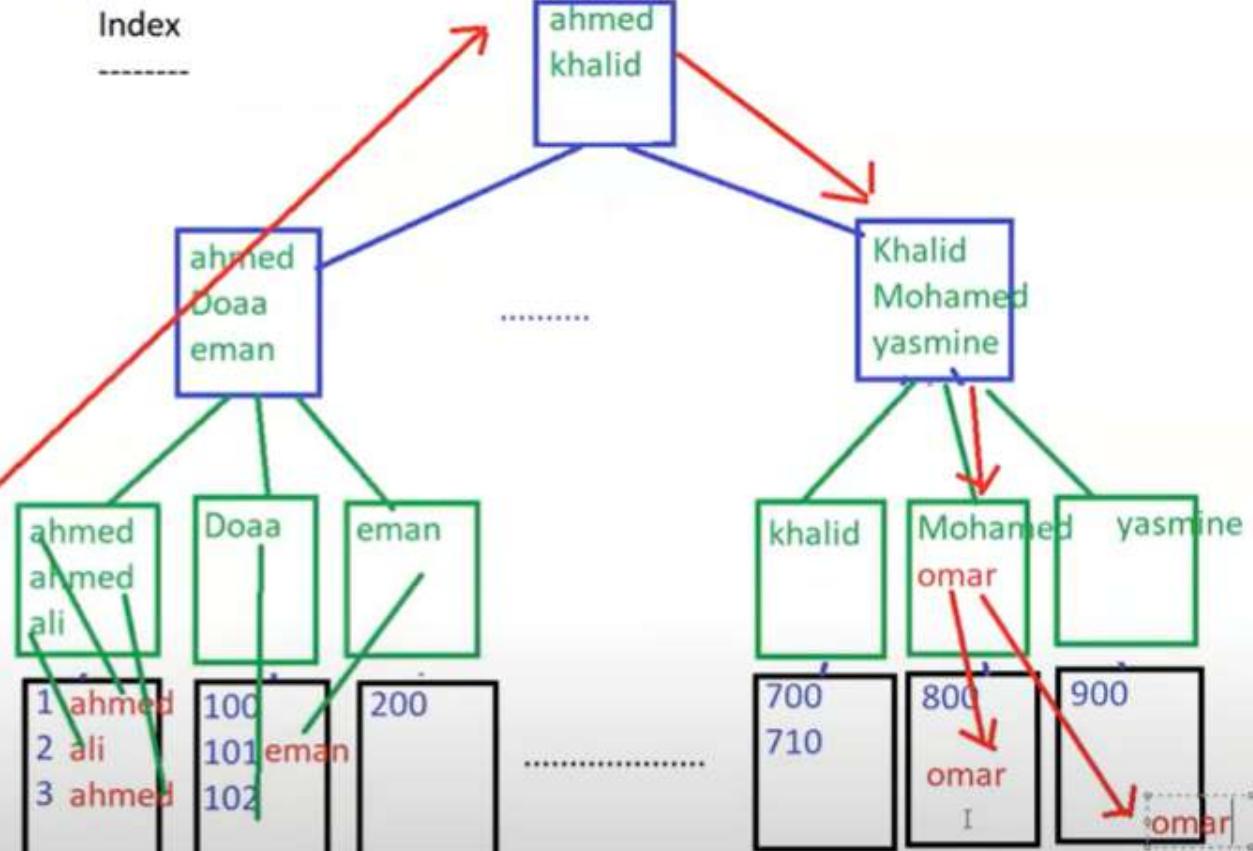
PK  
Sorted  
Clustered Index

NonClustered Index (Sname)

Sid	Sname	age
1	ahmed	21
2	ali	23
4	eman	24
7	omar	25

Select \*  
from student  
where name='omar'

Index



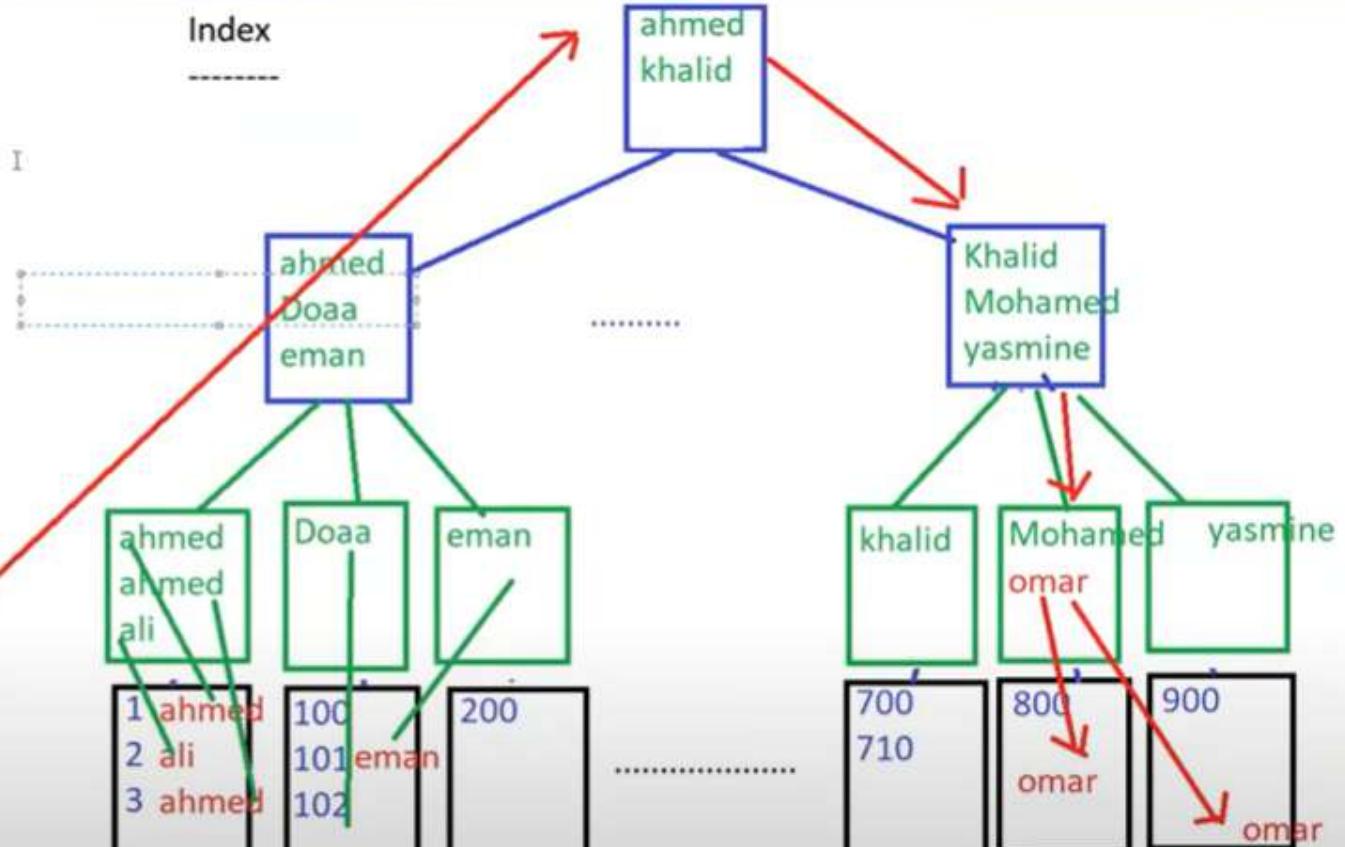
Activate Windows  
Go to Settings to activate Windows

PK  
Sorted  
Clustered Index

NonClustered Index (Sname)

Sid	Sname	age
1	ahmed	21
2	ali	23
4	eman	24
7	omar	25

Select \*  
from student  
where name='omar'



Activate Windows  
Go to Settings to activate Windows.

```
- create clustered index myindex  
on student(st_fname)
```

```
- create nonclustered index myindex  
on student(st_fname)
```

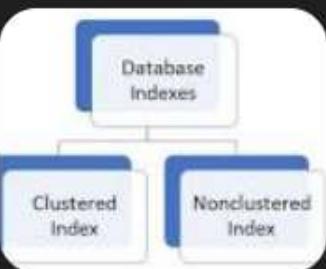
## difference between clustered and nonclustered index

All Images Videos Short videos Forums Web News More ▾

### AI Overview

The key difference between clustered and non-clustered indexes lies in how they affect the physical storage of data.

A clustered index determines the physical order of data rows in a table, while a non-clustered index creates a separate structure that points to the data rows. This distinction impacts performance characteristics for different types of queries.



Here's a more detailed breakdown:

#### Clustered Index:

- **Physical Order:** The data rows in the table are physically stored in the order of the clustered index key.
- **One per Table:** Only one clustered index can exist on a table.
- **Primary Key:** Often, the primary key is also the clustered index.
- **Performance:** Efficient for range-based queries and sorting, as data is physically organized for those operations.
- **Example:** A table storing customer information might have a clustered index on CustomerID to quickly retrieve customers by ID or to sort them by ID.

```
create clustered index myindex  
on student(st_fname)
```

```
create nonclustered index myindex  
on student(st_fname)
```

```
- create nonclustered index myindex2  
on student(st_Lname)
```

```
- select *  
from student  
where st_id=100
```

--PK --Constraint --> Clustered index

```
where st_id=100
```

```
--PK      --Constraint    --> Clustered index  
--unique  --Constraint    --> nonclustered
```

```
create table test22
```

```
(
```

```
    id int
```

```
)
```

```
create clustered index myindex
on student(st_fname)

create nonclustered index myindex
on student(st_fname)

create nonclustered index myindex2
```

--unique --Constraint --> nonclustered

```
create table test22
(
    id int primary key,
    name varchar(20),
    age int unique
)
```