course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Relational Databases

By

Dr.S.P. Siddique Ibrahim

VIT-AP University

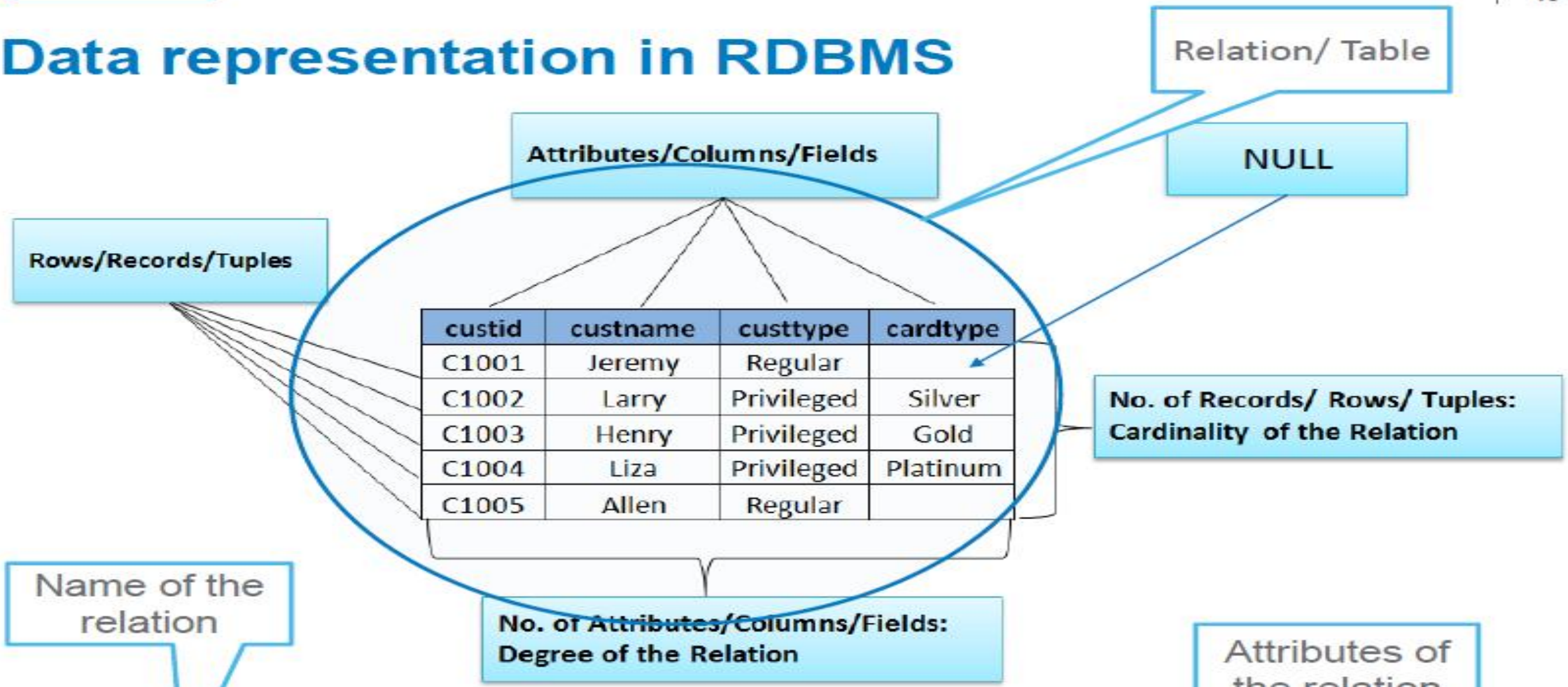
Amaravati

Relation/Tables

- A relational database is a set of formally described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables.
- Database consist of collection of tables/relations , each of which is assigned a unique name.

- Reference:
- https://searchdatamanagement.techtarget.com/definition/relational-database
- https://www.tutorialspoint.com/dbms/relational_data_model

Data representation in RDBMS



custid	custname	custtype	cardtype
C1001	Jeremy	Regular	
C1002	Larry	Privileged	Silver
C1003	Henry	Privileged	Gold
C1004	Liza	Privileged	Platinum
C1005	Allen	Regular	

What is in a relational database model?

- Each table, which is sometimes called a *relation*, in a relational database contains one or more data categories in columns, or *attributes/Fields*.
- A row in a table represents a *relationship* among a set of values.
- Since a table is a collection of such relationships, there is a close correspondence
- between the concept of *table* and the mathematical concept of *relation*, from which the relational data model takes its name.

course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

Figure 2.3 The *prereq* relation.

a row in the *prereq* table indicates that two courses are *related* in the sense that one course is a prerequisite for the other

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.

table *instructor*, a row in the table can be thought of as representing the relationship between a specified *ID* and the corresponding values for *name*, *dept name*, and *salary* values.

Relational database

TABLE (RELATION)

	Column (Attribute)	Column (Attribute)	Column (Attribute)
Row (Record or tuple)			
Row (Record or tuple)			

- Each row, also called a *record* or *tuple*, contains a unique instance of data, or *key*, for the categories defined by the columns

- Popular examples of relational databases include Microsoft SQL Server, Oracle Database, MySQL and IBM DB2.


LOOK A LIKE DRUGS
SOUND A LIKE DRUGS
SOCRIL

Keys

- Key is an attribute used to identify a record. It is used to establish and identify relation between tables.
- An attribute or set of attributes used to identify a row(tuple) in a relation(table).
- To find the relation between two tables.
- To uniquely identify a row in a table by a combination of one or more columns in that table.

Types of Keys

- Super Key
- Candidate Key
- Primary Key
- Foreign Key
- Alternate Key

Candidate Key in DBMS (Candidate key, Primary Key, Foreign Key, Alternate Key, Super Key)
https://www.youtube.com/watch?v=aSXdGPGcELo(Duration: 6:58)
https://www.youtube.com/watch?v=_rPhtARxZQ4(Duration: 9:56)

Super Key

- Set of one or more attributes to identify uniquely a tuple in the relation.
- May contain extraneous attributes
- Super keys in Student relation
  - **S.ID**
  - **Roll.No**
  - **Email**
  - **Phone No**
  - **(S.ID, Roll No)**
  - **(S.ID, Email)**
  - **(S.ID, Phone No)**
  - **(Roll.No, Email)**
  - **(Roll.No, Phone No)**
  - **(Email, Phone No)**
  - **(S.ID, Fname)**
  - **(S.ID, Lname)**
  - **(S.ID, Address)**
  - **--------**
  - **-------**
  - **(S.ID, Roll.no, Fname, Lanem, Email, Address, Phone No)**

S.ID	Roll. No	Fname	Lname	Email	Address	Phone No.
200	101	Nike	Son	nike@gmail.com	134, nestle street	7003498890
102	102	Mansoon	Linsen	man101@gmail.com	15, link mark layout	8755644556
203	103	Liu	Musker	liuettte@yahoo.com	64/39, thunderstreet	9907745873
204	104	Johnson	Linskey	linske@hotmail.com	8, Darkwhite home	8809233346
205	105	Watson	Mu	mu1245@gmail.com	93/023, Mic apartment	7222367767

Super Key

- Super keys in Account relation
  - (Acc. No.)
  - (CIF No., Acc. No.)
  - (CIF No., Acc. Type)
  - (Acc. No., Acc. Type)
  - (CIF No., Acc. No., Acc. Type)
  - (Acc. No., Acc. Type, Balance)
  - (CIF No., Acc. Type, Balance)
  - (CIF No., Acc. No., Acc. Type, Balance)

CIF No.	Acc. No.	Acc. Type	Balance
123456789	12451661234	Savings	30000
123459875	12451661456	Savings	40459
123434322	12451669823	Savings	23245
123456789	12451664567	Loan	100000
123459875	12451662328	Current	348776

Candidate Key

- Minimal number of super key by which a tuple can be identified is called a candidate key.
- Removal of any attribute from a candidate key would not yield unique identification of a tuple in a relation.
- Candidate keys in Account_Overview relation
  - (Acc. No.)
  - (CIF No., ~~Acc. No.)~~ 

Acc. No.	Acc. Holder Name
12451661234	Taylor
12451661456	Watson
12451669823	Henry
12451664567	Taylor
12451662328	Watson

Candidate Key

- Candidate keys in Account relation
  - (Acc. No.)
  - (CIF No., Acc. Type)

CIF No.	Acc. No.	Acc. Type	Balance
123456789	12451661234	Savings	30000
123459875	12451661456	Savings	40459
123434322	12451669823	Savings	23245
123456789	12451664567	Loan	100000
123459875	12451662328	Current	348776

S. No.	Primary Key	Candidate Key
1.	Primary key is a unique and non–null key which identify a record uniquely in table.	Candidate key is also a unique key to identify a record uniquely in a table.
2.	Primary key column value cannot be NULL.	Candidate key column can have NULL value.
3.	Primary key is most important part of any relation or table.	Candidate key signifies as which key can be used as Primary Key.
4.	Primary Key is a candidate key.	Candidate key may or may not be a primary key.
5.	A table can have only one primary key.	A table can have multiple candidate keys.

Primary Key

- A candidate key chosen by the database designer to uniquely identify tuples in a relation.
- Can not be NULL
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.
- Primay key in Account_Overview relation
  - (Acc. No.)

Acc. No.	Acc. Holder Name
12451661234	Taylor
12451661456	Watson
12451669823	Henry
12451664567	Taylor
12451662328	Watson

Primary Key

- Primary key in Account relation
  - Either (Acc. No.) or (CIF No., Acc. Type)
  - Choosing Acc. No. will be best

CIF No.	Acc. No.	Acc. Type	Balance
123456789	12451661234	Savings	30000
123459875	12451661456	Savings	40459
123434322	12451669823	Savings	23245
123456789	12451664567	Loan	100000
123459875	12451662328	Current	348776

Foreign Key

- An attribute which is primary key of another relation
- Referencing relation
- Referenced relation
- Referencing relation – (Ex: Account) -Foreign key
- Referenced relation – (Ex: Account_Overview) - Primary key

- Acc.No which is a primary key in Account_Overview relation is added as one attribute in Account relation. Hence Acc.No is Foreign key in Account relation

Alternate Keys

- All the keys which are not primary key are called an alternate key.
- It is a candidate key which is currently not the primary key.
- A table may have single or multiple choices for the primary key.

Constraints/Integrity Constraint

- Constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.
- These constraints are typically specified along with the **CREATE TABLE** statement.
- Constraints are classified into multiple types based on the number of columns they act upon as well as on the way they are specified.

Constraints

Various constraints that can be created on database tables are:

- NOT NULL
- PRIMARY KEY
- CHECK
- UNIQUE
- FOREIGN KEY

- We can also specify DEFAULT value for a column. Oracle database does not consider DEFAULT as a constraint.
-

CONSTRAINT	DESCRIPTION
NOT NULL	In MySQL NOT NULL constraint allows to specify that a column can not contain any NULL value. MySQL NOT NULL can be used to CREATE and ALTER a table.
UNIQUE	The UNIQUE constraint in MySQL does not allow to insert a duplicate value in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table.
PRIMARY KEY	A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint creates a unique index for accessing the table faster.
FOREIGN KEY	A FOREIGN KEY in MySQL creates a link between two tables by one specific column of both tables. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY.
CHECK	A CHECK constraint controls the values in the associated column. The CHECK constraint determines whether the value is valid or not from a logical expression.
DEFAULT	In a MySQL table, each column must contain a value (including a NULL). While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT.

Domain constraints-Check

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

Entity integrity constaint-NULL

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

Referential Integrity Constraints

(Table 1)

EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

Foreign key

Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined

Relationships

(Table 2)

Primary Key

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

Primary key constaint

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

NOT NULL

Syntax :
CREATE TABLE [table name] ([column name] [data type]([size]) [column constraint]…. [table constraint] ([[column name]……])……);

MySQL CREATE TABLE with NULL CONSTRAINT:

Using the default value as NOT NULL, while creating a MySQL table, it can be enforced that a column in a table is not allowed to store NULL values.

Example:
If you want to create a table 'newauthor' where no columns are allowed to store NULL VALUES the following statement can be used.

CREATE TABLE newauthor (aut_id varchar(8) NOT NULL, aut_name varchar(50) NOT NULL, country varchar(25) NOT NULL, home_city varchar(25) NOT NULL);
Here in the above statement the constraint 'NOT NULL' have been used to exclude the NULL VALUE.

MySQL CREATE TABLE to check values with CHECK CONSTRAINT

Adding a CHECK CONSTRAINT on a column of a table, you can limit the range of values allowed to be stored in that column.

Example

If you want to create a table 'newbook_mast' with a PRIMARY KEY on 'book _id' column, a unique constraint on 'isbn_no' column and a set the no_page in such, that it would hold values more than zero only, the following statement can be used.

```
CREATE TABLE
 newbook_mast (book_id varchar(15) NOT NULL UNIQUE,
 book_name varchar(50) ,
 isbn_no varchar(15) NOT NULL UNIQUE ,
 cate_id varchar(8) ,
aut_id varchar(8) ,
pub_id varchar(8) ,
dt_of_pub date ,
pub_lang varchar(15) ,
 no_page decimal(5,0) CHECK(no_page>0) ,
book_price decimal(8,2) ,
 PRIMARY KEY (book_id) );
```

Here in this MySQL statement will create a table 'newbook_mast' with a PRIMARY KEY on 'book _id' column, unique constraint on 'isbn_no' column and adding CHECK(no_page>0) will set the no_page in such, that it would hold values more than zero only.

MySQL CREATE TABLE with CHECK CONSTRAINT using IN operator:

MySQL CHECK CONSTRAINT can be applied to a column of a table, to set a limit for storing values within a range, along with IN operator.

Example

If you want to create a table 'newauthor' with a PRIMARY KEY on a combination of two columns (aut_id,home_city) and checking a limit value for the column country are 'USA','UK' and 'India', the following statement can be used.

```
CREATE TABLE IF NOT EXISTS newauthor(aut_id varchar(8) NOT NULL ,
aut_name varchar(50) NOT NULL,
 country varchar(25) NOT NULL CHECK (country IN ('USA','UK','India')),
home_city varchar(25) NOT NULL,
PRIMARY KEY (aut_id,home_city));
```

Here in the above MySQL statement will create a table 'newauthor' with a PRIMARY KEY on a combination of two columns (aut_id,home_city) and the value for the column country has been limited by using IN operator.

MySQL UNIQUE CONSTRAINT:

The UNIQUE constraint creates an index such that, all values in the index column must be unique.

An error occurs when any body tries to add a new row with a key value that already exists in that row.

Example

The MySQL statement stated below will create a table 'newauthor' with a column 'aut_id' which will store unique values only since UNIQUE (aut_id) is used.

```
CREATE TABLE IF NOT EXISTS newauthor(aut_id varchar(8) NOT NULL ,
 aut_name varchar(50) NOT NULL,
country varchar(25) NOT NULL,
 home_city varchar(25) NOT NULL,
 UNIQUE (aut_id));
```

MySQL CREATE TABLE with DEFAULT CONSTRAINT:

While creating a table, MySQL allows you assign DEFAULT CONSTRAINTS to columns.

DEFAULT is used to set a default value for a column and is applied using DEFAULT default_value;

where default_value is the default value set to the column.

Example:

The MySQL statement stated below will create a table 'newpublisher' with a PRIMARY KEY on 'pub_id' column, a CHECK constraint with logical operators for country and pub-city columns and a default value for pub_id, pub_name, pub_city and country columns.

The MySQL statement also sets the default value white space for pub_id, pub_name, pub_city columns and 'India' as a default value for a country column.

Example:

```
CREATE TABLE IF NOT EXISTS newpublisher (pub_id varchar(8) NOT NULL UNIQUE
DEFAULT '' ,
 pub_name varchar(50) NOT NULL DEFAULT '' ,
pub_city varchar(25) NOT NULL DEFAULT '' ,
 country varchar(25) NOT NULL DEFAULT 'India',
 country_office varchar(25) ,
no_of_branch int(3),
estd date CHECK ((country='India' AND pub_city='Mumbai') OR (country='India'
AND pub_city='New Delhi')) ,
PRIMARY KEY (pub_id));
```

MySQL CREATE TABLE with AUTO INCREMENT:

MySQL allows you to set AUTO_INCREMENT to a column. Doing so will increase the value of that column by 1 automatically, each time a new record is added.

Example:

The MySQL statement stated below will create a table 'newauthor' with a PRIMARY KEY on 'id' column and the 'id' column is an auto incremented field.

```
CREATE TABLE IF NOT EXISTS newauthor (id int NOT NULL AUTO_INCREMENT,
aut_id varchar(8),
aut_name varchar(50),
 country varchar(25),
home_city varchar(25) NOT NULL,
 PRIMARY KEY (id));
```

MySQL PRIMARY KEY CONSTRAINT:

•Usually, a table has a column or combination of columns that contain values used to uniquely identify each row in the table.

•This column or combination of columns is called PRIMARY KEY and can be created by defining a PRIMARY KEY CONSTRAINT while creating a table.

•A table can have only one PRIMARY KEY.

•A PRIMARY KEY column cannot contain NULL values.

Example:

The MySQL statement stated below will create a table 'newauthor' in which PRIMARY KEY set to the column aut_id.

CREATE TABLE IF NOT EXISTS newauthor(aut_id varchar(8) NOT NULL , aut_name varchar(50) NOT NULL, country varchar(25) NOT NULL, home_city varchar(25) NOT NULL, PRIMARY KEY (aut_id));

MySQL creating table with FOREIGN KEY CONSTRAINT:

•While creating (or modifying) a MySQL table, you can set a FOREIGN KEY CONSTRAINT to a column of the table.

•A foreign key is a column or combination of columns which can be used to set a link between the data in two tables.

• PRIMARY KEY of a table is linked to the FOREIGN KEY of another table to enhance data integrity.

Syntax :
FOREIGN KEY [column list] REFERENCES [primary key table] ([column list]);
Arguments:

Name	Description
column list	A list of the columns on which FOREIGN KEY is to be set.
REFERENCES	Keyword.
primary key table	Table name which contains the PRIMARY KEY.
column list	A list of the columns on which PRIMARY KEY is set in the primary key table.

Example:

If you want to do the following tasks:
A new table 'newbook_mast' will be created.
The PRIMARY KEY for that table 'newbook_mast' is 'book_id'.
The FOREIGN KEY for the table 'newbook_mast' is 'aut_id'.
The 'aut_id' is the PRIMARY KEY for the table 'newauthor'.
The FOREIGN KEY 'aut_id' for the table 'newbook_mast' points to the PRIMARY KEY 'aut_id' of the table 'newauthor'.

That means the 'aut_id's which are present in the 'newauthor' table, only those authors will come to the 'newbook_mast' table.

Here is the MySQL statement below for the above tasks:

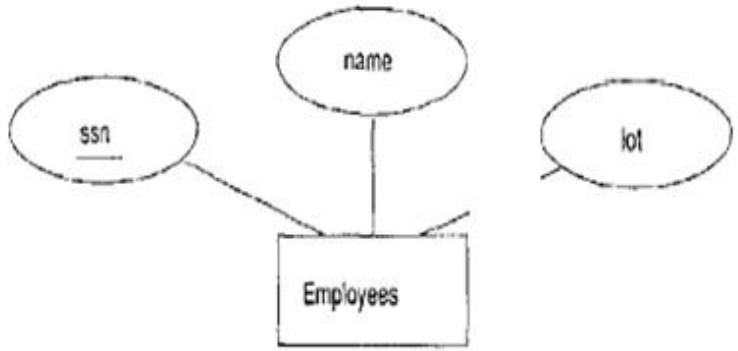
```
CREATE TABLE IF NOT EXISTS newbook_mast (book_id varchar(15) NOT NULL
PRIMARY KEY,
 book_name varchar(50) ,
 isbn_no varchar(15) NOT NULL ,
 cate_id varchar(8) ,
 aut_id varchar(8) ,
pub_id varchar(8) ,
 dt_of_pub date ,
pub_lang varchar(15) ,
no_page decimal(5,0) ,
 book_price decimal(8,2) ,
 FOREIGN KEY (aut_id) REFERENCES newauthor(aut_id));
```

Entity Sets to Database Tables

Each attribute of the entity set becomes an attribute of the table.

Note that we know both the domain of each attribute and the (primary) key of an entity set.

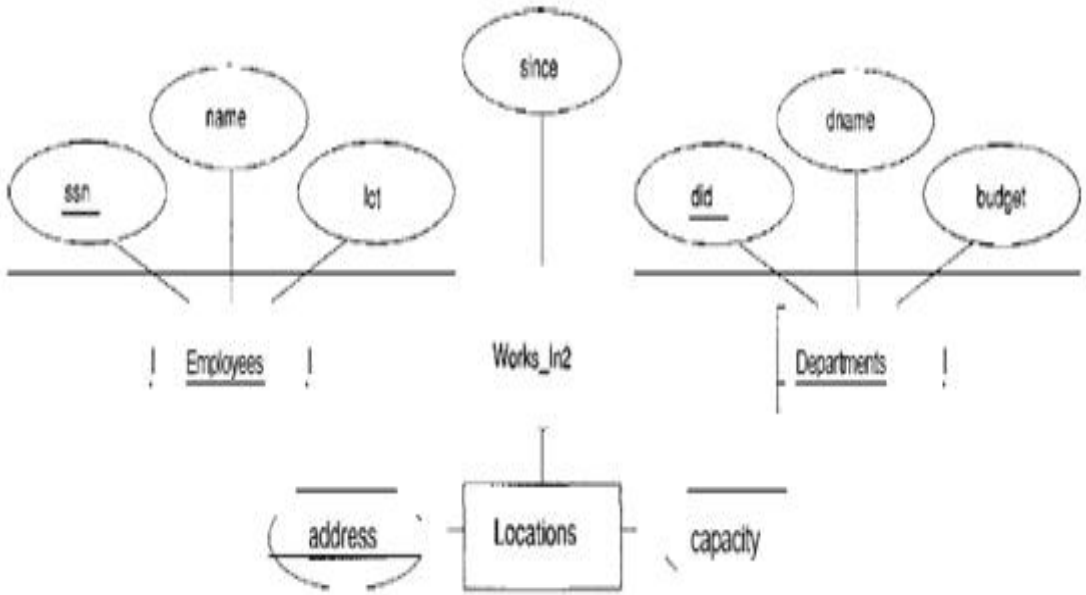
Entity Sets to Database Tables




```
CREATE TABLE Employees ( ssn CHAR(11),
                         name CHAR(30) ,
                         lot INTEGER,
                         PRIMARY KEY (ssn) )
```

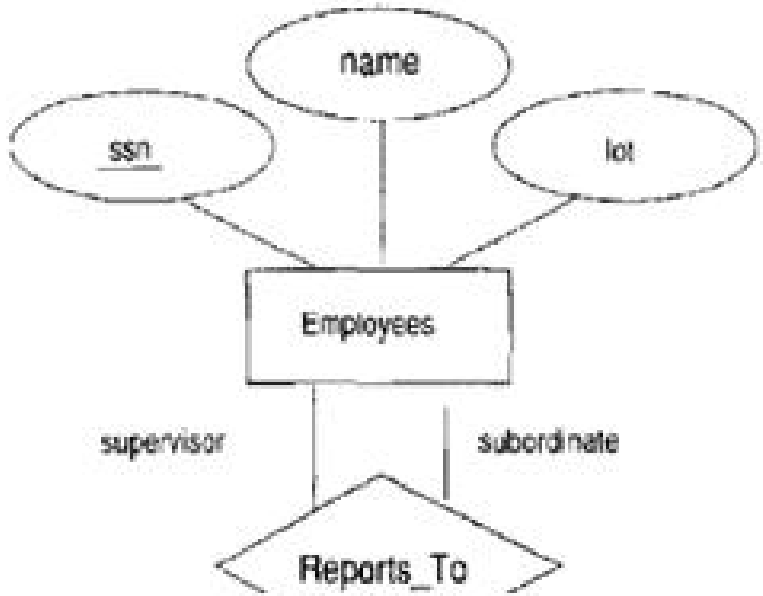
ssn	*name*	*lot*
123-22-3666	Attishoo	48
231-31-5368	Smiley	22
131-24-3650	Smethurst	35

Relationship Sets to Tables



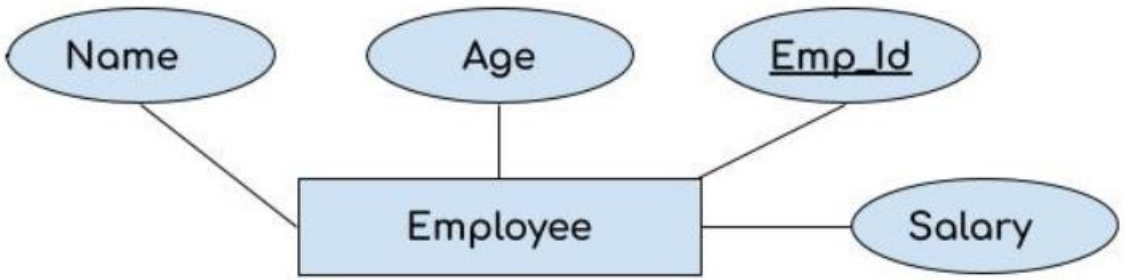

```
CREATE TABLE Works_In2 (      ssn CHAR(11),
                              did INTEGER,
                              address CHAR(20) ,
                              since DATE,
                              PRIMARY KEY (ssn, did, address),
                              FOREIGN KEY (ssn) REFERENCES Employees,
                              FOREIGN KEY (address) REFERENCES Locations,
                              FOREIGN KEY (did) REFERENCES Departments
                              )
```

Relationship Sets to Tables

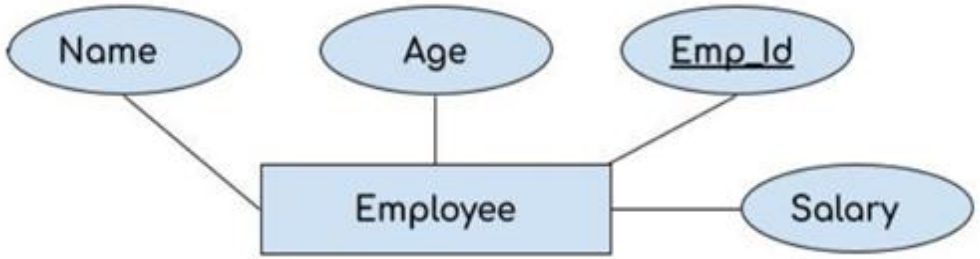


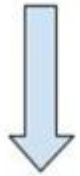

```
CREATE TABLE Reports_To (Supervisor_ssn CHAR (11),
                         Subordinate_ssn CHAR (11) ,
                         PRIMARY KEY (supervisor_ssn, subordinate_ssn),
                         FOREIGN KEY (supervisor_ssn) REFERENCES
                         Employees(ssn),
                         FOREIGN KEY (subordinate_ssn) REFERENCES
                         Employees(ssn) )
```

GENERAL EXAMPLS
(E-R to DATABASE)



E-R to DATABASE Solution

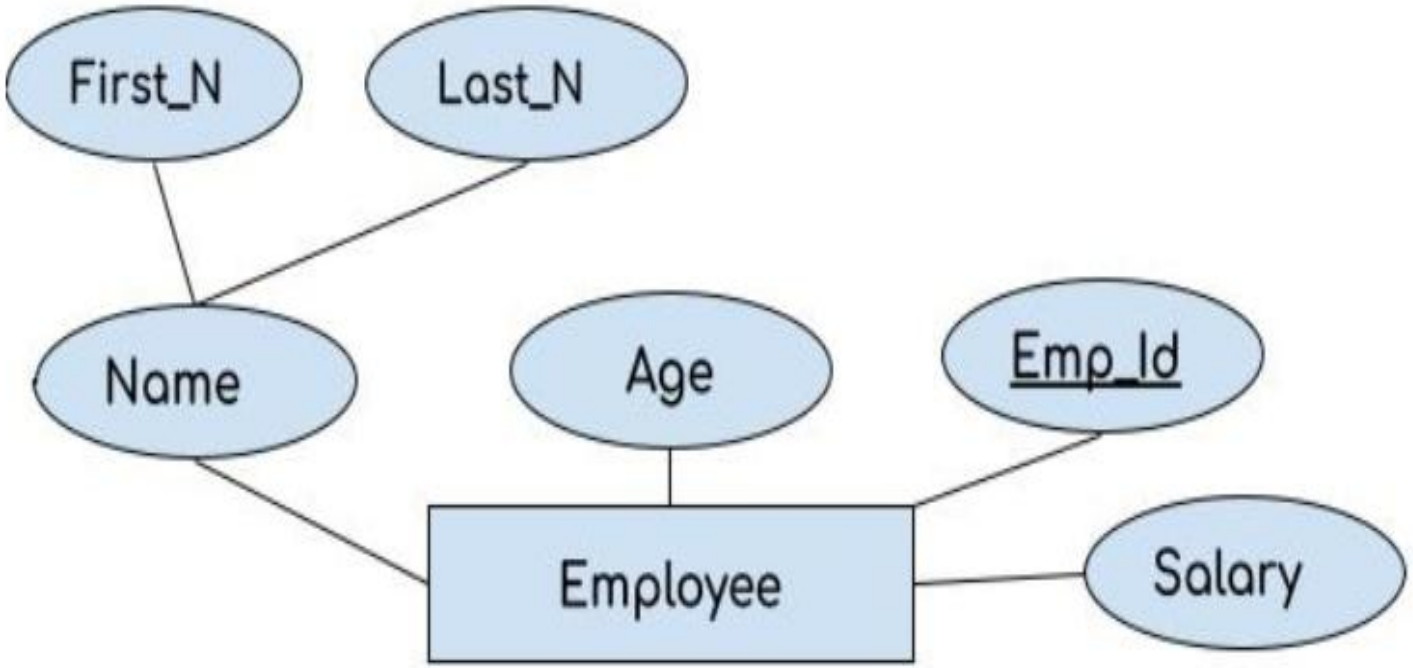




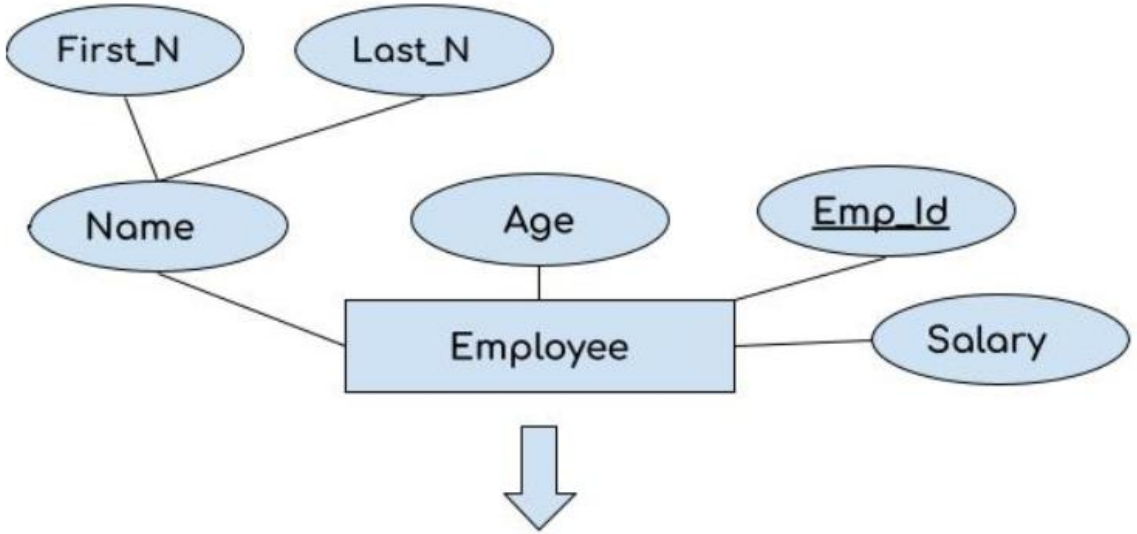
Emp_Id	Name	Age	Salary

Table Schema: (Emp_id, Name, Age, Salary)

GENERAL EXAMPLS
(E-R to DATABASE)



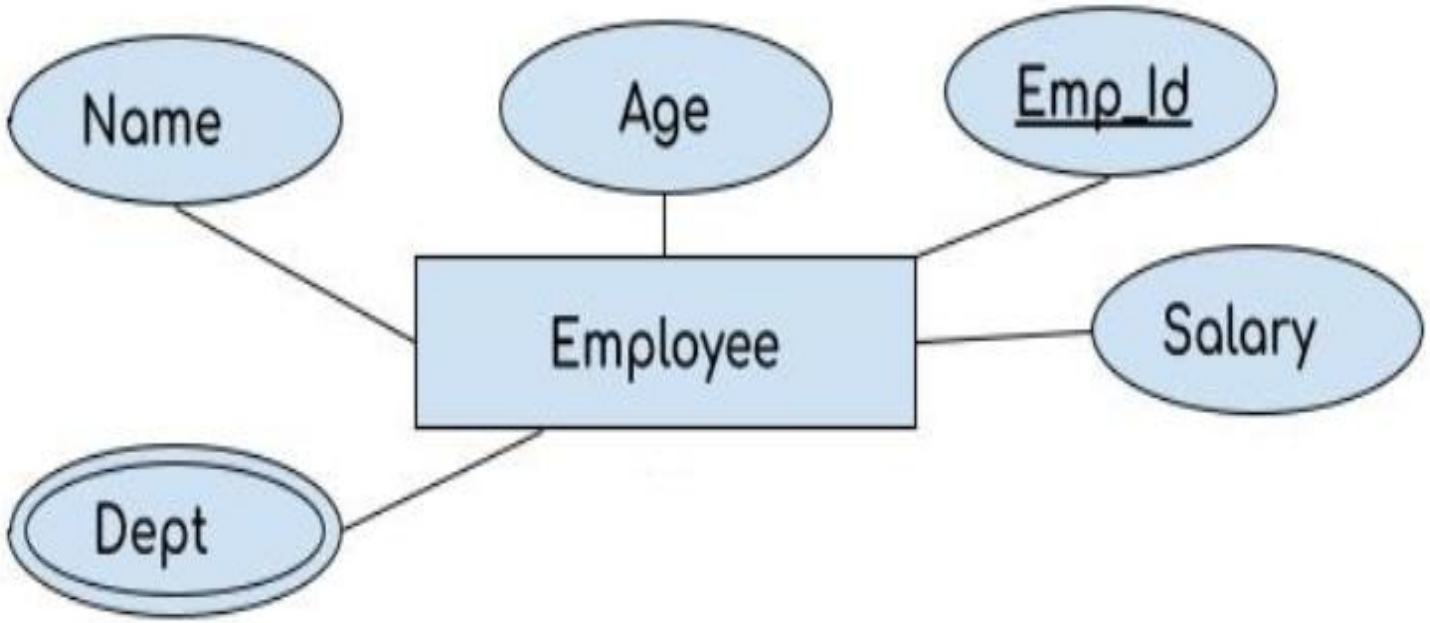
E-R to DATABASE Solution



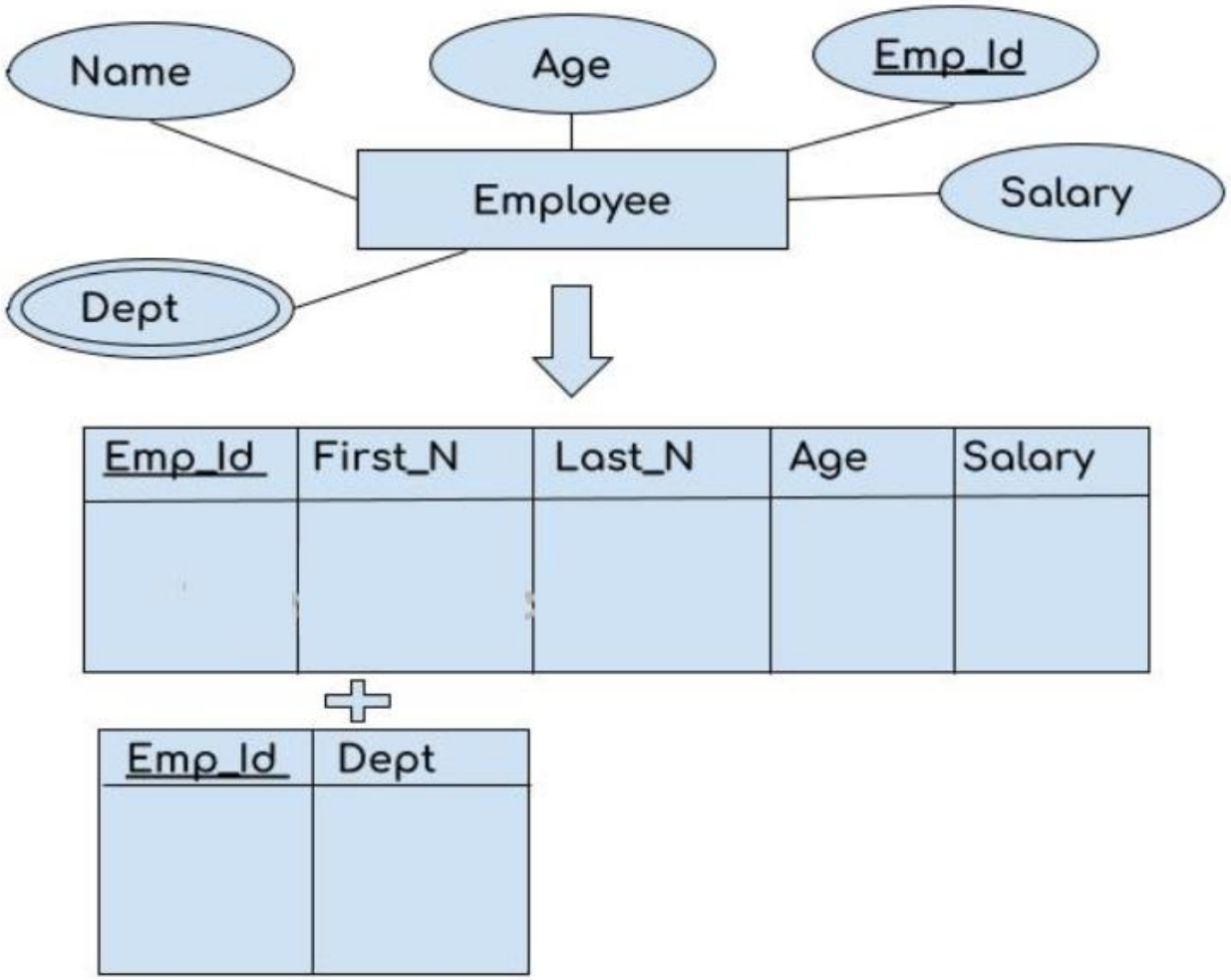
Emp_Id	First_N	Last_N	Age	Salary

Table Schema: (Emp_id, First_N, Last_N, Age, Salary)

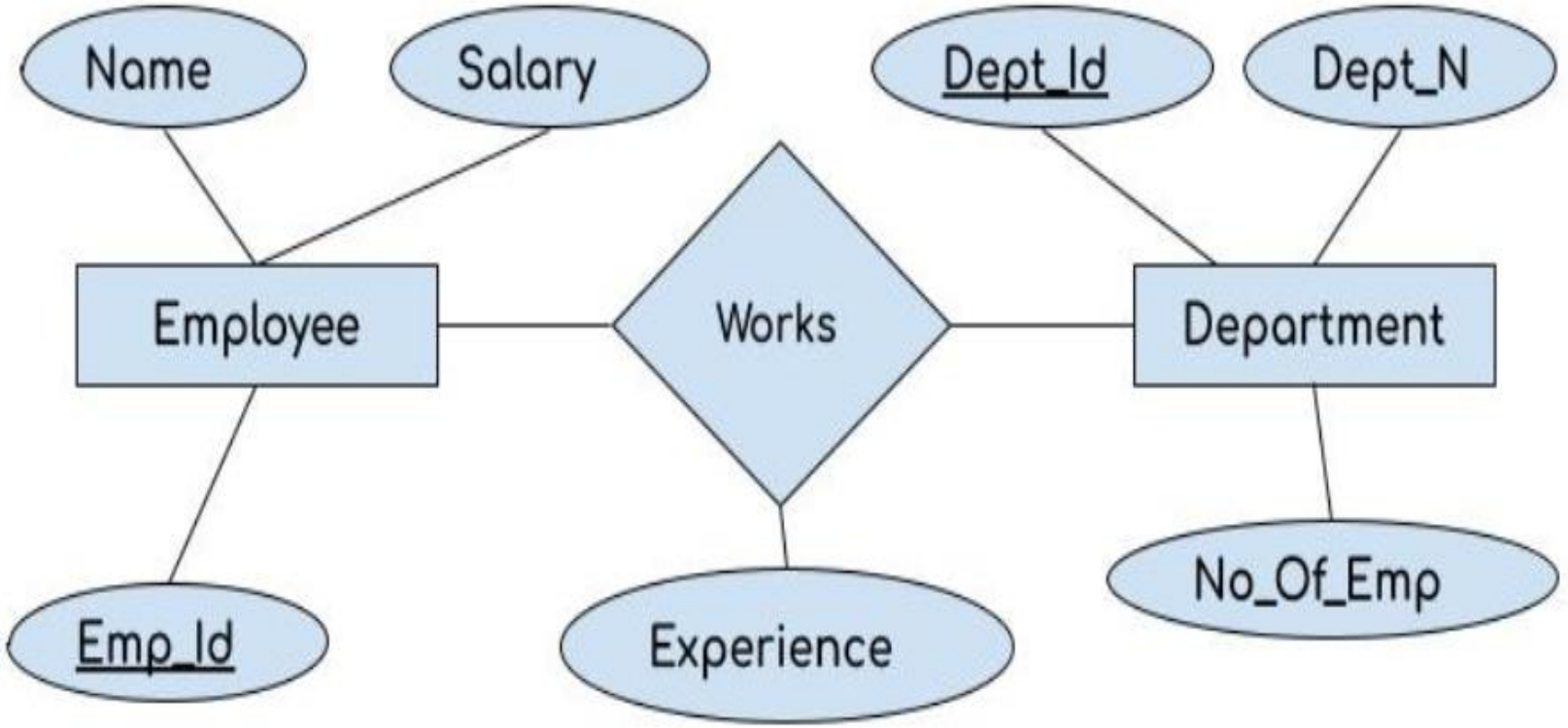
GENERAL EXAMPLS
(E-R to DATABASE)



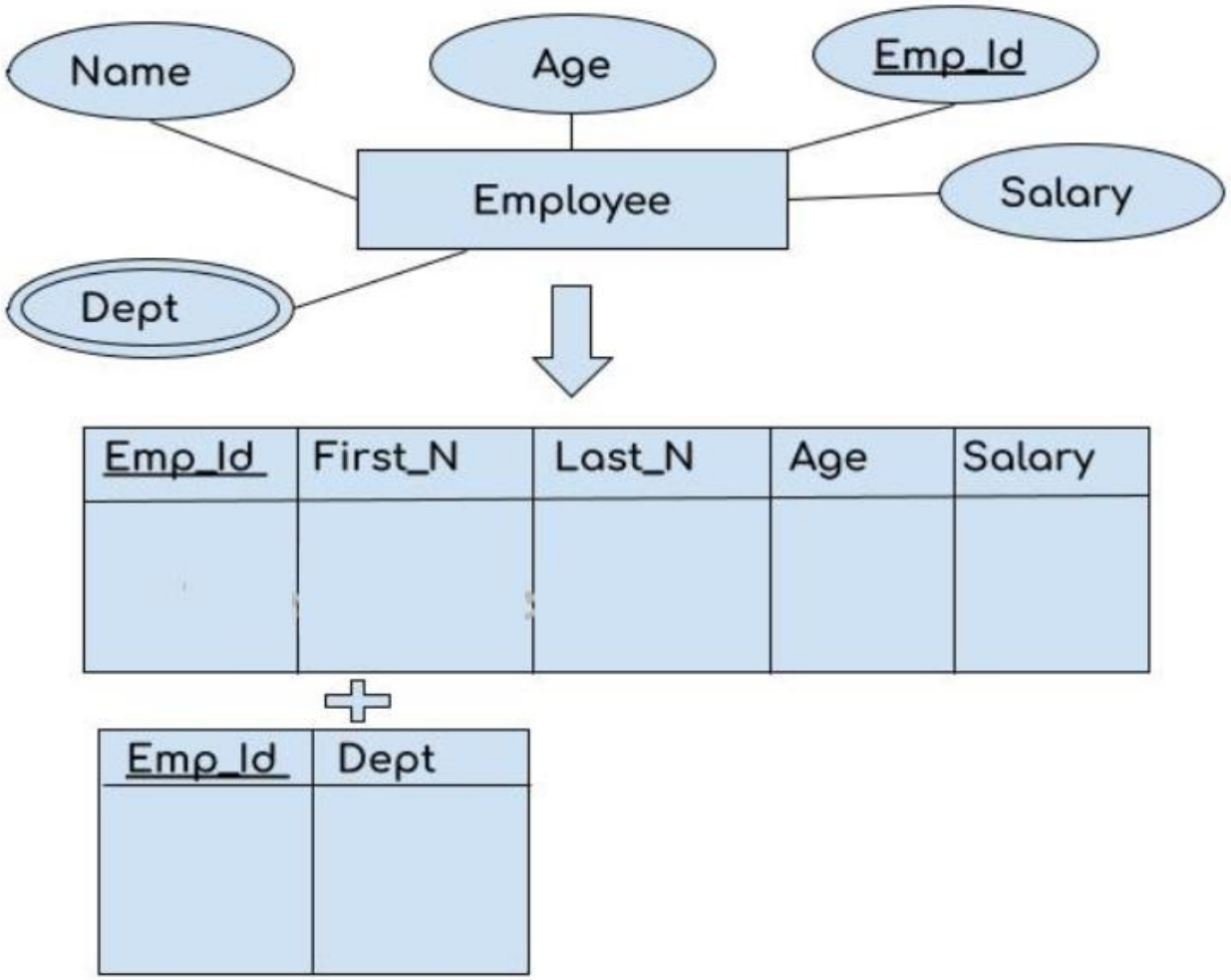
E-R to DATABASE
Solution



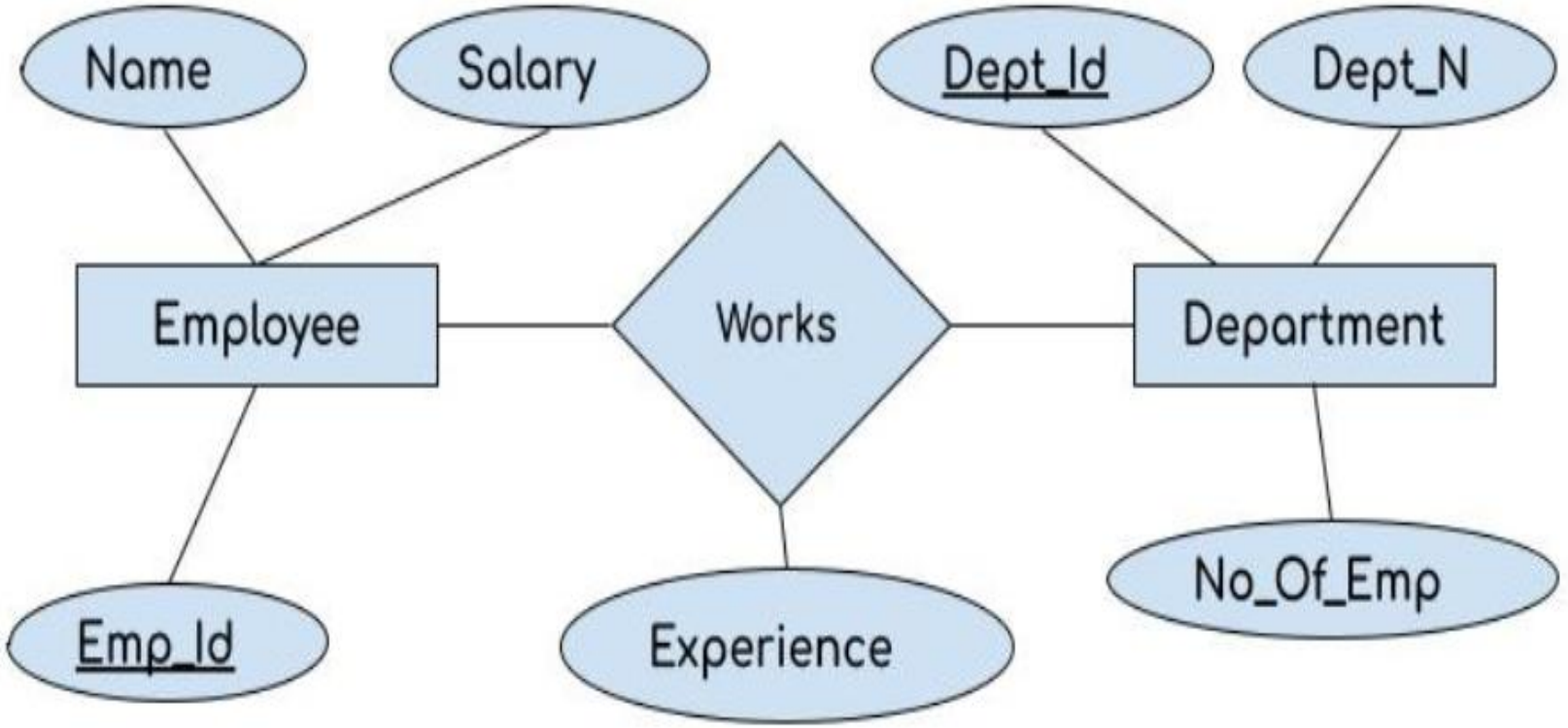
GENERAL EXAMPLS
(E-R to DATABASE)



E-R to DATABASE
Solution



GENERAL EXAMPLS
(E-R to DATABASE)



Initial Design of Entity Types: EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

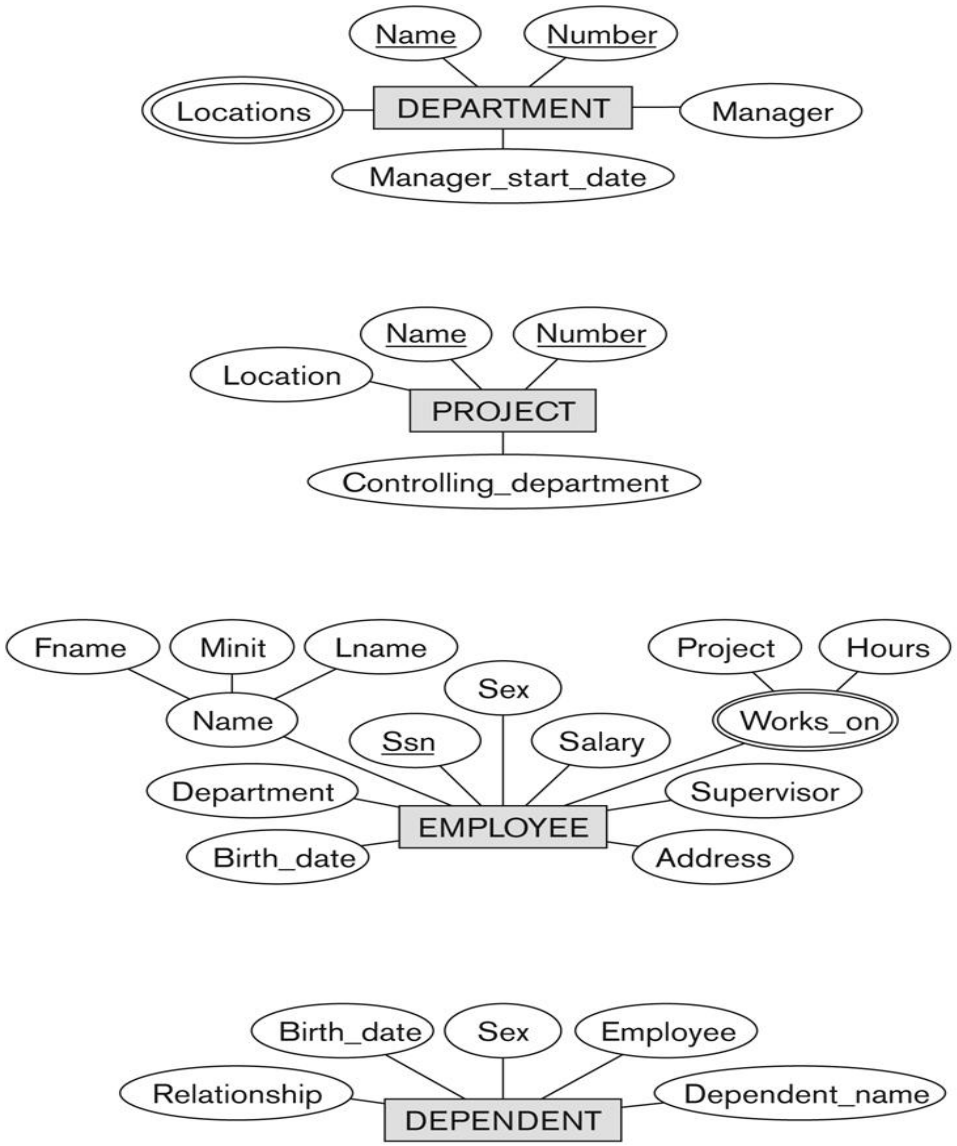


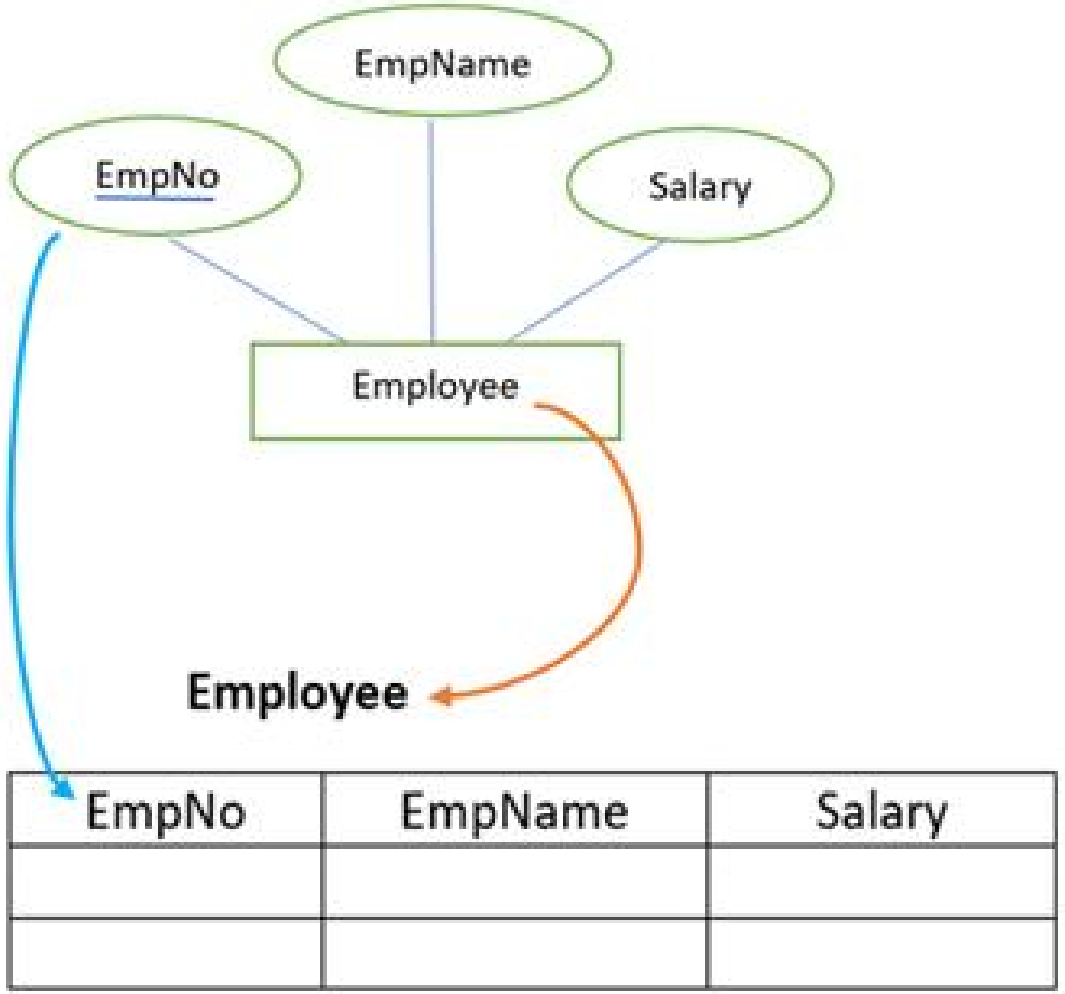
Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Conversion of E-R Diagram into Relational Model

- In general conversion of E-R diagram into a relational model involves the following:

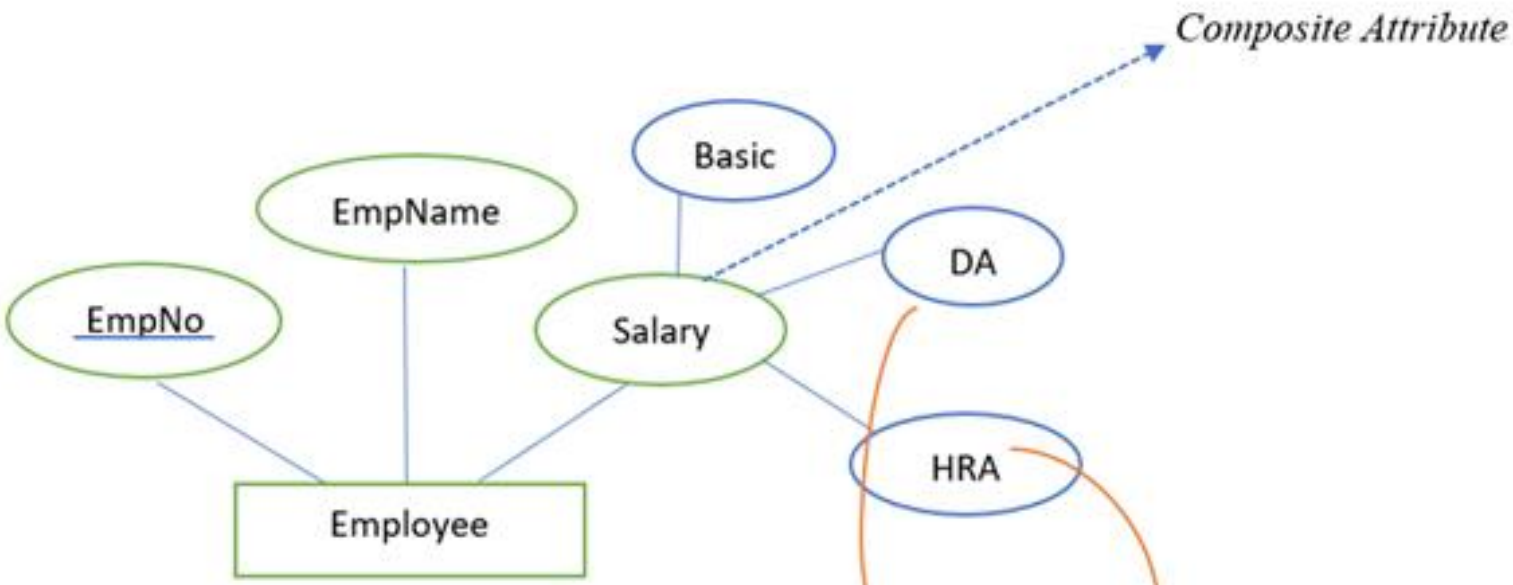
- Mapping of an entity set into relation (tables) of the database.
- The attributes of a table include the attributes of an entity
- The key attribute of an entity becomes the primary key of the relation
- For example,

- Consider the following E-R diagram in the figure below. The E-R diagram consists of Employee as an entity set and EmpNo, EmpName, and Salary as its attributes. Here we map entity set into a relation Employee and attributes of an entity set will become the attributes inside the table. The key attribute will become the primary key of the table.



Conversion of a simple E-R diagram into a Table

Entity set with a composite attribute

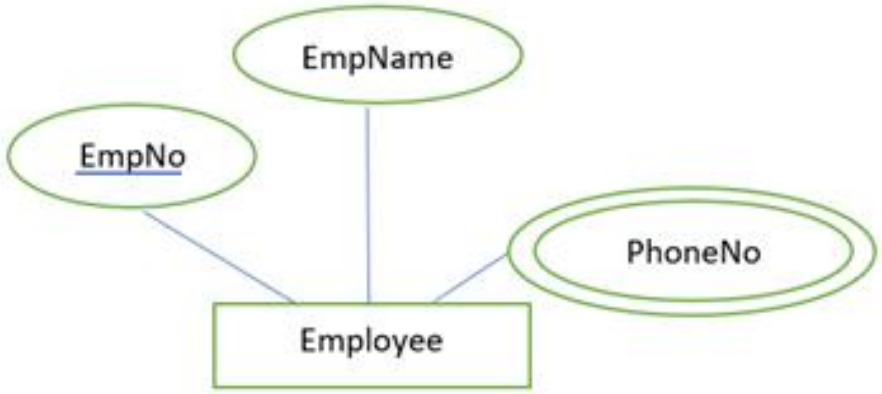


Employee

EmpNo	EmpName	Basic	DA	HRA

Conversion of an E-R diagram containing composite attribute

Entity set with multivalued attributes



E-R diagram containing multivalued attribute

- If we include the PhoneNo in the table with all other attributes, then for a single-valued tuple we may have multiple entries as shown in the table below

Employee

EmpNo	EmpName	PhoneNo
1	A	9821
1	A	9780
2	B	1234
....		

Duplicate values with multivalued attribute

Employee

EmpNo	EmpName	PhoneNo
1	A	9821
1	A	9780
2	B	1234
....		

Duplicate values with multivalued attribute

However, to avoid duplicate values in the table, we split the attributes into two different relations as shown in the figure below

EmpNo	EmpName

EmpNo	PhoneNo

Conversion of multivalued attributes into relation

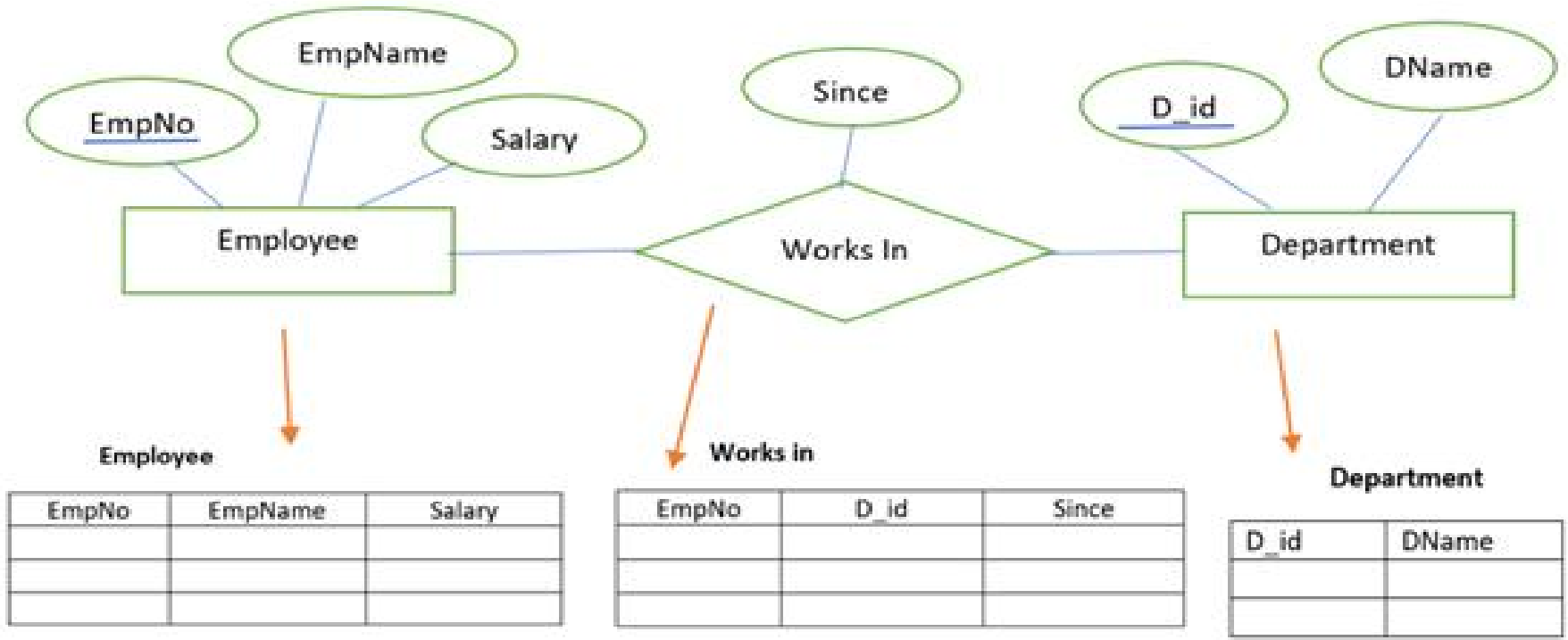
Translation of a relationship into a relation

- Likewise, we map the entity set into the relation in a relational model, we can also map a relationship set into a relation.

- The attribute of such a relation includes key attributes of the participating relations.

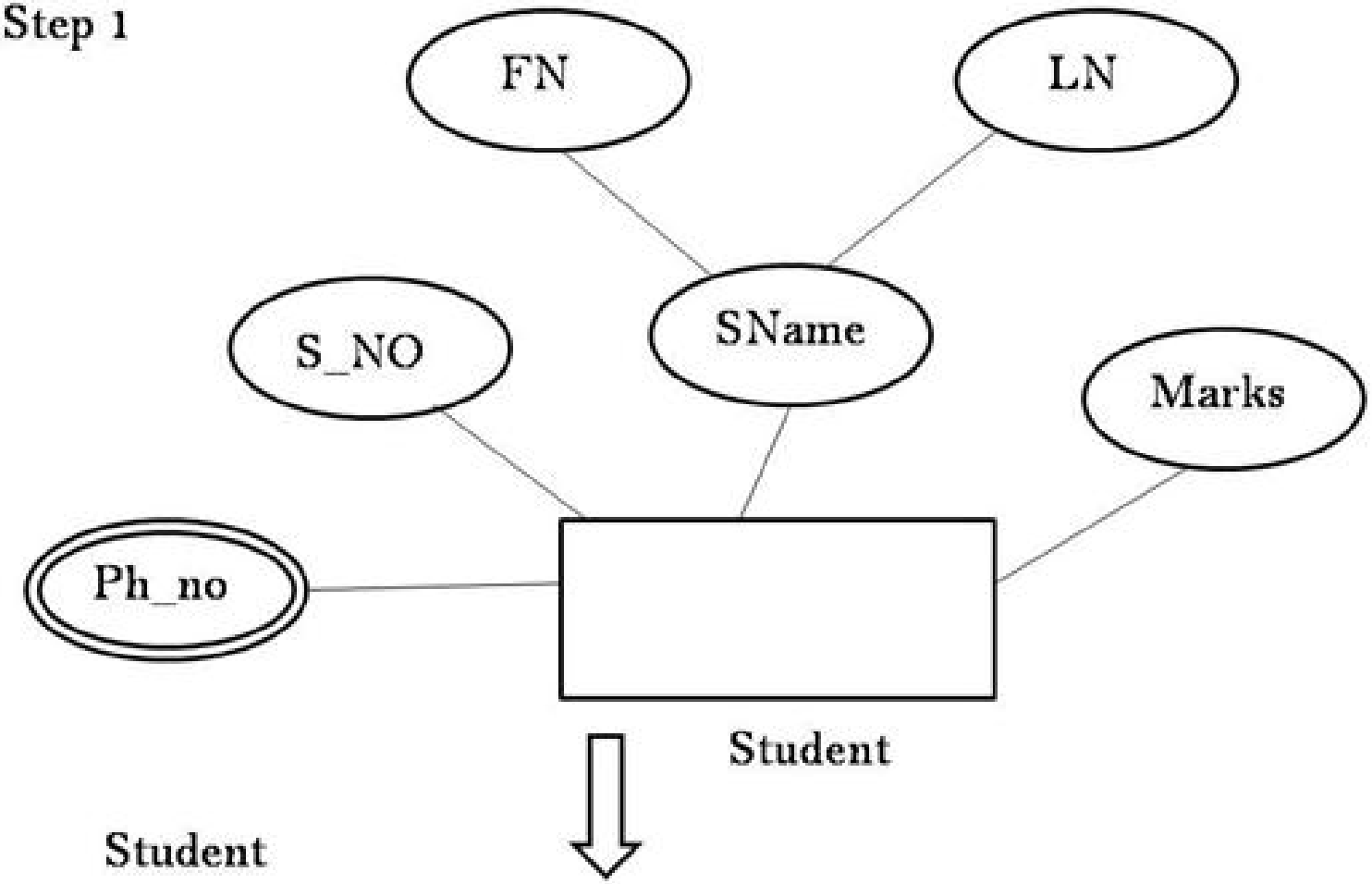
- The attributes are will become a foreign key.

For example, in the figure given below, there are two entity sets *Employee* and *Department*. These entity sets are participating in a relationship *works in*. The relationship set is converted into relation with attributes EmpNo from *Employee* relation, D_id from *Department* relation and *Since*, the attribute of the relationship set itself.



Translation of a relationship into a relation

Step 1



Student

S_NO	Marks	FN	LN

We do not include **sname**, only include single attribute **FN** and **LN**. Here in this table, **we are not considering attribute ph_no** from ER diagram because **ph_no** is multi-valued**.**

			X
S_NO	FN	LN	Ph-No

So, for **Ph_No** we can create **another table.**

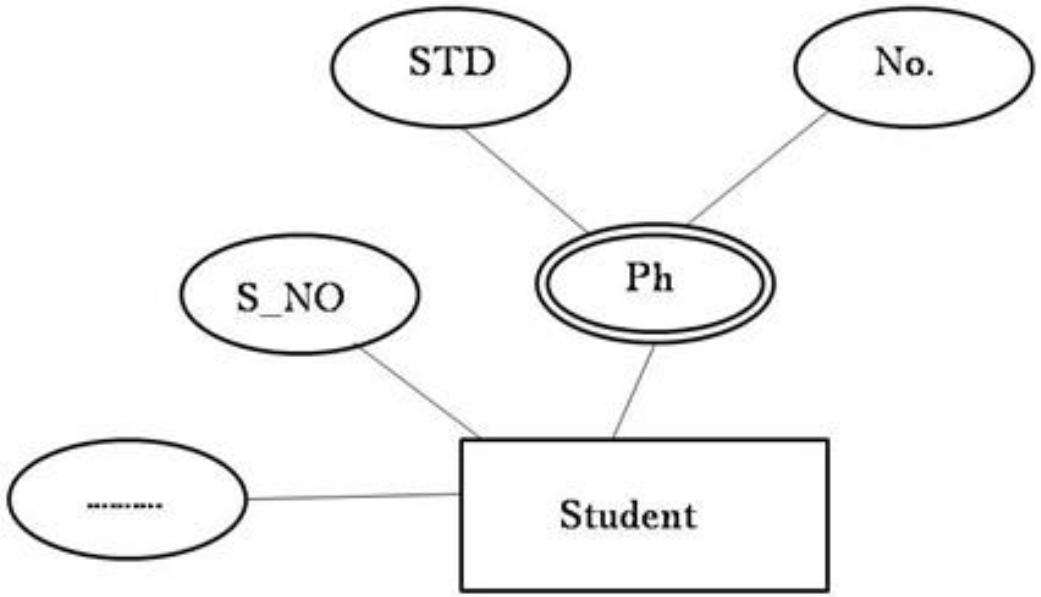
FK

S_NO	Ph_no
1	P1
1	P2
1	P3
2	P1
2	P2
3	P1

Here two may have duplicate value because a student has more than one phone number and two students may share the same room. So, better to make both as a primary as combined.

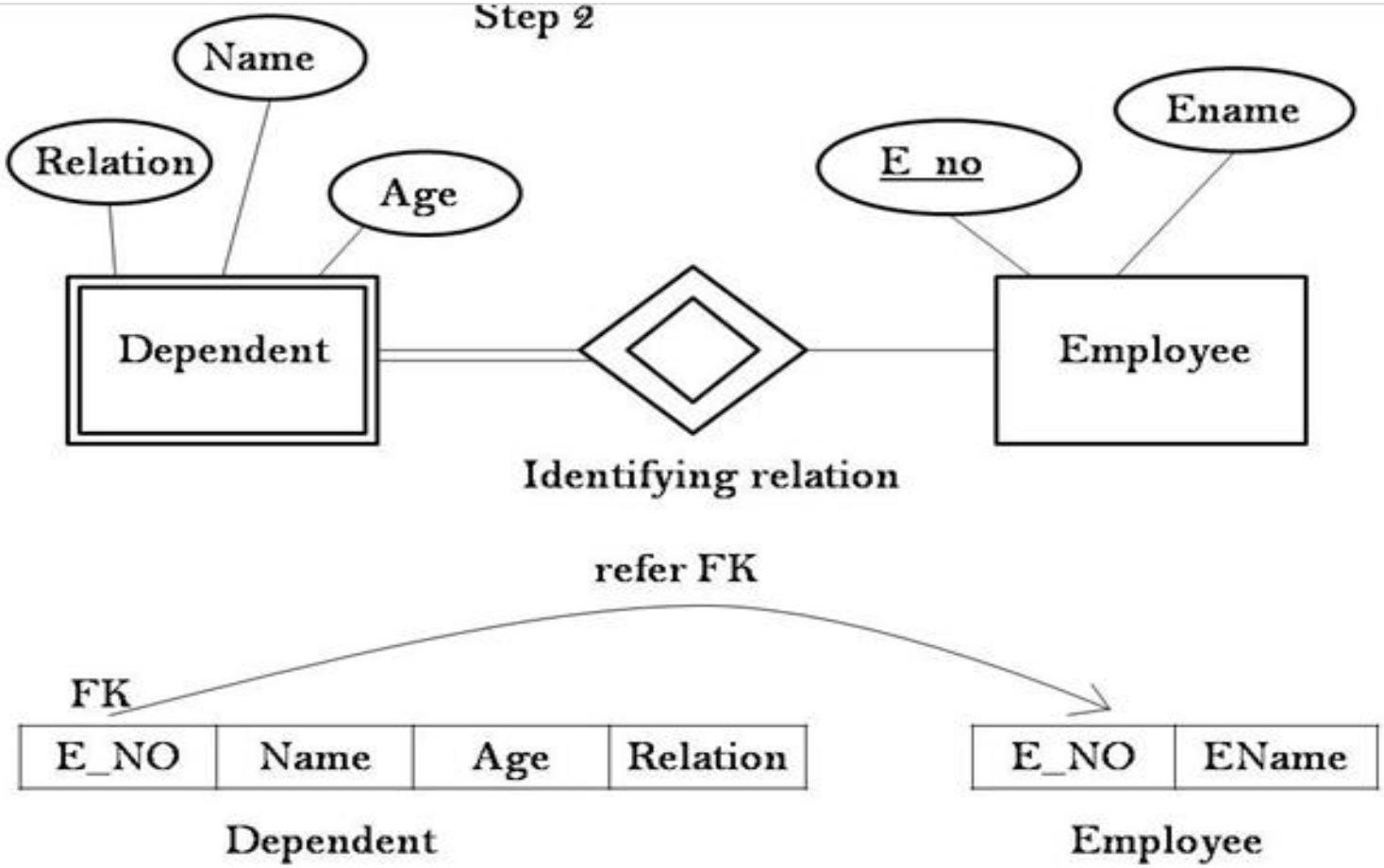
Sometimes Ph_No may be complex attribute also.

Sometimes Ph_No may be complex attribute also.



In that case same way in the new table, we include **STD** and **NO.**, not Ph_No**.**

S_NO	STD	NO.



In the case of the **weak entity** which depends on the **employee table** (relation), we should include **E_No** (**primary key** of employee table) at the **weak entity** (dependent) as a **foreign key.** And Name at the dependent table let work as the **partial key.**

Sample Question

- Consider the following tables:
- Course (Course_id,Course_name)

- Teacher (Teacher_id,Teacher_name)

- Assigned_to (Teacher_id, Course_id)

- a) How many tables will be created using the above scenario?

- b) What will be the foreign key?