

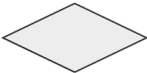







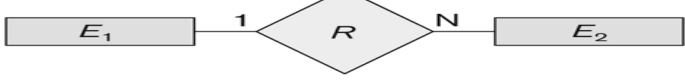
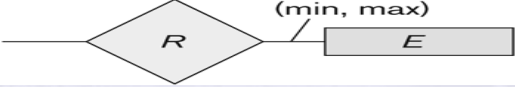


# Mapping

# 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. The table is a grid with 6 columns and 8 rows. The first row contains headers: 'SSAN', 'Name', 'Date of Birth', and three empty cells. The second row contains: an empty cell, an empty cell, '1/1/2012', and three empty cells. The third row is shaded gray. The fourth row contains: an empty cell, an empty cell, a shaded cell, and three empty cells. The fifth row contains: an empty cell, an empty cell, a shaded cell, and three empty cells. The sixth row contains: an empty cell, an empty cell, a shaded cell, and three empty cells. The seventh row contains: an empty cell, an empty cell, a shaded cell, and three empty cells. The eighth row contains: an empty cell, an empty cell, '31/12/2012', and three empty cells. Annotations include: 'Relation' pointing to the top-left corner, 'SSAN is a key' pointing to the 'SSAN' header, 'Column' pointing to the 'Date of Birth' header, and 'Tuple' pointing to the third row.

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

# Mapping -> DB Tables

CUSTOMER

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

Primary Key

ORDER

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

Foreign Key

ORDER LINE

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

composite primary key

PRODUCT

<u>Product_ID</u>	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.

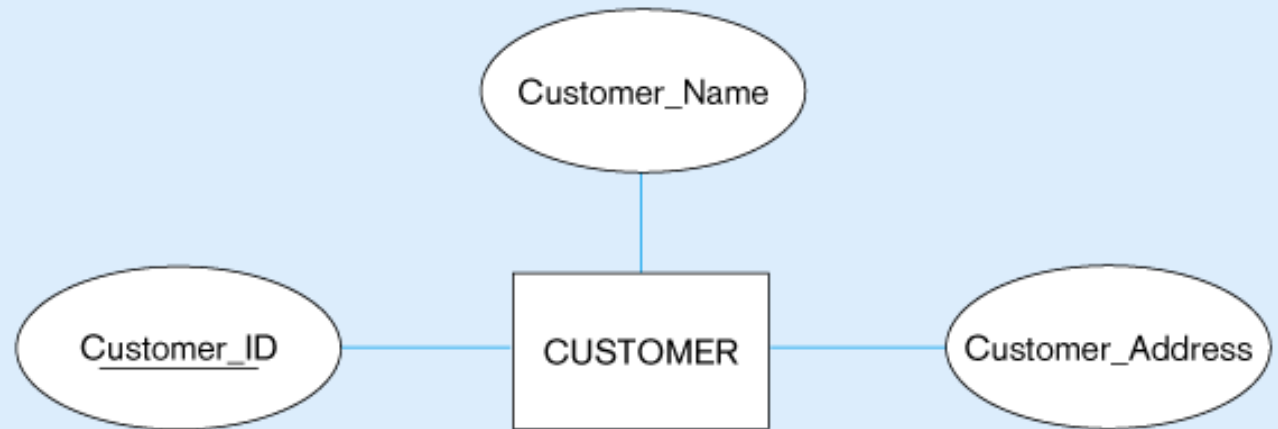
# Step 1: Mapping of Regular Entity Types

- 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

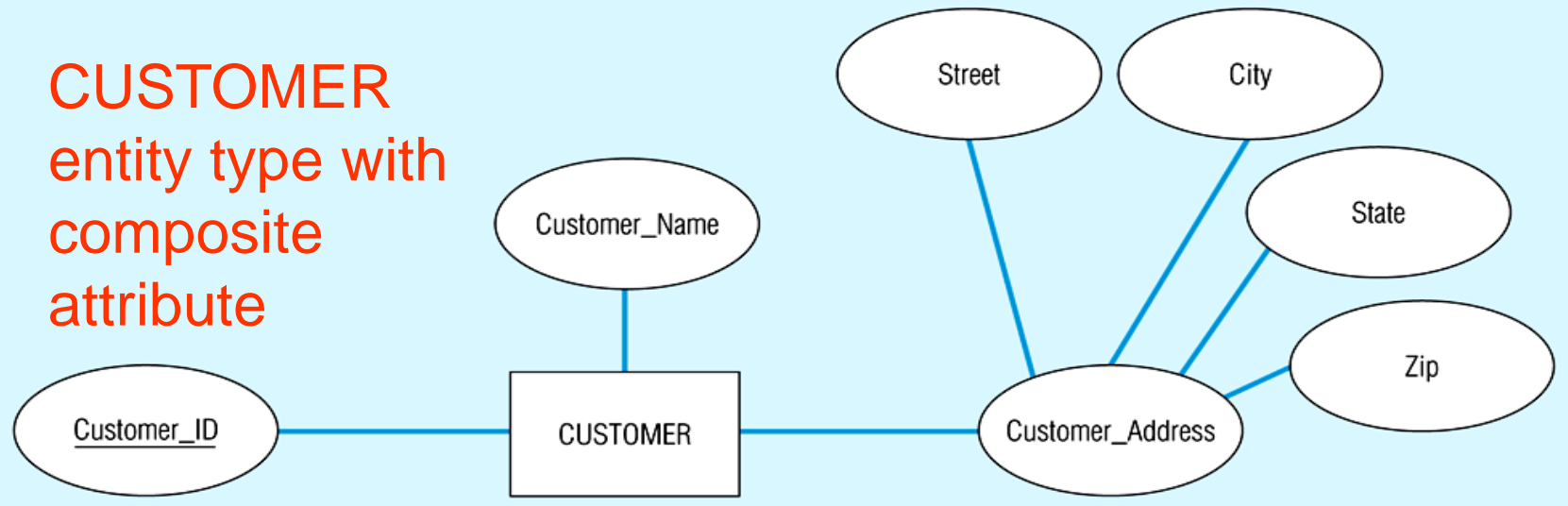


(b) CUSTOMER relation

CUSTOMER		
<u>Customer_ID</u>	Customer_Name	Customer_Address



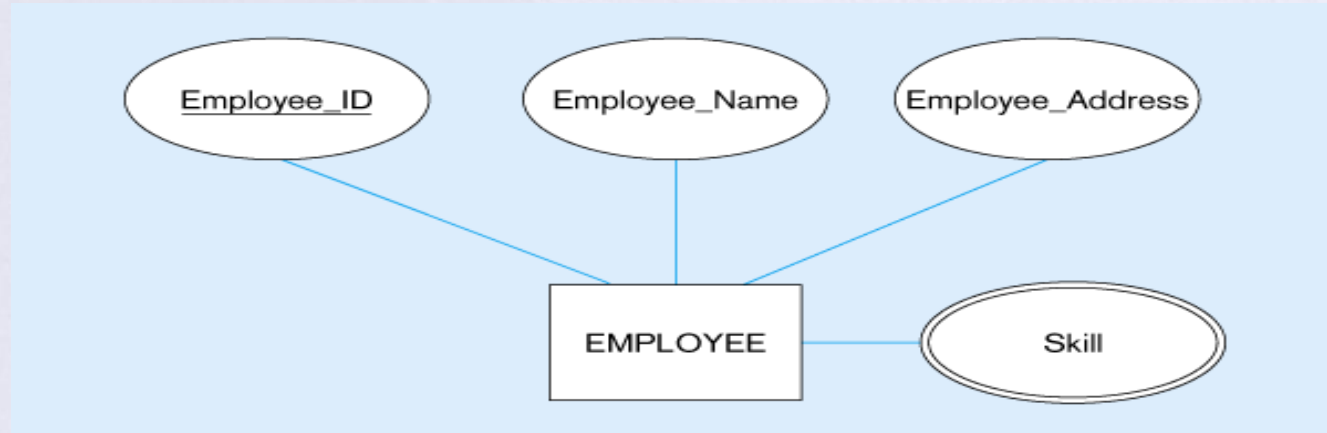
# Mapping Composite attribute



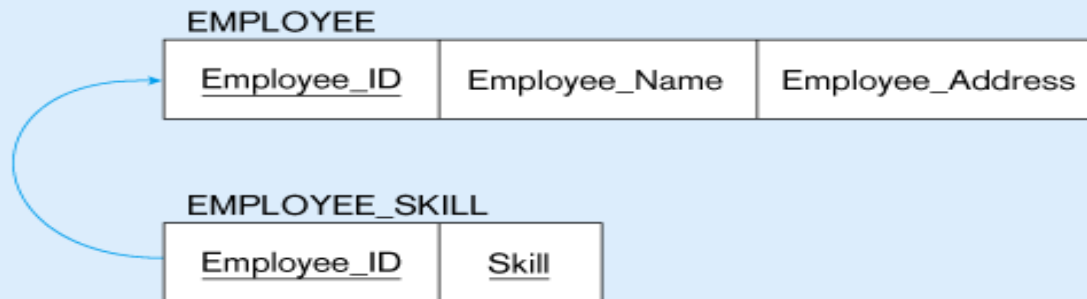
**CUSTOMER** relation with address detail

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

# Mapping Multivalued Attribute



**Multivalued attribute becomes a separate relation with foreign key**



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

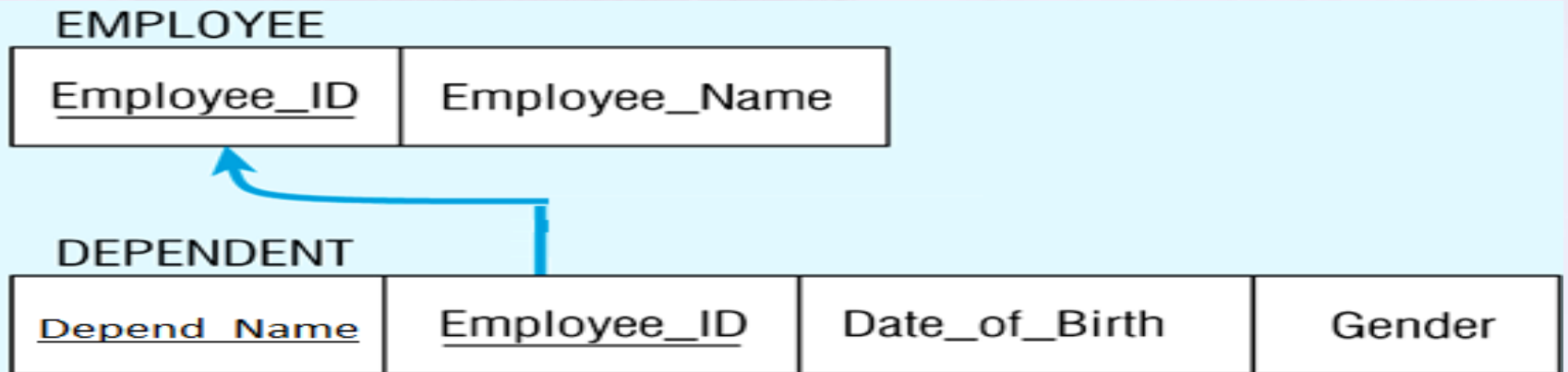
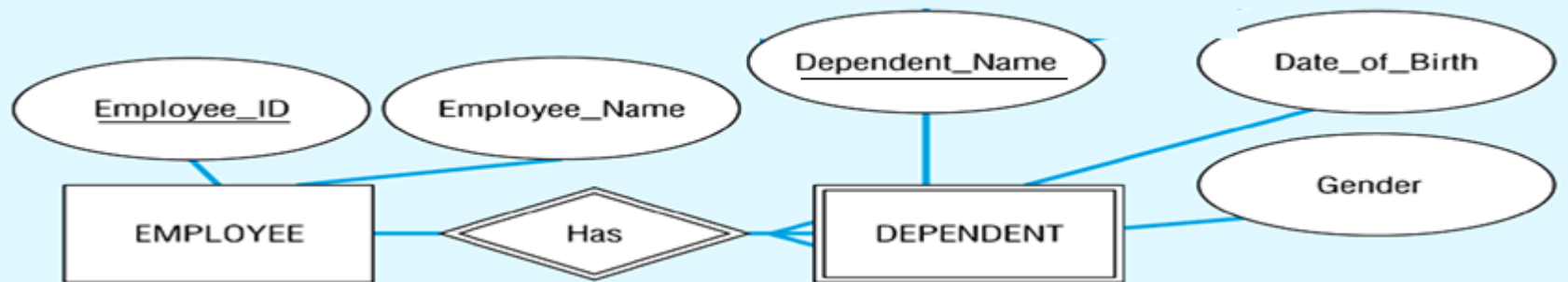
# Mapping Derived & Complex

- 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

## 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



Composite primary key

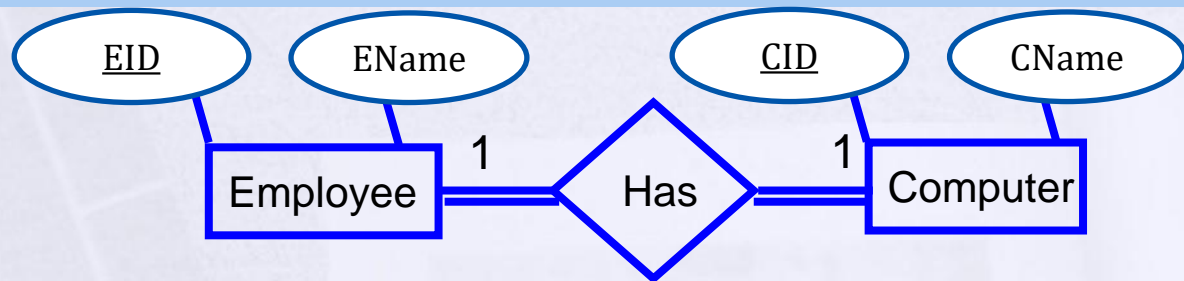
# Step 3: Mapping of Binary 1:1 Relation Types

- 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

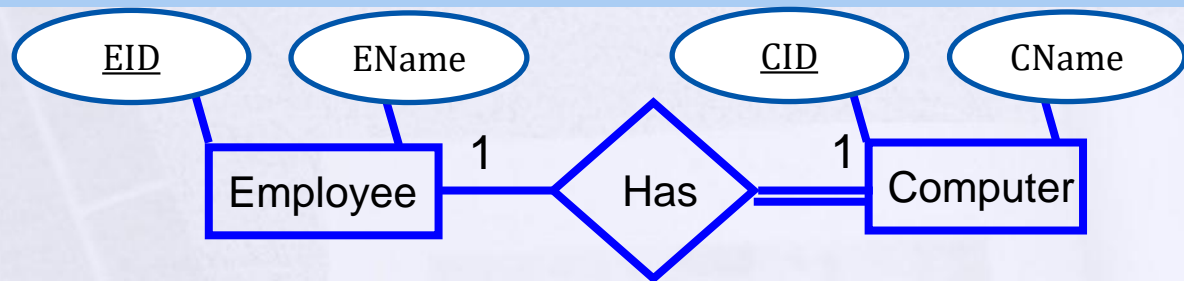
X optional – Y mandatory



### 2 tables

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

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



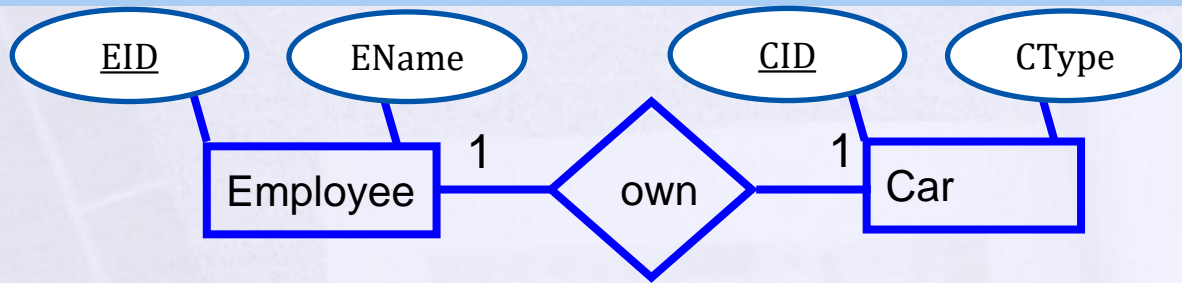
Employee(EID, Ename)

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

## 2 Optional

# One-to-One

## 2 Optional



## 3 tables

tbl\_x (PKx, ..., ..... )tbl\_y (PKy, ..., .....)tbl\_xy (PKxy,...,...,FKxy,...)
$$PK_{xy} = PK_x \text{ or } PK_y$$
Employee(EID, Ename)Car(CID, CType)

Emp\_Car(EID, CID\_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 Mandatory

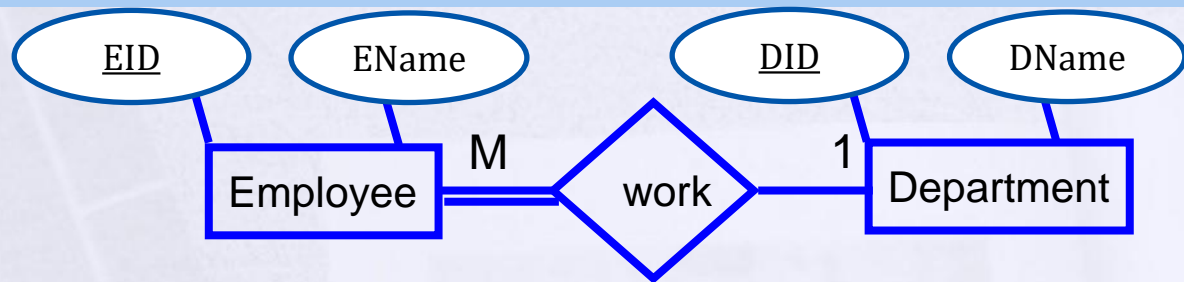
## One-to-Many

X whatever– Y mandatory



### 2 tables

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



Department(DID, Dname)

Employee(EID, Ename, **DID**)

# Many is Optional

## One-to-Many

X whatever– Y Optional



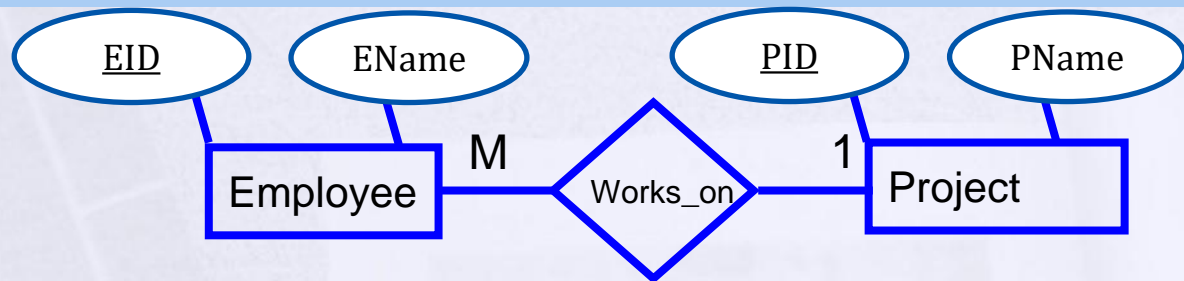
### 3 tables

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

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

tbl\_xy (PK<sub>xy</sub>, ..., .....

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



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



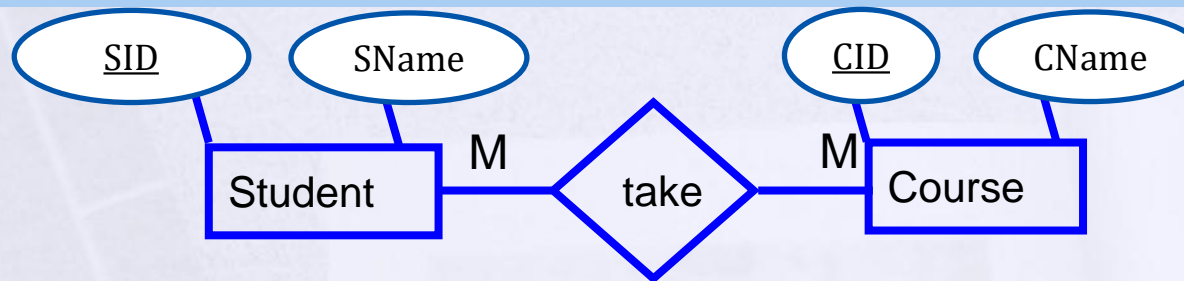
### 3 tables

tbl\_x (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)

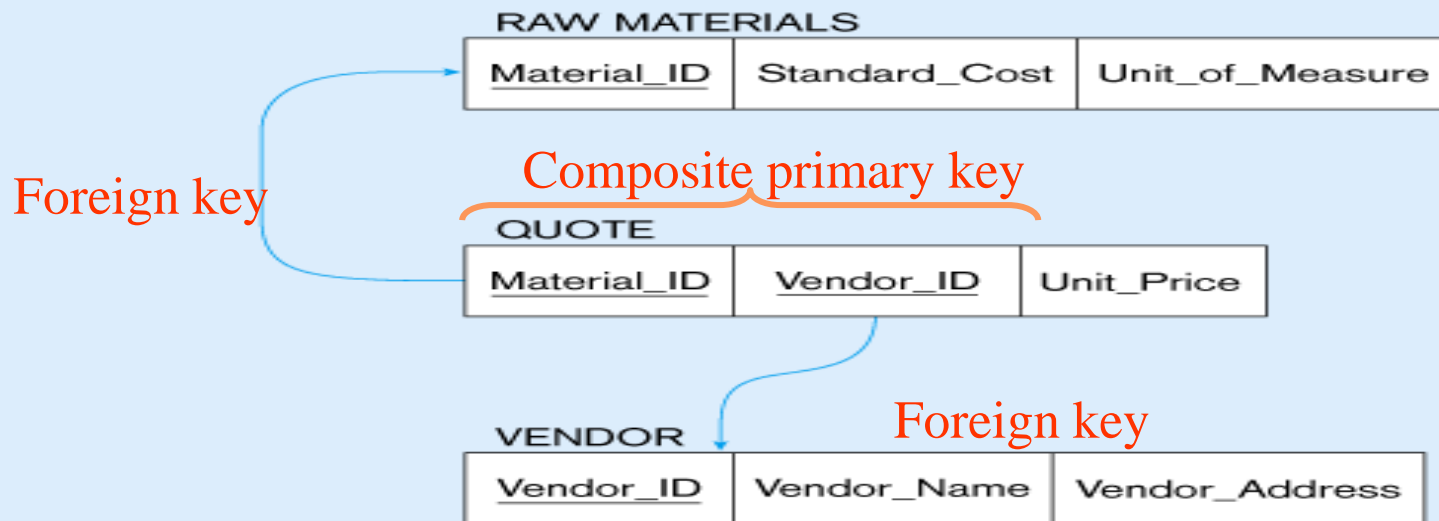
Stud\_Course(SID, CID)



# M:N with attribute



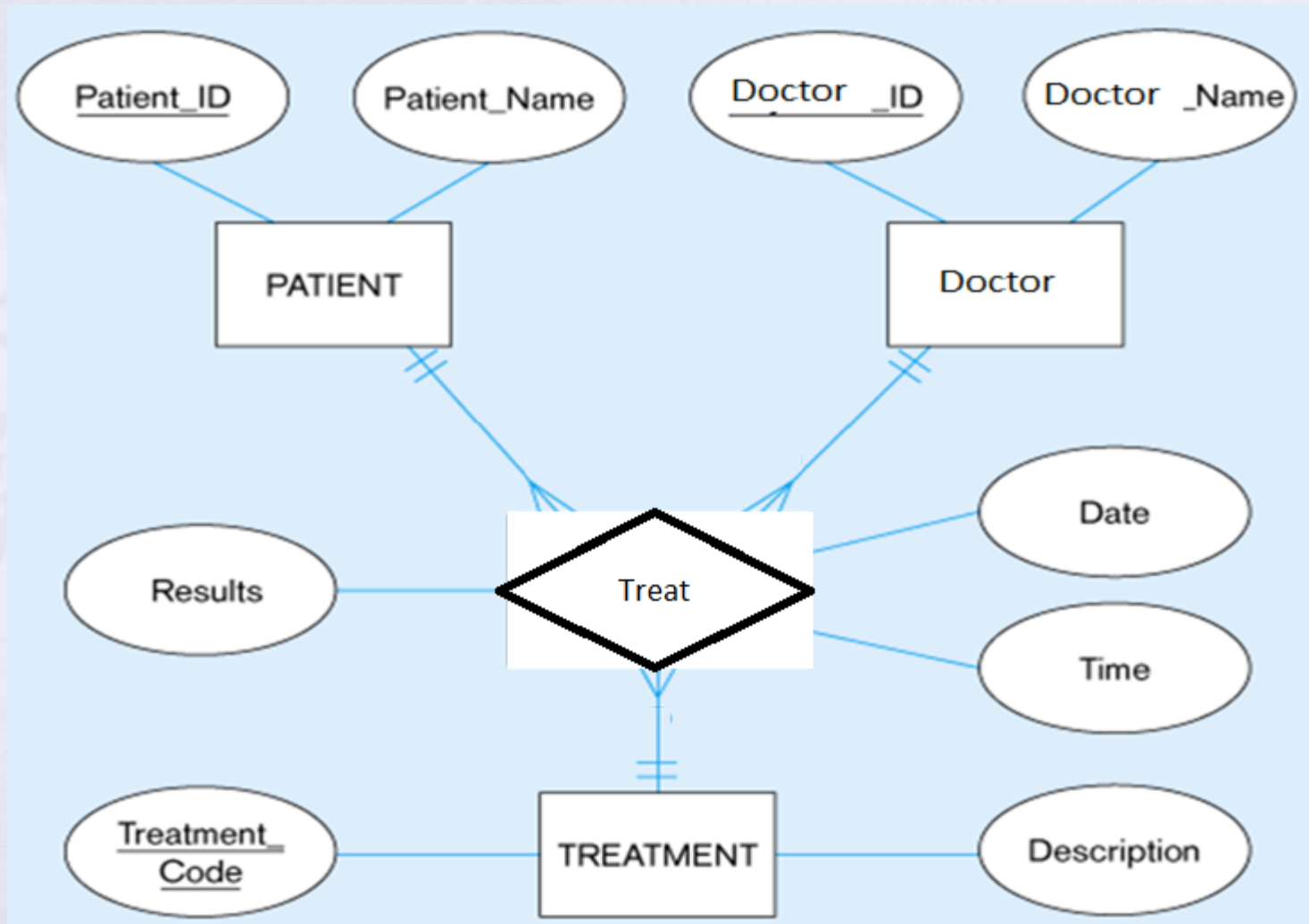
The *Supplies* relationship will need to become a separate relation



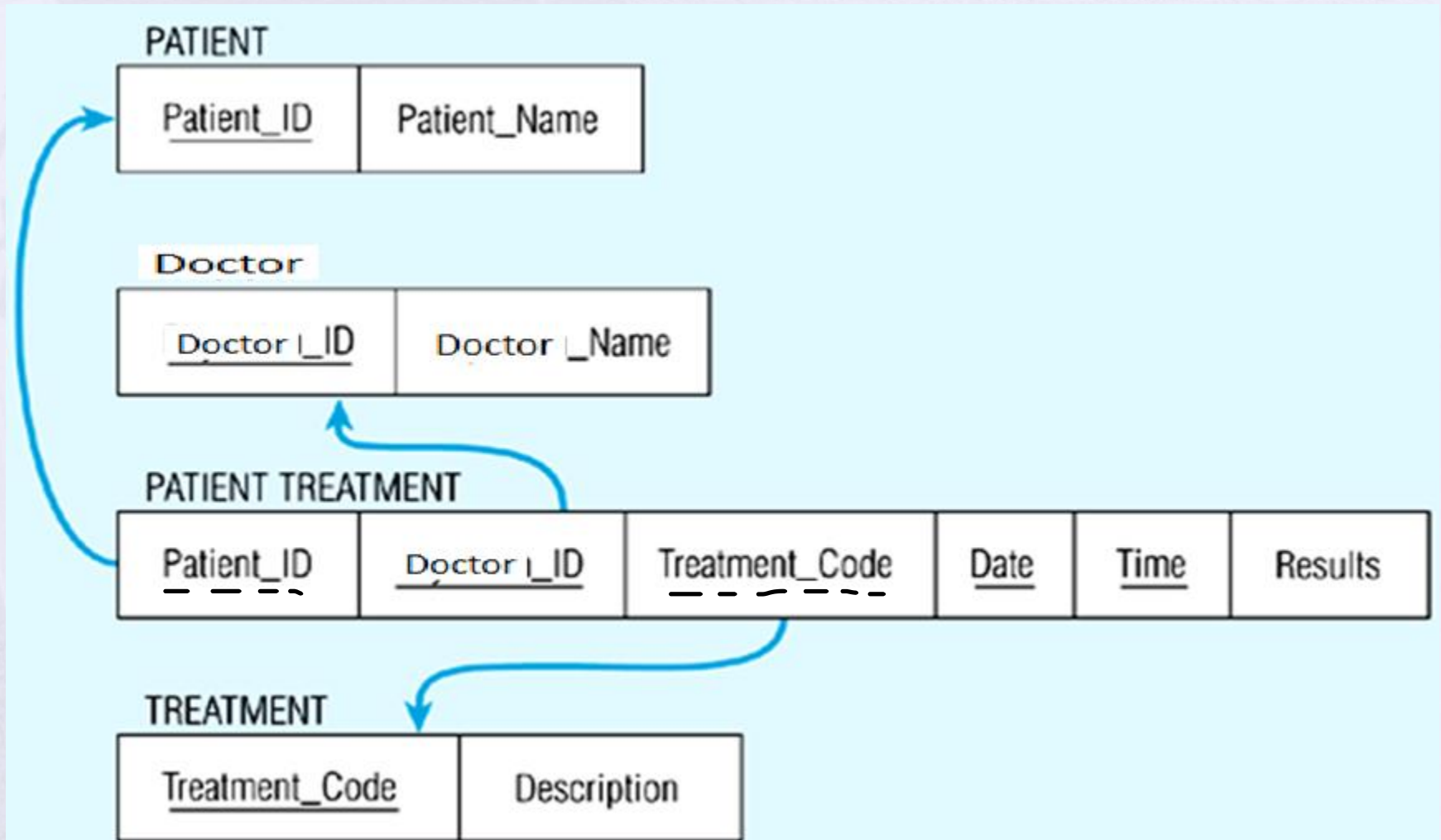
# Step 6: Mapping of N-ary Relationship Types.

- 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.

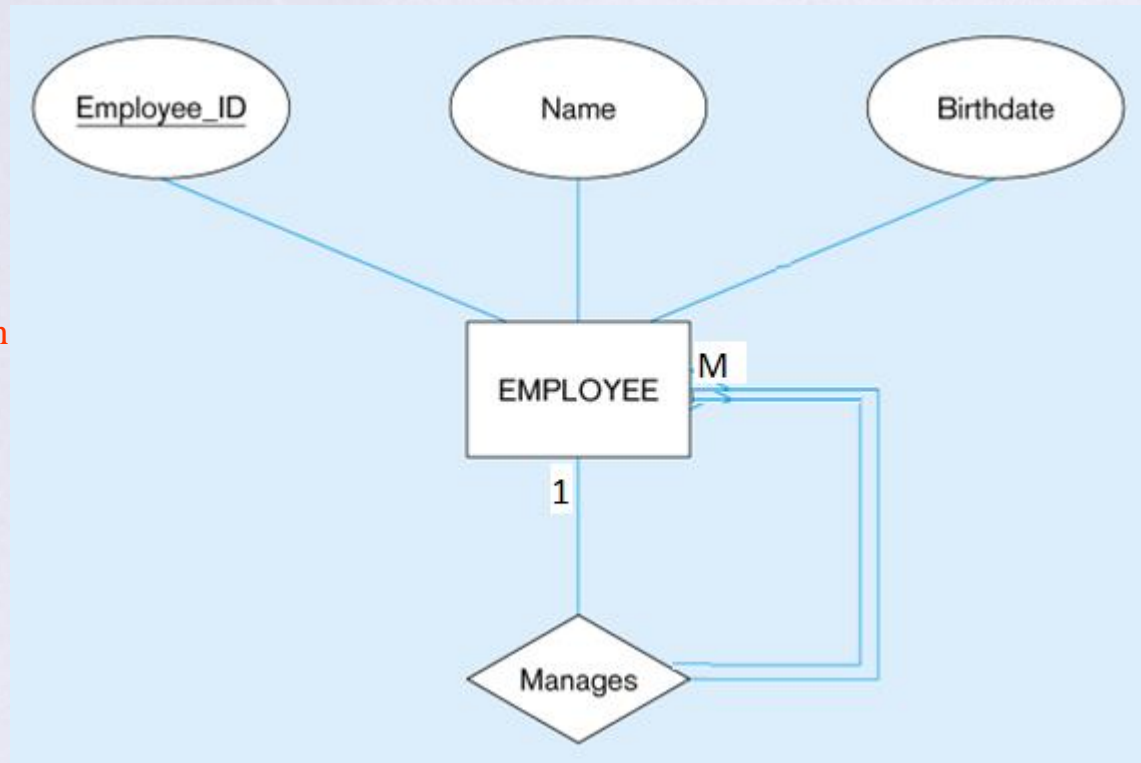


# Step 6: Mapping of N-ary Relationship Types.

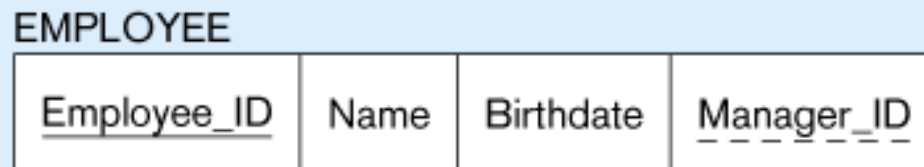


# Step 7: Mapping Unary Relationship

(a) EMPLOYEE entity with  
Manages relationship



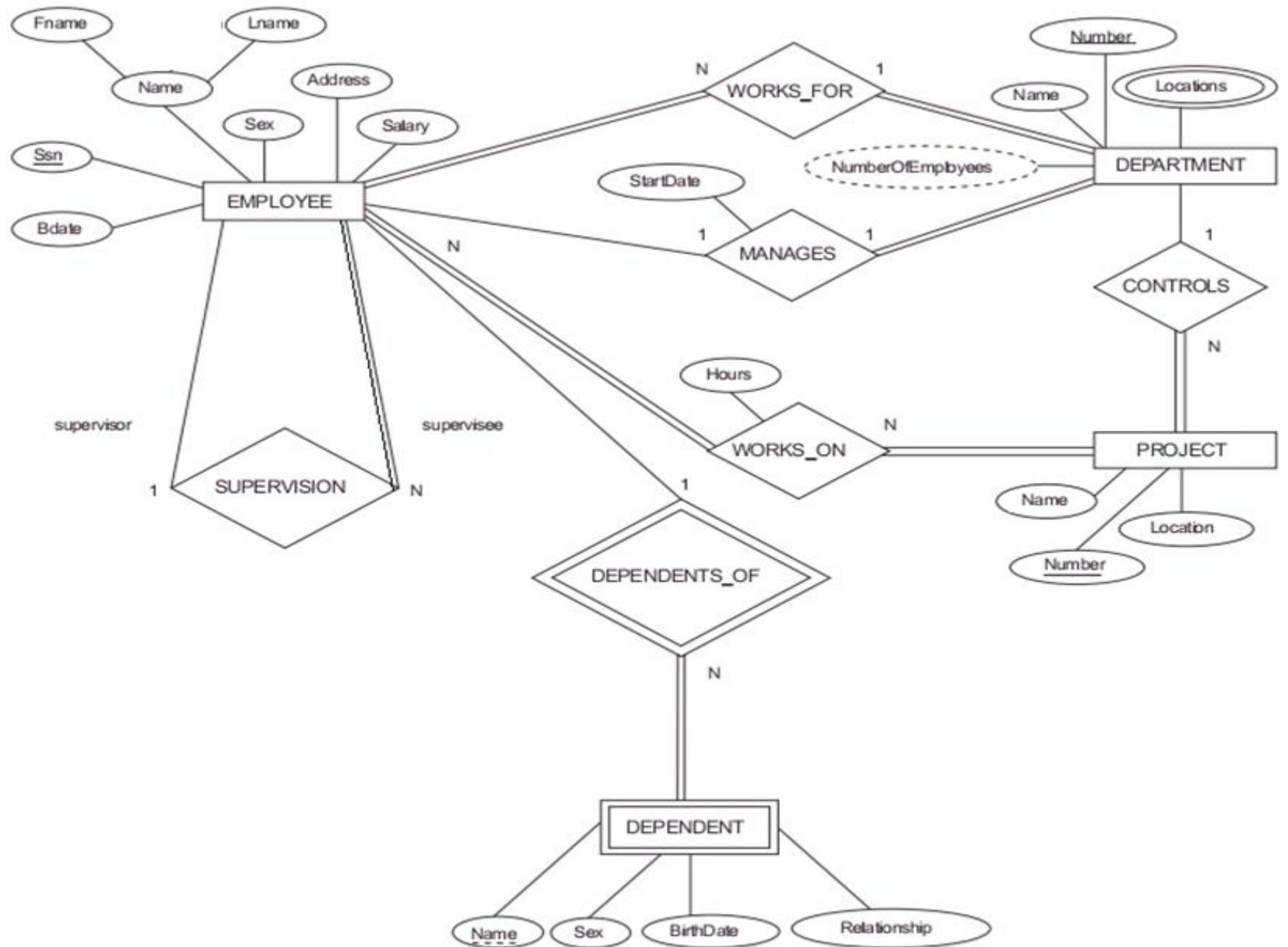
(b) EMPLOYEE  
relation with  
recursive foreign  
key



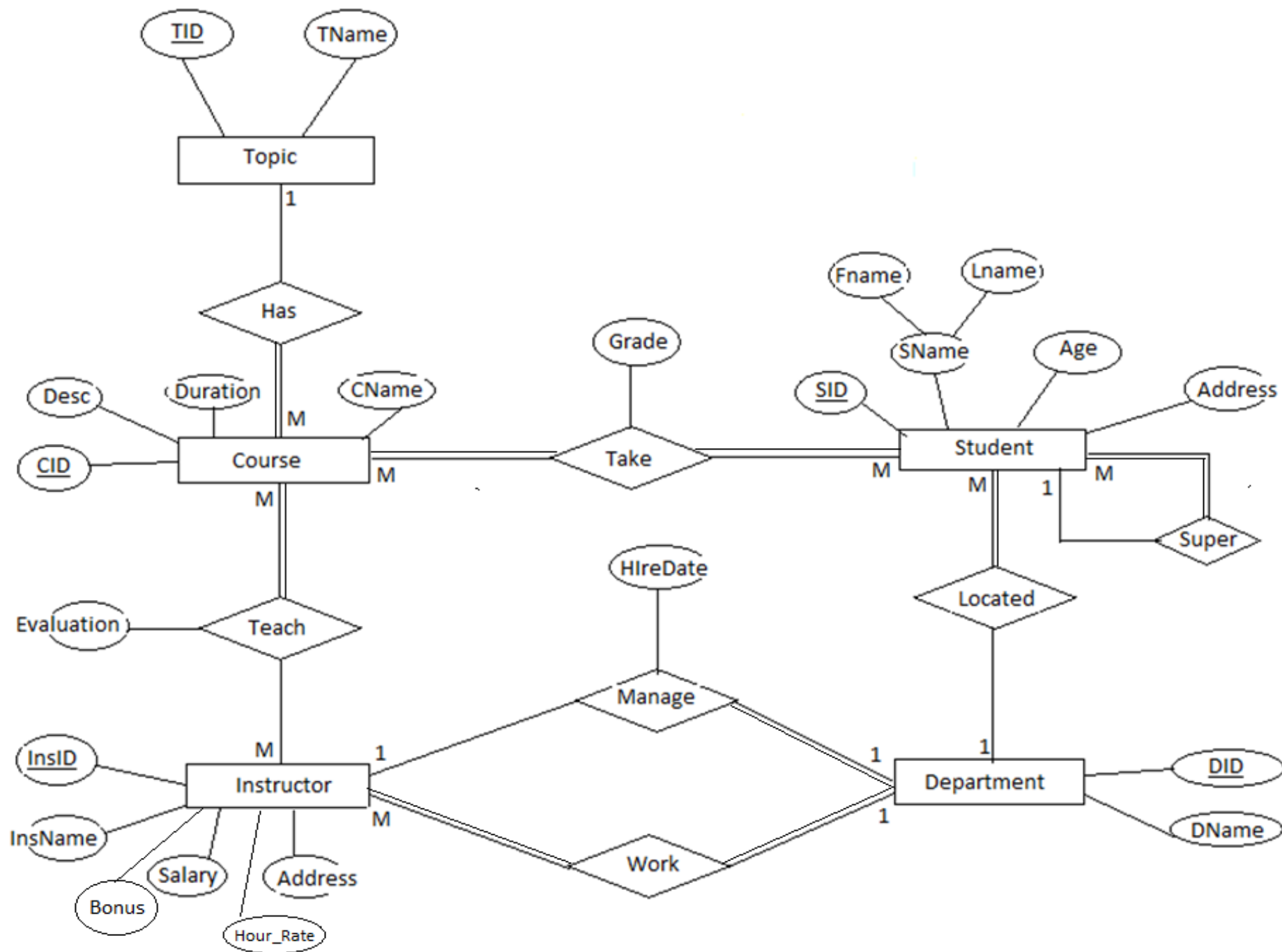


# Case Study









# 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**,Evaluation)
- Department(**Dept ID**,Dept\_Name,Manager\_ID,HireDate)

# Thank You !!!