

# Module 2 Syllabus

- Structure of Relational Databases – Integrity Constraints, Synthesizing ER diagram to relational schema
- Introduction to Relational Algebra - select, project, cartesian product operations, join - Equi-join, natural join. query examples,
- Introduction to Structured Query Language(SQL), Data Definition Language (DDL), Table definitions and operations – CREATE, DROP, ALTER, INSERT, DELETE, UPDATE.

# Relational Model Concepts

The relational model organizes data into a structured form using tables, also known as **relations**.

# Relational data model

- Represent database as a collection of **relations**
- Relation is a **table** which has values and rows in table is a collection of related data values
- Row in relational table is called a **tuple**, column is **attribute** and table is a **relation**

**Attributes**

**Relation name**

**EMPLOYEE**

EMP_NO	Name	Address	Mobile number	Age	Salary
101	RAM	XYZ	9898989898	20	10000
102	SAM	CVF	9999999999	21	20000
103	SITA	FDFD	8888888888	22	30000

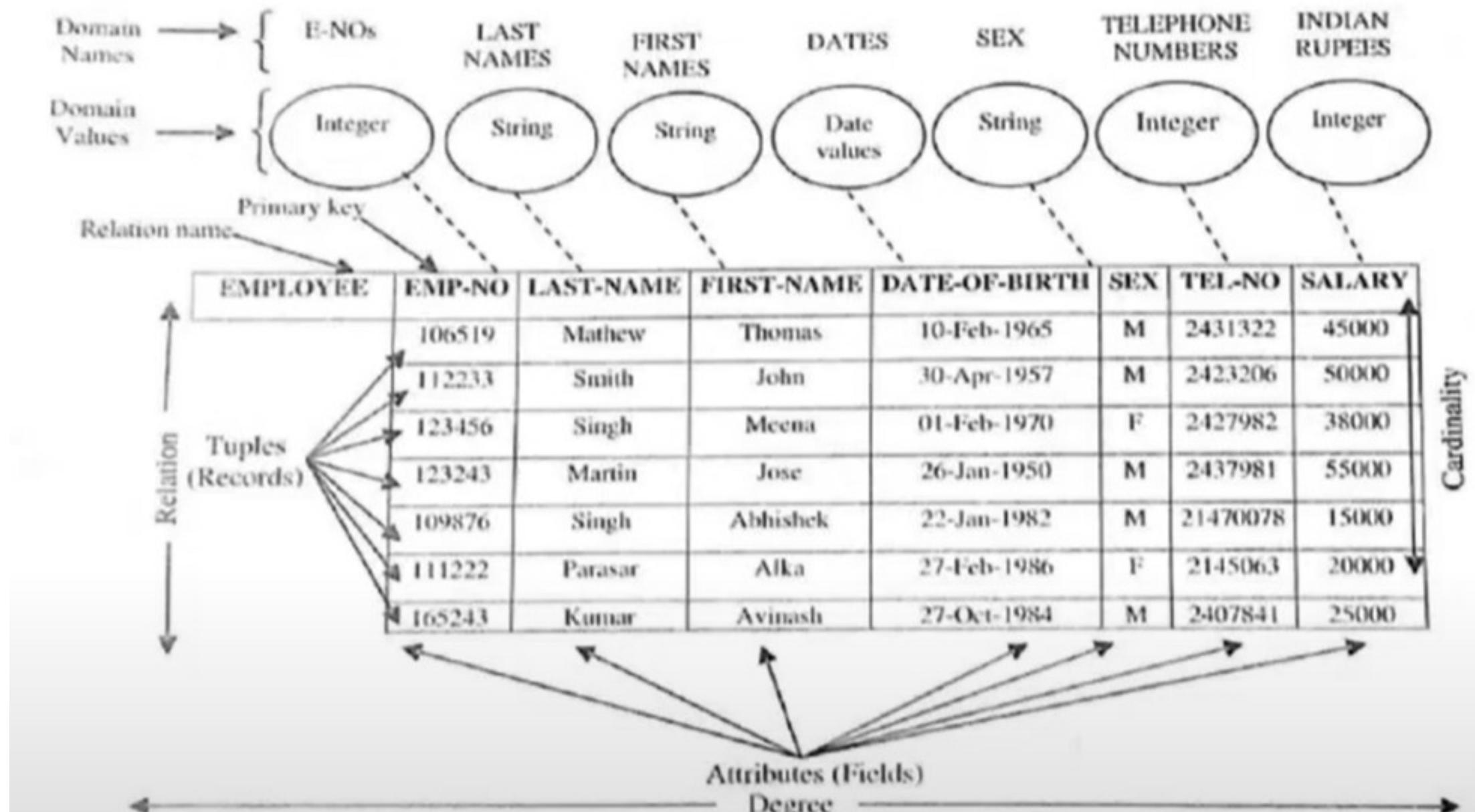
**Tuples**

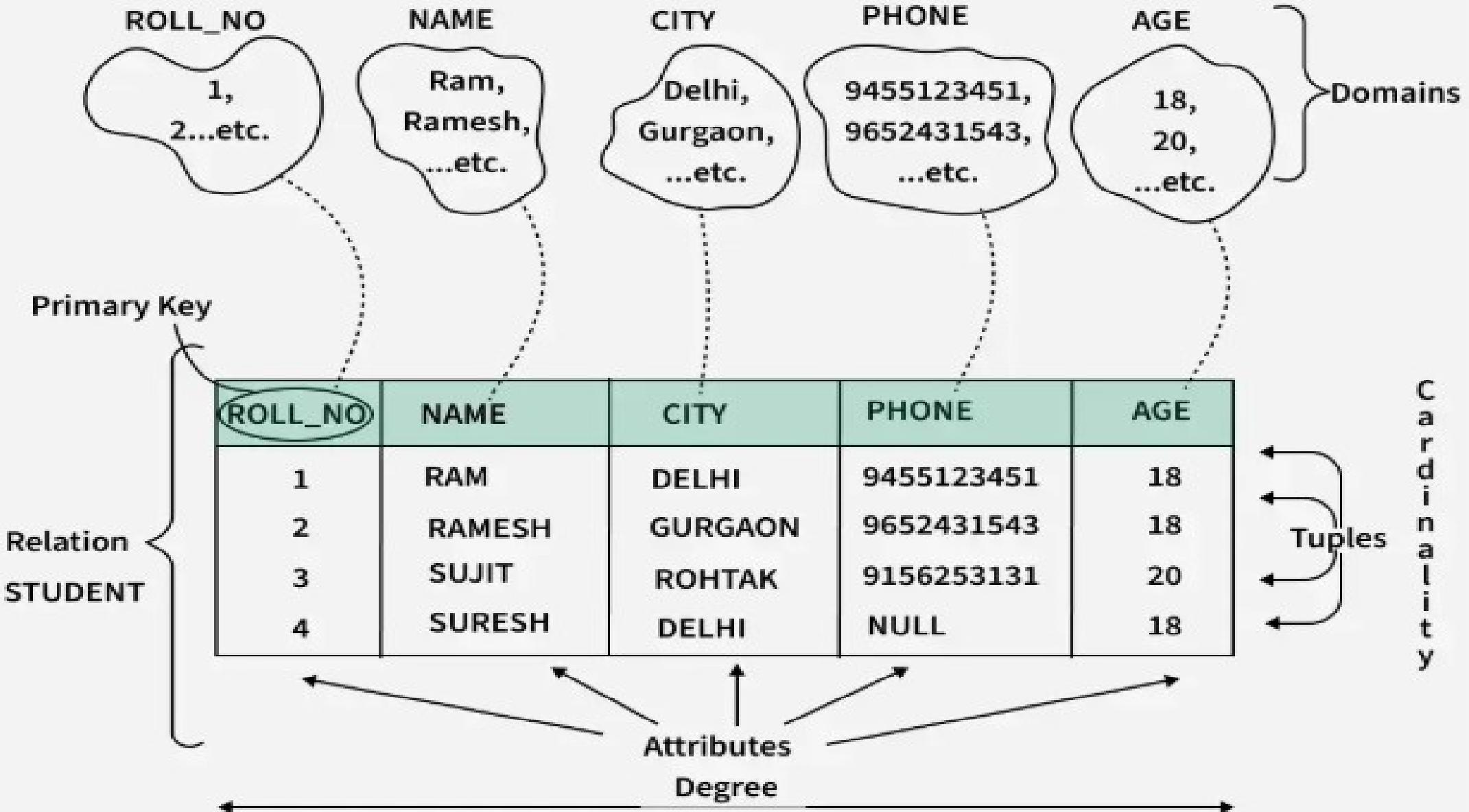
# **Components of relational database**

- The main components of relational database structure are as follows:
  - 1. Domains**
  - 2. Tuples (rows)**
  - 3. Columns**
  - 4. Keys**
  - 5. Relations (Tables)**

# Domain

- A domain is a **unique set of values permitted for an attribute in a table**
- It has three parts
  - Name
  - Data type
  - Values
- Eg : a domain of **month-of-year** have  
**Domain Name** : Month  
**Data type** : Date  
**Values** : Jan, Feb, Mar.....





# Tuples (rows)

- A tuple is an **ordered set of values**
- Tuple is a portion of a table containing **data that described only entity,**
- Also known as **record**
- Each value is derived from an **appropriate domain.**

DATA MODELING  
RELATIONAL DB

**Scheme Name** → **Customer**

**Attribute Names** → **CFirstName**   **CLastName**   **CStreet**   **CZipCode**   **CPhone**

**Row (tuple)** → **Kumar**   **Singh**   **52/57 store**   **223001**   **9889898989**

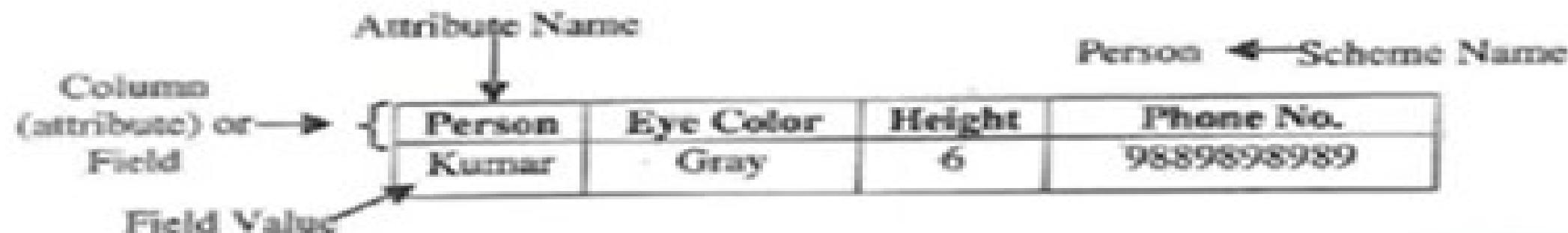
**Data Cell** → (Domain value assigned to attribute Name)

CFirstName	CLastName	CStreet	CZipCode	CPhone
Kumar	Singh	52/57 store	223001	9889898989

- **<Kumar, Singh, 52/57 store, 223001, 9889898989>** is a tuple belonging to the CUSTOMER relation.

# Columns

- Columns in a table are also called **attributes or fields of the relation**.
- A single cell in a table called **field value, attribute value or data element**.
- For example, for the entity person, attributes could include eye colour and height.



# Key of a Relation

- Each row has a value of a data item (or set of items) that **uniquely identifies that row in the table**
- Called the *key*

Key

STUDENT					
STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNT	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajasthan	India	18
4	SURESH		Punjab	India	21

# Relations (Tables)

- A table of values
- A relation may be thought of as a **set of rows & columns.**
- That is a table is perceived as a two-dimensional structure composed of rows and columns.
- Each row represents a fact that corresponds to a real-world **entity or relationship.**

# Terms related to Relational Model

- Schema of a Relation
- Degree of a Relation
- Relation state

# Schema of a Relation

- It is basically an outline of **how data is organized**
- It is denoted by  $R(A_1, A_2, \dots, A_n)$ 
  - R is relation name and it has some attributes  $A_1$  to  $A_n$
- Each attribute have some domain and it is represented by **dom( $A_i$ )**
- For example, the domain of Cust-id is 6 digit numbers

# Degree of a relation

- Degree of a relation is **number of attributes in a relation**
- Eg STUDENT(Id, Name, Age, Departmentno) Has degree 4
- Using datatype of each the definition can be written as

STUDENT(Id:Integer, Name:String,Age:integer,Departmentno:integer)

# Relation State

- The relation state is a **subset of the Cartesian product of the domains of its attributes**
- each domain contains the set of all possible values the attribute can take.
- Example: attribute Cust-name is defined over the domain of character strings of maximum length 25
  - $\text{dom}(\text{Cust-name})$  is  $\text{varchar}(25)$

- A **relation state**  $r(R)$  is a mathematical relation of degree  $n$  on the domains  $\text{dom}(A_1), \text{dom}(A_2), \dots, \text{dom}(A_n)$  which is a subset of Cartesian product( $X$ ) of domains that define  $R$

$$r(R) = (\text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times$$

$$r(R) \subseteq (\text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n))$$

- Cartesian product specifies all possible combination of values from underlying domains

The diagram illustrates a relation named STUDENT. The relation has seven attributes: Name, SSN, HomePhone, Address, OfficePhone, Age, and GPA. There are five tuples in the relation:

STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

# Relational Integrity Constraints

# Relational Integrity Constraints

- Constraints are conditions that must hold on all valid relation states.
- There are three main types of constraints in the relational model:
  - **Key constraints**
  - **Entity integrity constraints**
  - **Referential integrity constraints**

# Key Constraints

- **Superkey :**
- It is the combination of one or more attribute which can be used to uniquely identify a tuple/row in a relation R
- Consider the following relation Student

Student Relation					
Rollno	Name	Class	Section	Age	Address
1	Akhil	10	A	16	EKM
2	Amal	10	A	16	TVM
3	Aji	10	A	16	EKM
1	Ali	10	B	16	TVM
2	Akhil	10	B	16	TVM

Student Relation					
Rollno	Name	Class	Section	Age	Address
1	Akhil	10	A	16	EKM
2	Amal	10	A	16	TVM
3	Aji	10	A	16	EKM
1	Ali	10	B	16	TVM
2	Akhil	10	B	16	TVM

- Super Key – {Rollno,Name}
- Super Key – {Rollno,Name,Section}
- Super Key – {Rollno,Name,Section, Address}
- Super Key – {Rollno,Name,Address}
- Super Key – {Rollno,Address}

## Candidate Key

- It is **minimal Super Key**
- It is a Super Key which cannot be reducible further
- A candidate key is a **subset of a super key set**
- A Candidate Key can be a Super Key but Super Key cannot be candidate Key

Eg Consider the following Super Key sets

- Super Key – {Rollno, Name}
- Super Key – {Rollno, Name, Section}
- Super Key – {Rollno, Name, Section, Address}
- Super Key – {Rollno, Name, Address}
- Super Key – {Rollno, Address}

Candidate Keys - {Rollno, Name}, {Rollno, Address}

Student Relation					
Rollno	Name	Class	Section	Age	Address
1	Akhil	10	A	16	EKM
2	Amal	10	A	16	TVM
3	Aji	10	A	16	EKM
1	Ali	10	B	16	TVM
2	Akhil	10	B	16	TVM

## Primary Key

- Data base designers while designing the db which choose any of the candidate key to uniquely identify a record in a relation which will be considered as Primary Key
- Consider we have 2 Candidate Key sets

Candidate Keys - {Rollno,Name}, {Rollno, Address}

Any one set can be considered as primary key

Primary Key - {Rollno,Name},

**Alternate key:** the key that is not chosen to be

# Entity Integrity

- The **entity integrity constraint states that no primary key value can be NULL.**
- This is because the primary key value is used to identify individual tuples in a relation.
- Having NULL values for the primary key implies that we cannot identify some tuples.

# Referential Integrity Constraint

- A constraint involving two relations
- Used to specify a relationship among tuples in two relations:
  - One relation is called **referencing relation**
  - Other relation is called **referenced relation**

## Referential Integrity Constraint

- A constraint involving **two relations**
- Used to specify a **relationship among tuples in two relations:**
  - One relation is called **referencing relation**
  - Other relation is called **referenced relation**



## EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

## DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

- Eg :attribute Dno of EMPLOYEE gives the department number for which each employee works; hence, its value in every EMPLOYEE tuple must match the Dnumber value of some tuple in the DEPARTMENT relation.

- Tuples in the referencing relation R1 have attributes FK (called **foreign key attributes**) that reference the primary key attributes PK of the referenced relation R2.
- i.e **Foreign key** is an attribute in a relation which refers to the primary key of another relation

- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------



# Example 2

Student

Rollno	Name	Age	Address	Course_id
1	Adil	21	TVM	CS201
2	Akhil	22	EKM	CS203
3	Anu	21	TRC	CS202
4	Ben	21	EKM	CS201

Course

Course_id	Course Name	Credit
CS201	OS	4
CS202	DB	4
CS203	COA	4

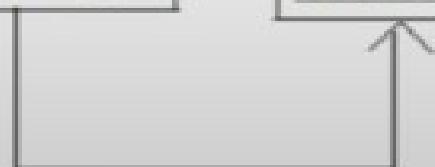
## Relational Schema

Student

Rollno	Name	Age	Address	Course_id

Course

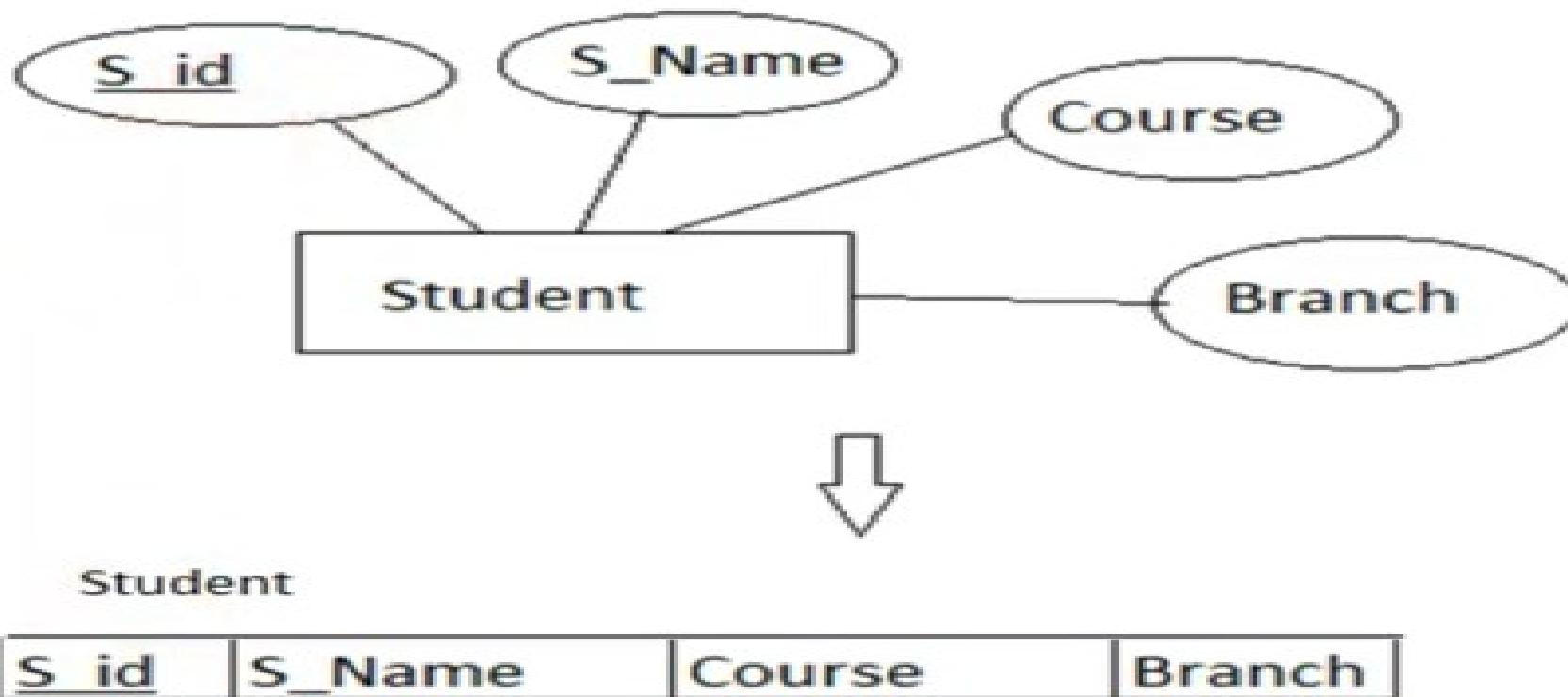
Course_id	Course Name	Credit



# **Convert ER diagram to Relational Schema**

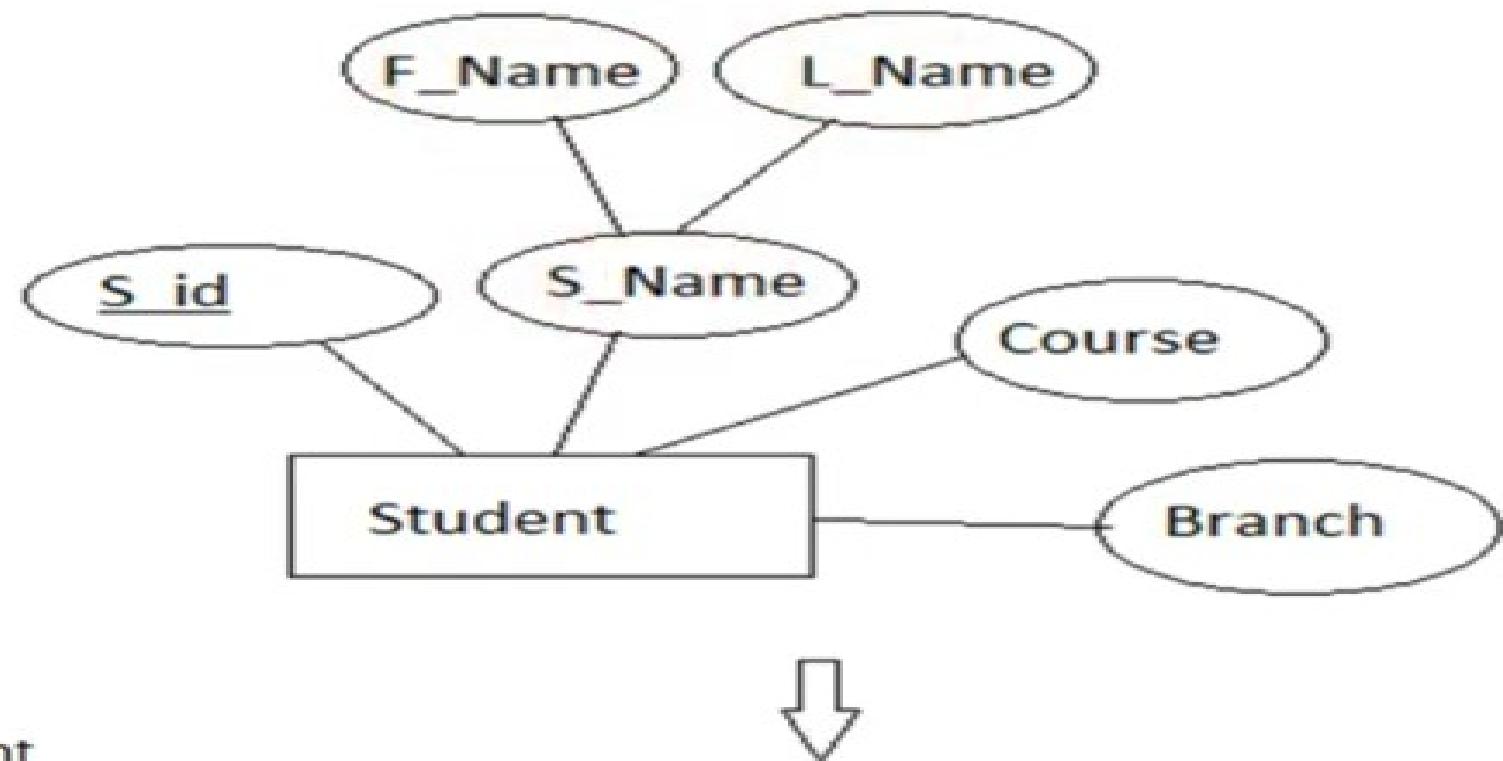
# Converting ER diagram to relational Schema

## 1) Converting Strong Entity set into Relational Schema



## 2) Composite Attributes to Relational schema

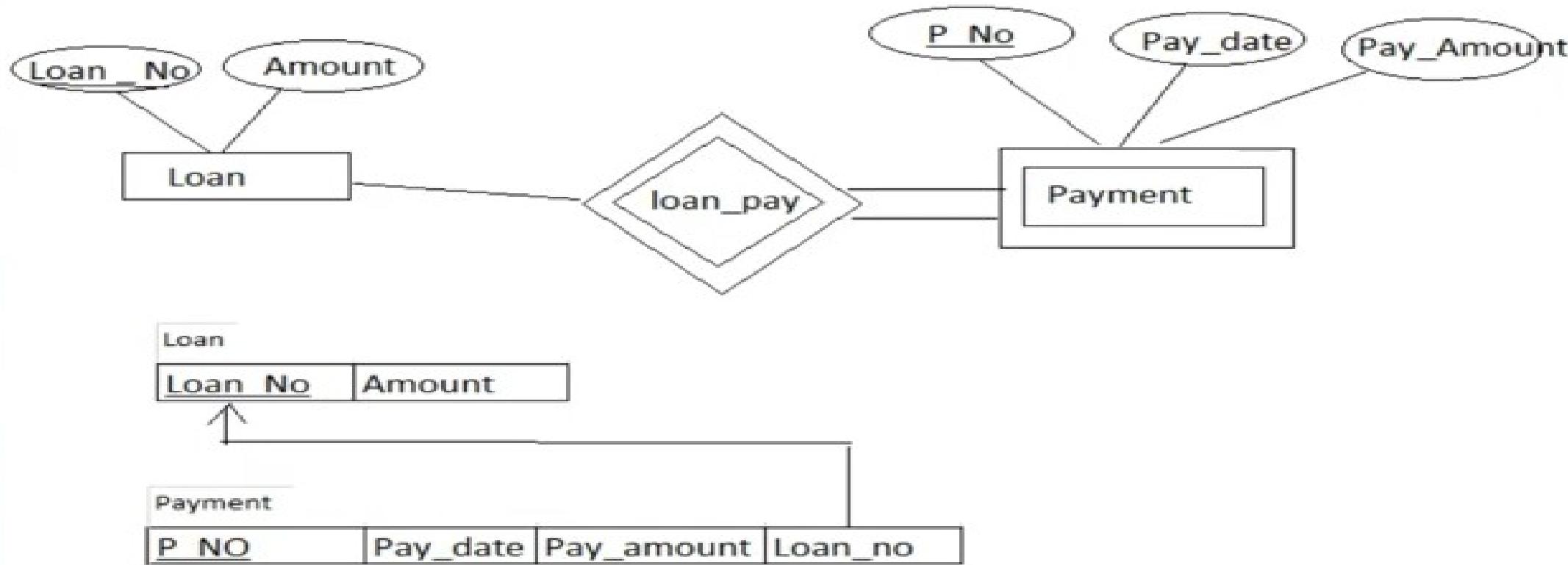
Make F\_Name,L\_Name as individual attribute



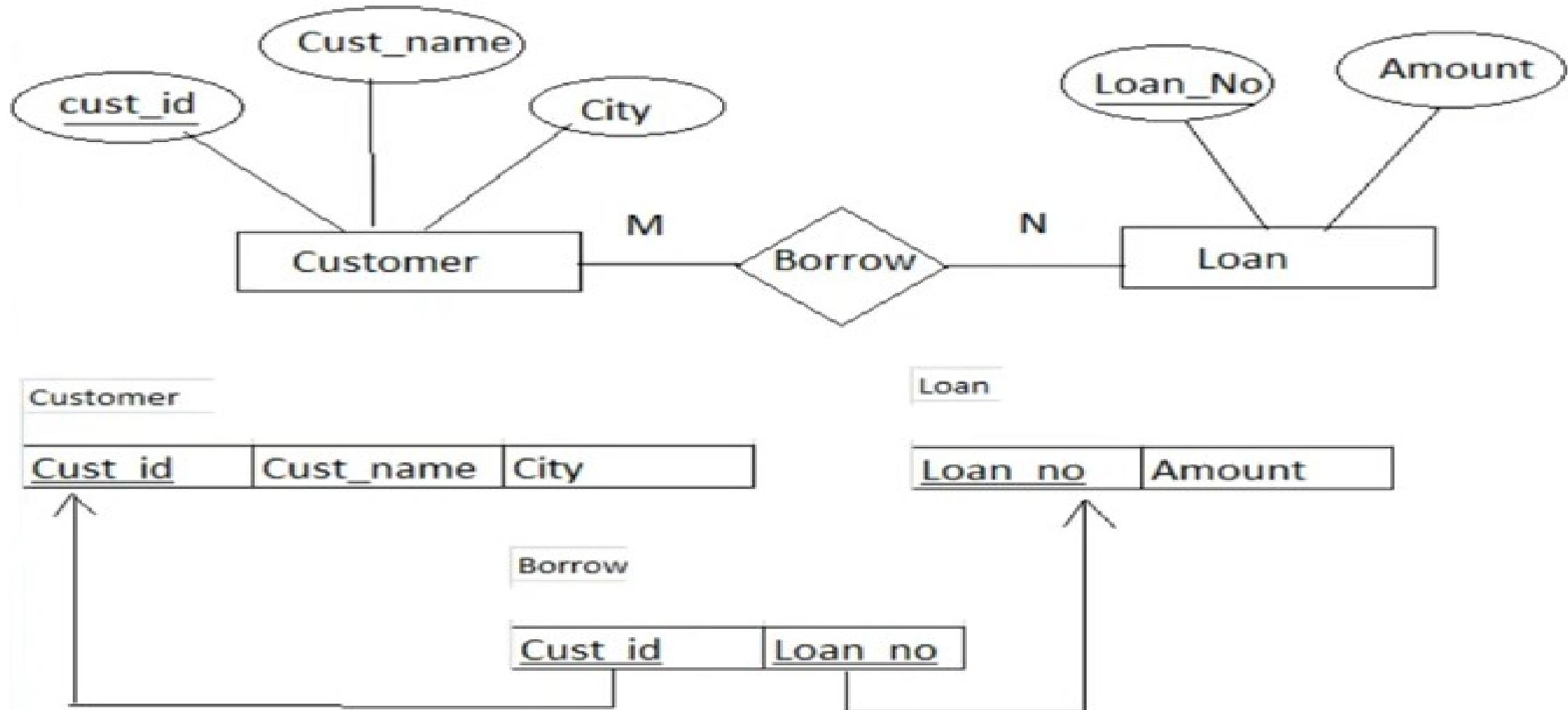
<u>S_id</u>	<u>F_Name</u>	<u>L_Name</u>	Course	Branch
-------------	---------------	---------------	--------	--------

### 3) Representing weak entity set in relational schema

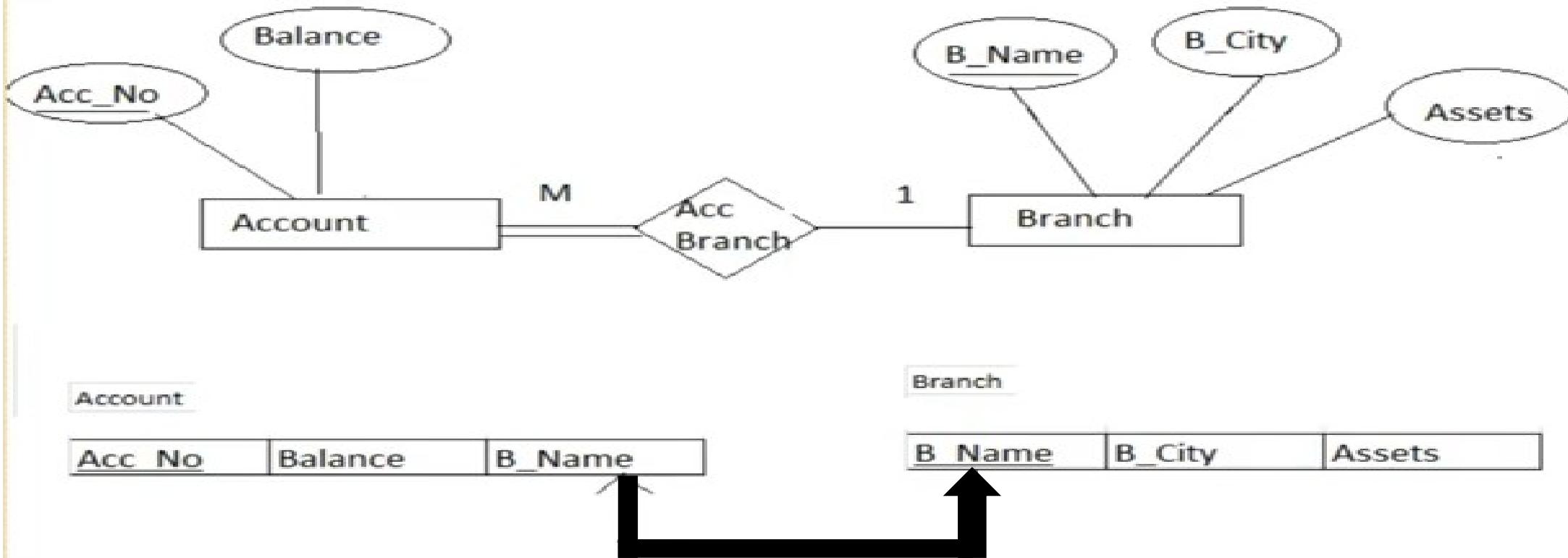
- Include the primary key of strong entity set in the weak entity



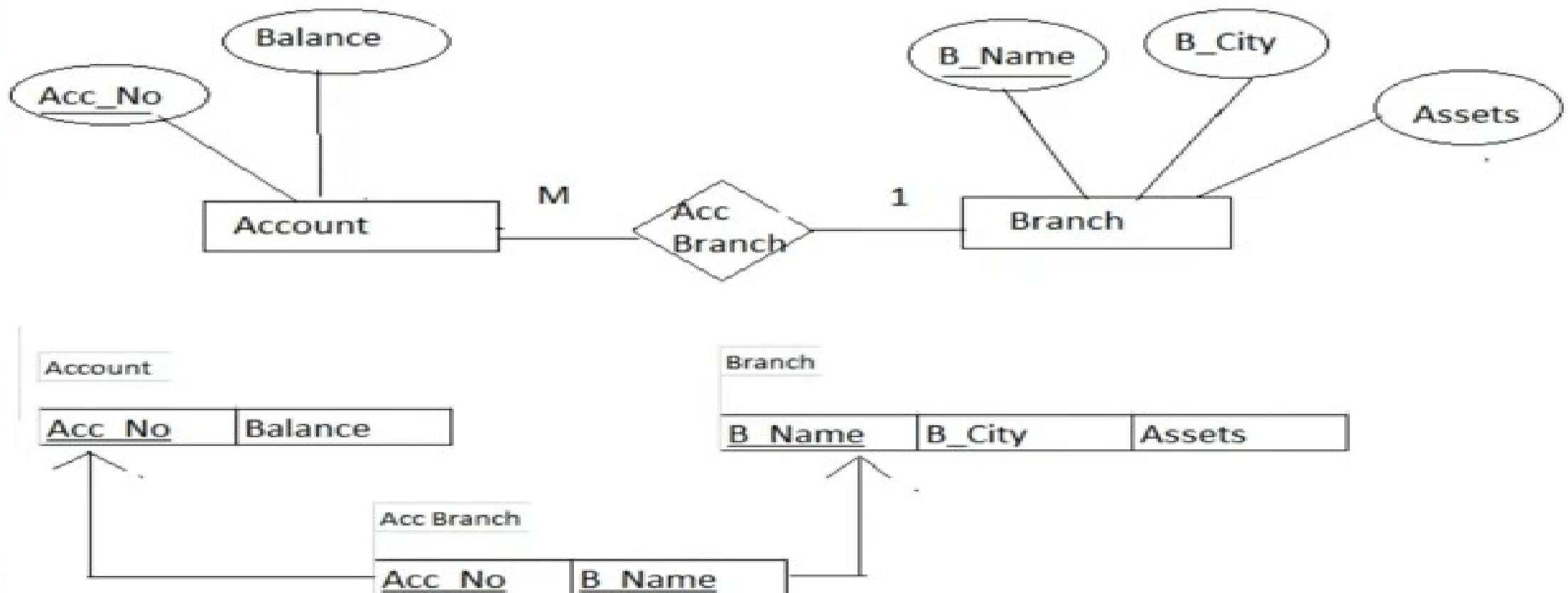
#### 4) Representing Many- Many relationship in relational schema



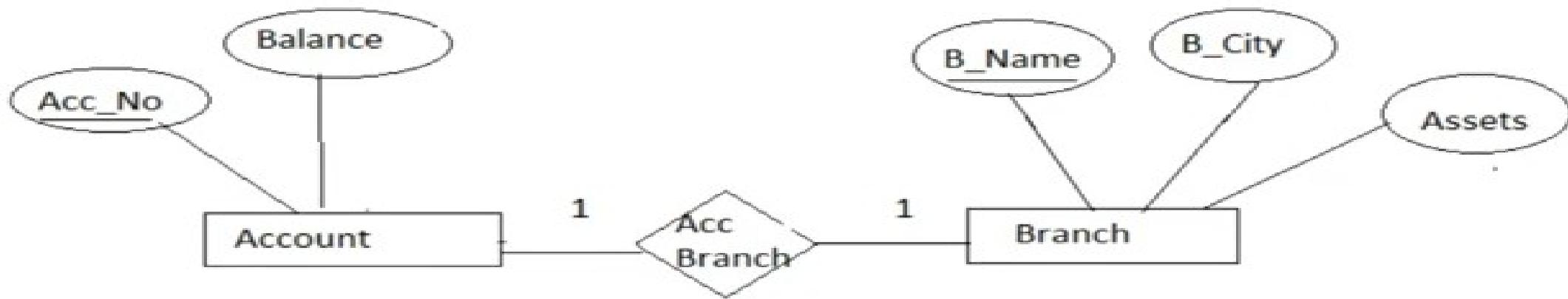
## 5) Representing Many- One or One-Many relationship with **total participation** in relational schema



## 6) Representing Many- One or One-Many relationship in relational schema



## 7) Representing One- One relationship in relational schema



Account

Acc No	Balance
--------	---------

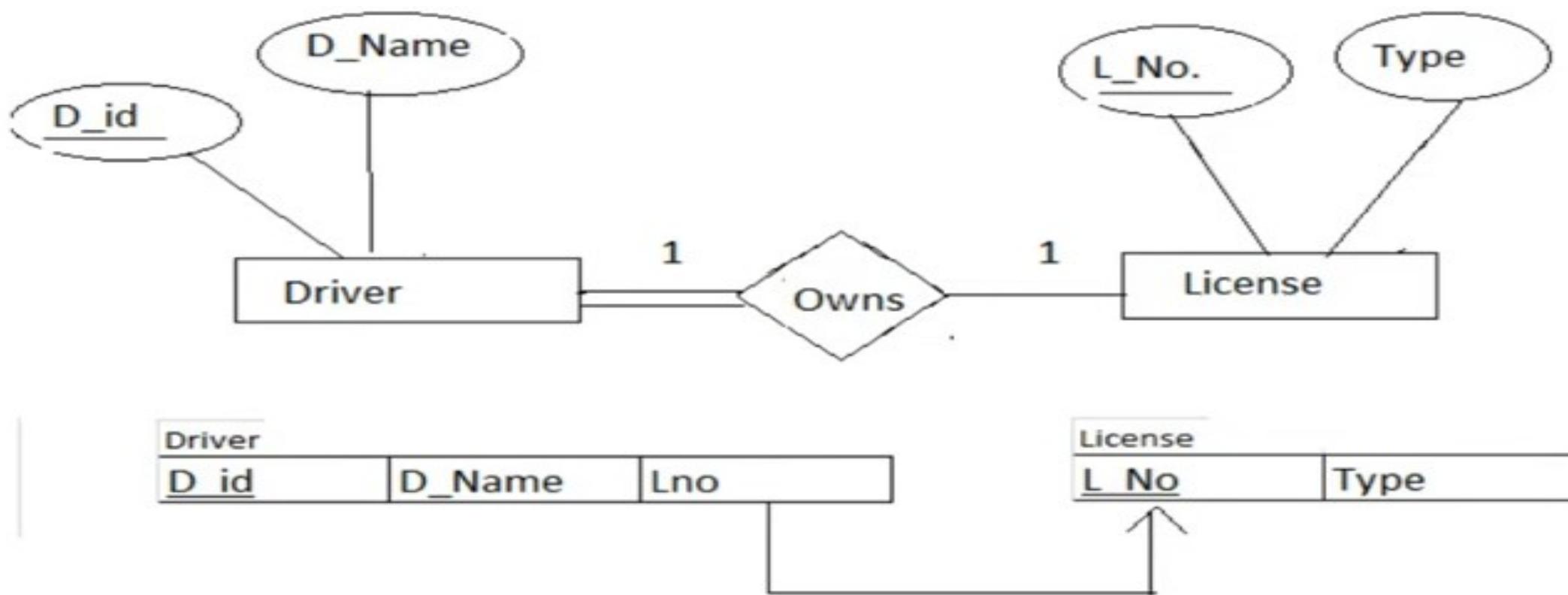
Branch

B Name	B City	Assets
--------	--------	--------

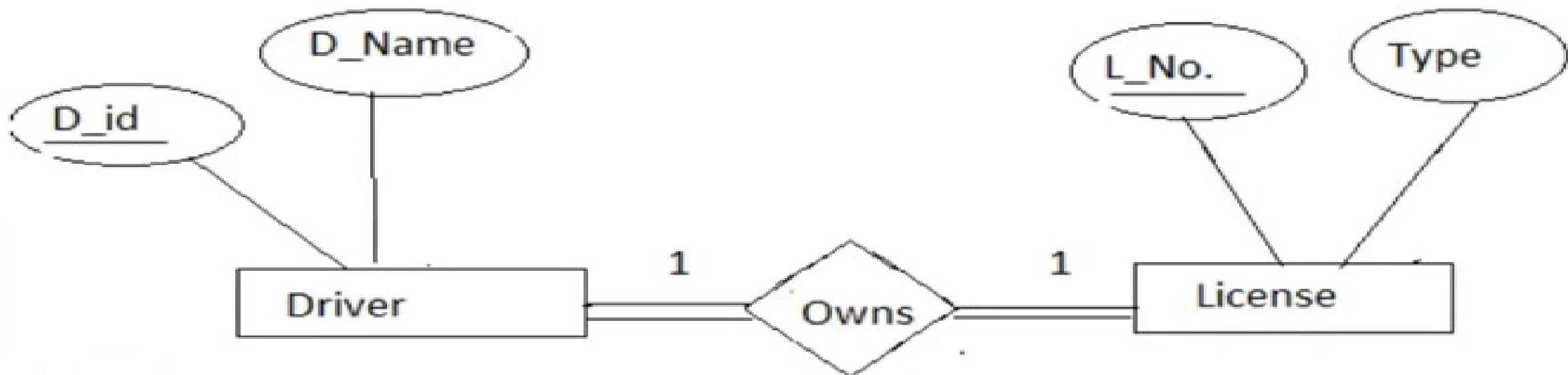
Acc Branch

Acc No	B Name
--------	--------

8) Representing One- One relationship  
with **total participation** in relational  
schema

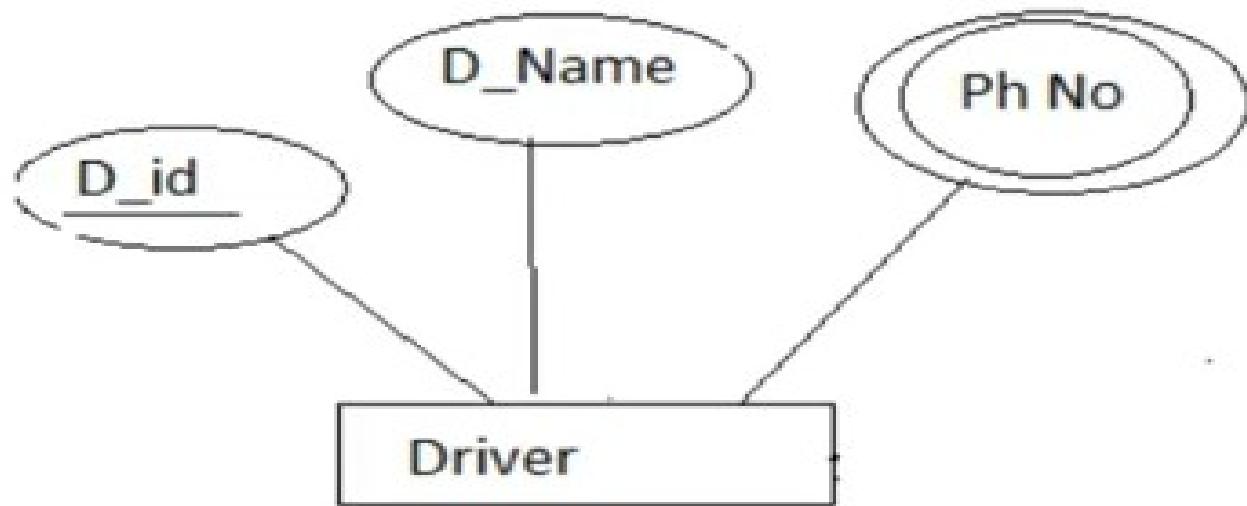


9) Representing One- One relationship  
with **total participation at both ends** in relational schema



<u>D_id</u>	<u>D_Name</u>	<u>L_No</u>	<u>Type</u>
-------------	---------------	-------------	-------------

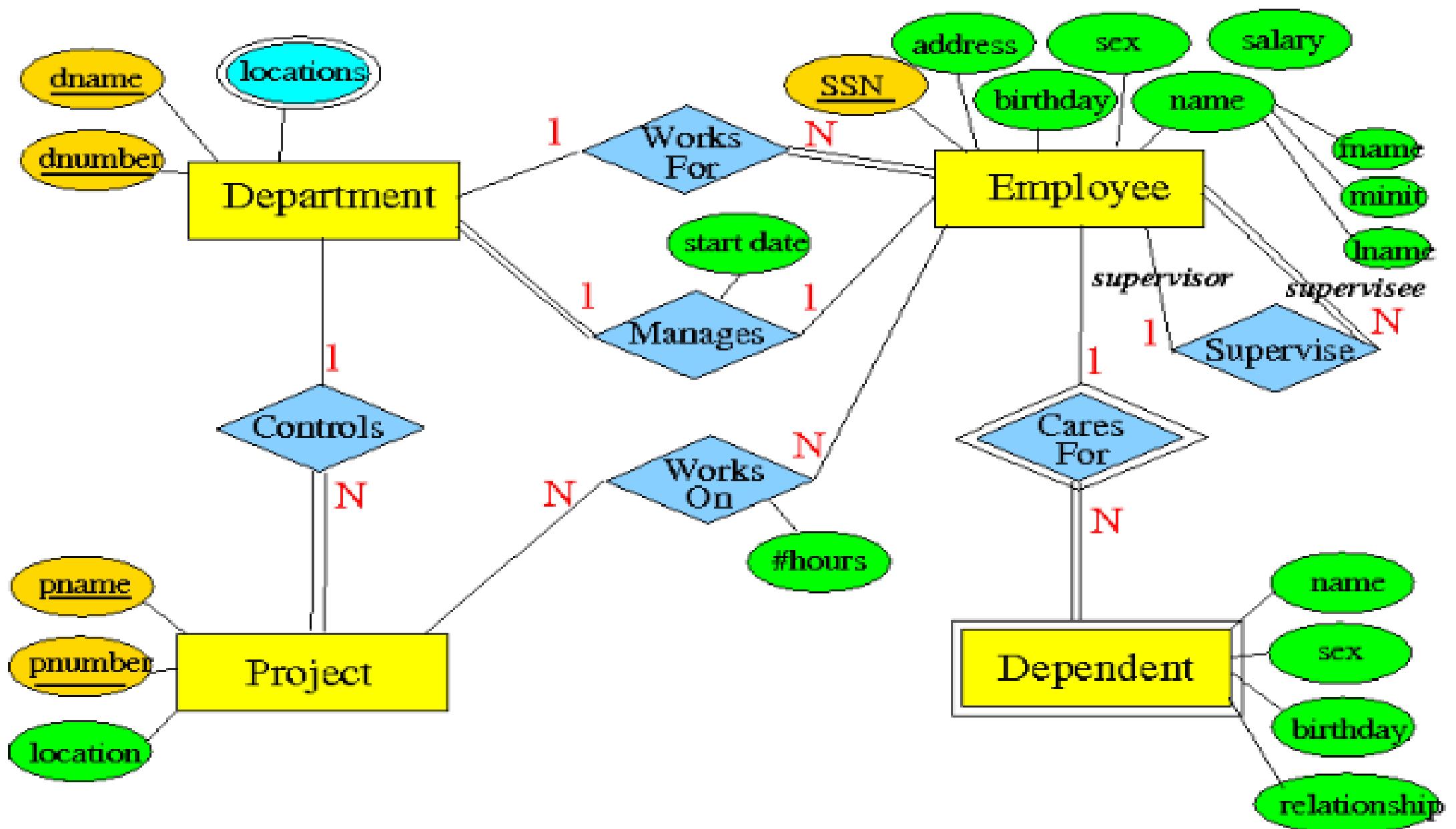
# 10) Representing Multivalued attributes in relational schema



Driver	
D_id	D_Name

Driver_Photo	
D_id	Ph No



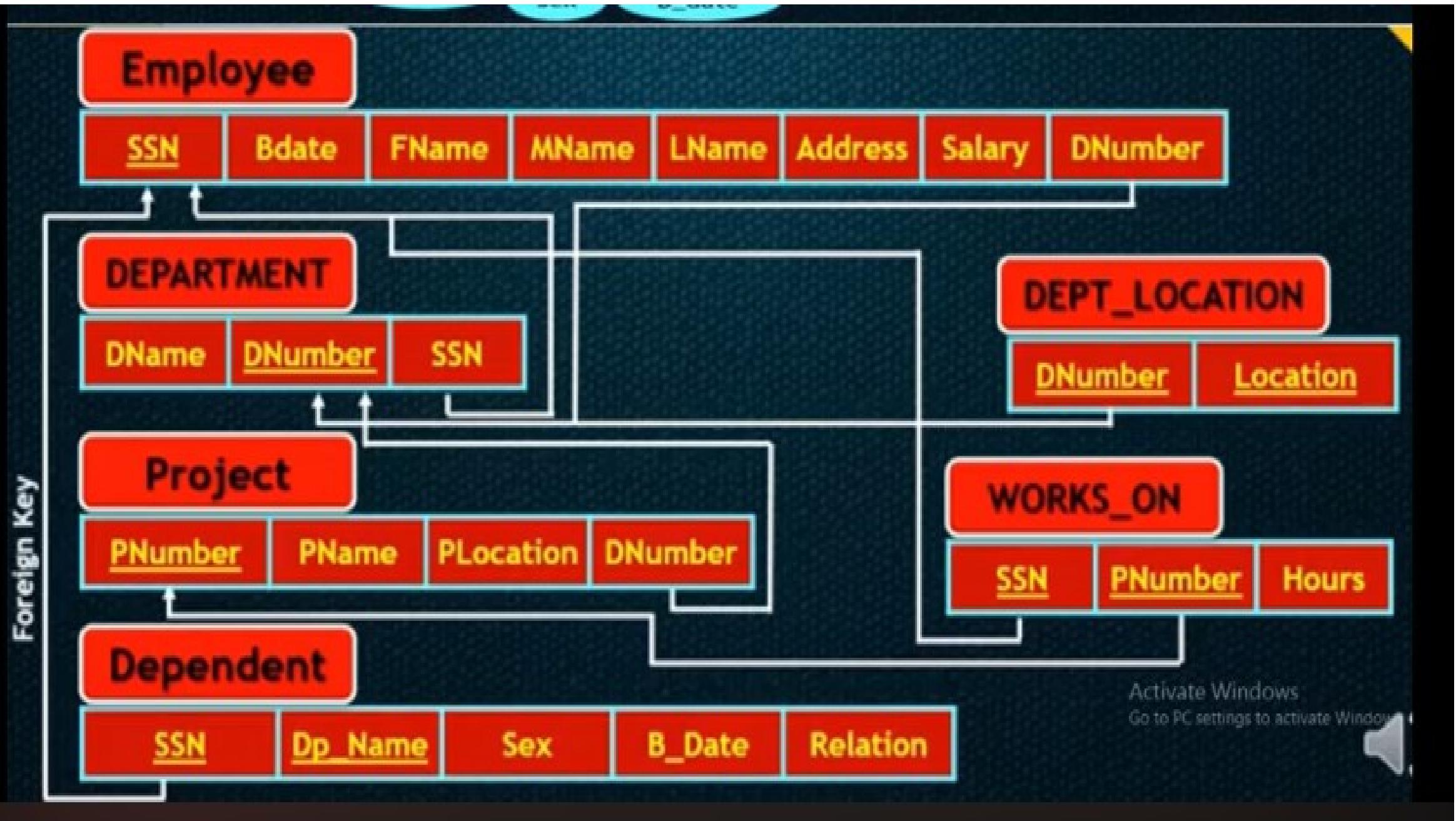


key attribute

non-key attribute

locations

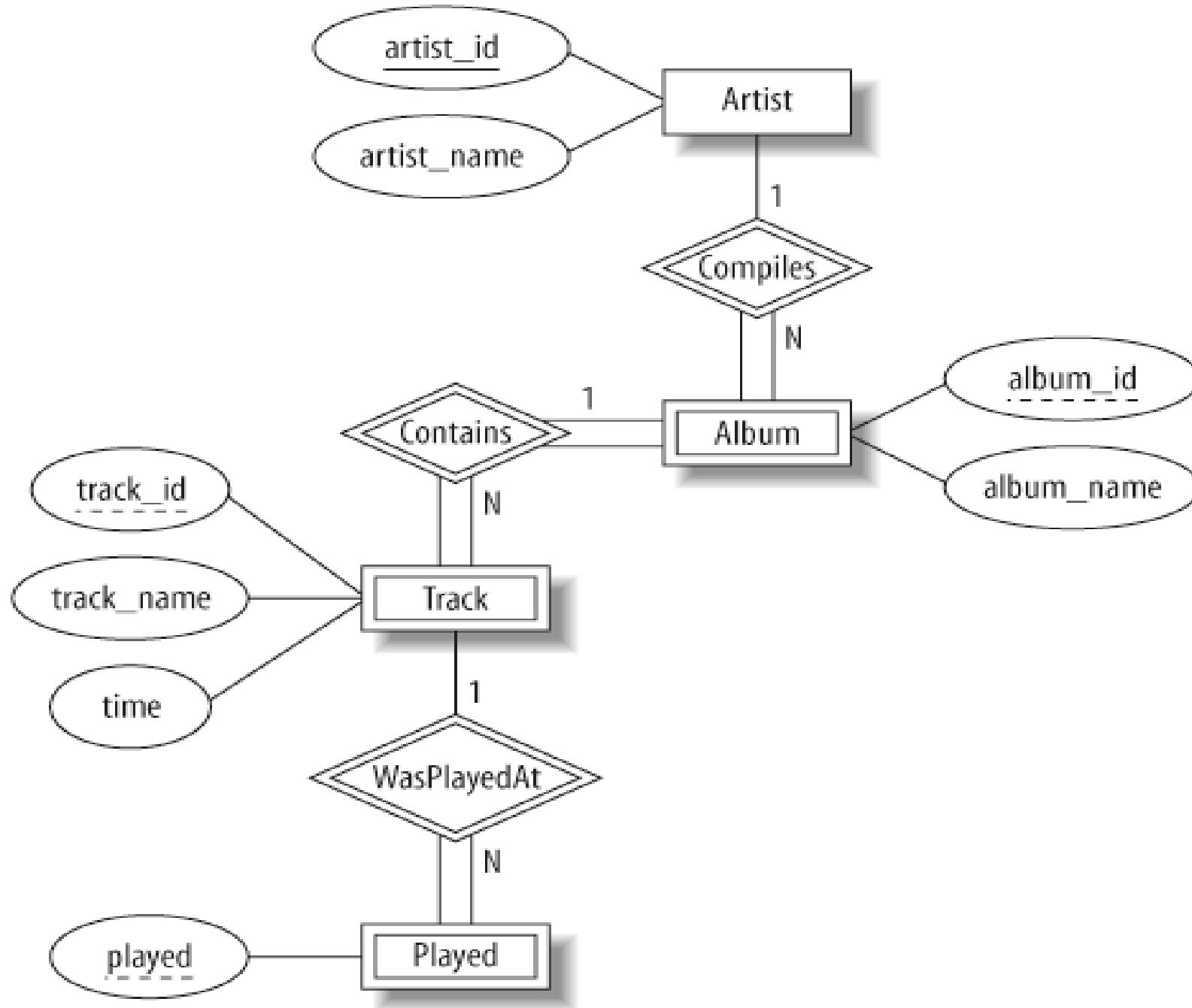
multi-valued attribute



Activate Windows

Go to PC settings to activate Windows





Please note that every weak entity set becomes strong by borrowing the key of its owner entity set. Thus the conversion can start from the strong set Artist.

Artist(artist-id, artist\_name)

Album(album\_id, artist\_id, album\_name)

Track (track\_id, album\_id, artist\_id, track\_name, time)

Played(played, track\_id, album\_id, artist\_id)

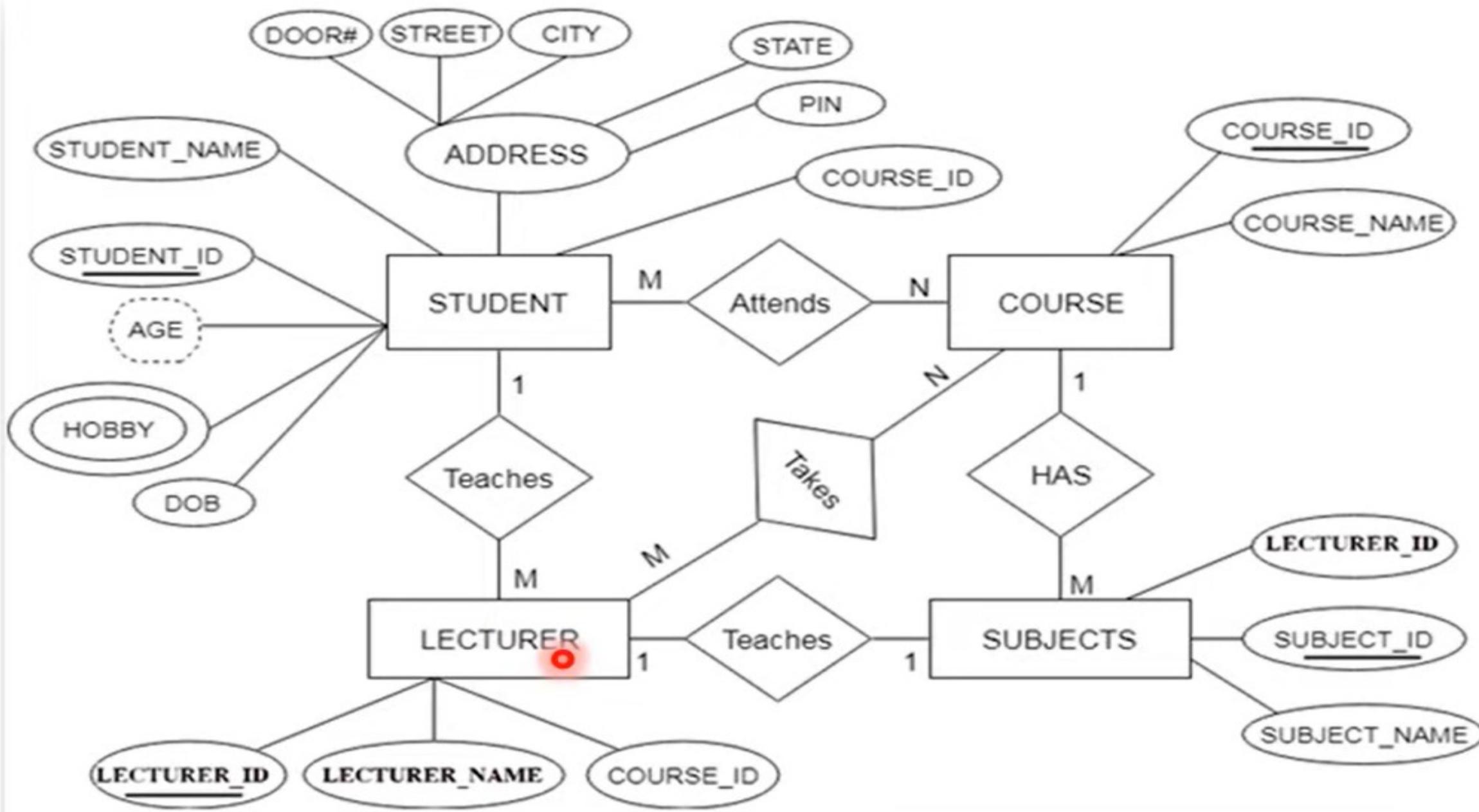
Primary keys are underlined.

Foreign keys are:

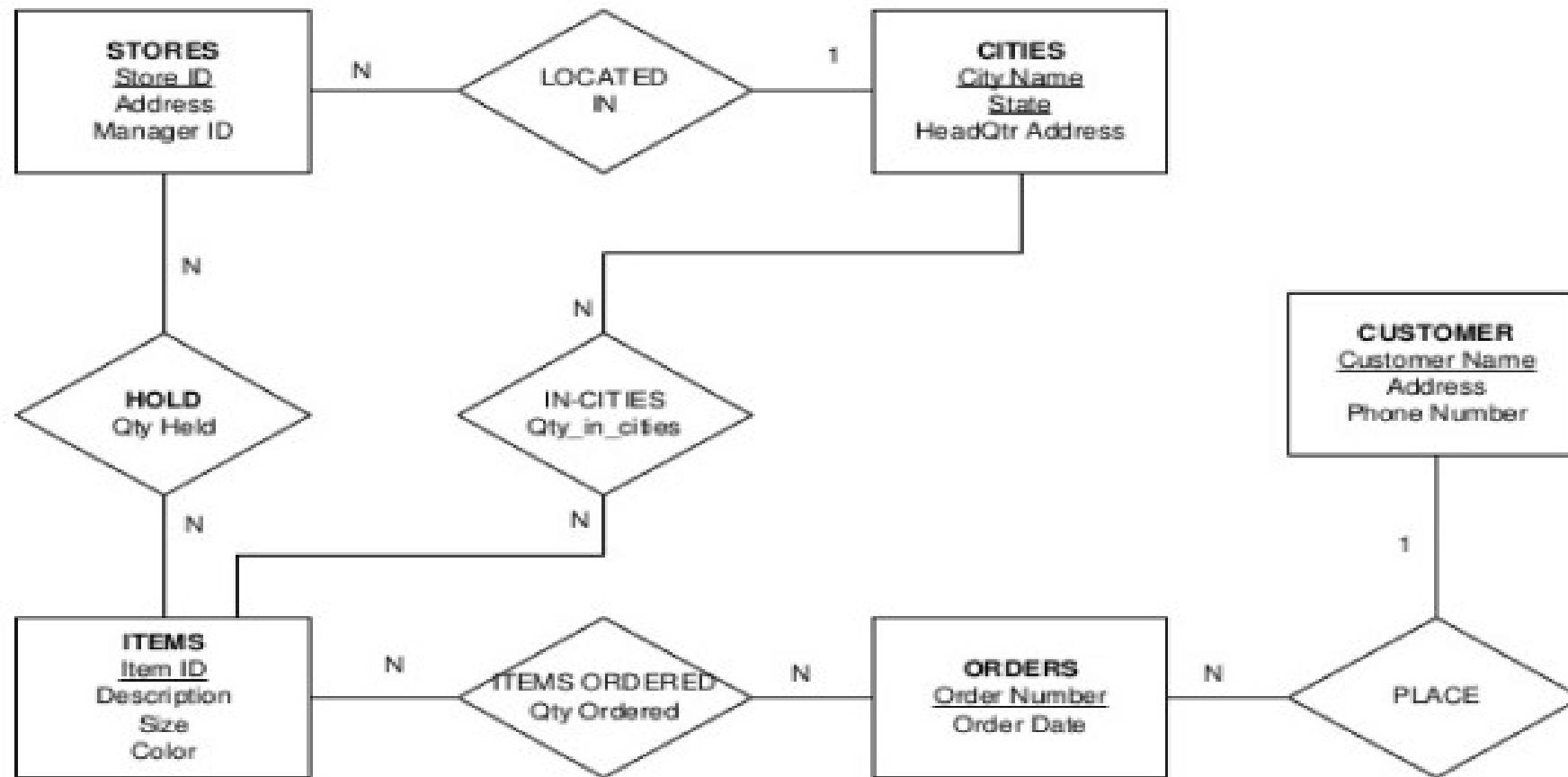
Album : artist\_id

Track : album\_id, artist\_id

Played: track\_id, album\_id, artist\_id



- a. Justify the importance of weak entity sets with the help of an example. (3)
- b. In the ER diagram below, names of entity sets and relationships are shown in capital and corresponding attributes are listed under each such name. Key attributes are underlined. All the *participations* are *total*. Use the standard synthesis procedure to convert the ER diagram into the corresponding relational schema. Clearly show primary and foreign keys. (6)



Stores

Store Id	Address	Manager_id	City Name	state
----------	---------	------------	-----------	-------

Hold

Store Id	Item id	Qty Held
----------	---------	----------

Cities

City name	state	HeadQtr Add
-----------	-------	-------------

In\_Cities

City name	state	Item id	Qty_in_Cities
-----------	-------	---------	---------------

Item

Item id	Descriptions	size	color
---------	--------------	------	-------

items ordered

Item id	order no	Qty Ordered
---------	----------	-------------

Orders

order no	order date	Cust_name
----------	------------	-----------

Customer

Cust name	Address	Phone No
-----------	---------	----------

# RELATIONAL ALGEBRA

# RELATIONAL ALGEBRA

- Relational Algebra is **procedural query language**, which takes Relation as input and generate relation as output
- The basic set of operations for the relational model is the **relational algebra**
- These operations enable a user to specify basic retrieval requests as *relational algebra expressions*
- Relational algebra is composed of various operations.

# The SELECT Operation

- The SELECT operation is used to choose a tuples (rows) from a relation that satisfies a selection condition.
- In general, the SELECT operation is denoted by

$$\sigma_{\text{<selection condition>}}(R)$$

- where the symbol **σ (sigma)** is used to denote the SELECT operator and the **selection condition** is a Boolean expression (condition) specified on the attributes of **relation R**

**EMPLOYEE**

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

 $\sigma_{Dno=4}(\text{EMPLOYEE})$  $\sigma_{\text{Salary} > 30000}(\text{EMPLOYEE})$

# The SELECT Operation

- For example, to select the tuples for all employees who either work in department 4 and make over \$25,000 per year, or work in department 5 and make over \$30,000, we can specify the following SELECT operation:

$$\sigma_{(Dno=4 \text{ AND } \text{Salary}>25000) \text{ OR } (Dno=5 \text{ AND } \text{Salary}>30000)}(\text{EMPLOYEE})$$

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

# The PROJECT Operation

- The PROJECT operation selects certain columns from the table and discards the other columns
- The general form of the PROJECT operation is

$$\pi_{\langle \text{attribute list} \rangle}(R)$$

(pi) is the symbol used to represent the PROJECT operation,  
 $\langle \text{attribute list} \rangle$  is the list of attributes from the attributes of relation  $R$

**EMPLOYEE**

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

- For example, to list each employee's first and last name and salary, we can use the PROJECT operation as follows:

$$\pi_{\text{Lname}, \text{Fname}, \text{Salary}}(\text{EMPLOYEE})$$

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000

- For example, to list each employee's first and last name and salary whose salary is greater than 30000

For example, to retrieve the first name, last name, and salary of all employees who work in department number 5

- For example, to list each employee's first and last name and salary whose salary is greater than 30000

$\Pi \text{Firstname, Lastname}(\sigma_{\text{salary} > 3000}(\text{Employee}))$

For example, to retrieve the first name, last name, and salary of all employees who work in department number 5

$\pi_{\text{Fname, Lname, Salary}}(\sigma_{\text{Dno}=5}(\text{EMPLOYEE}))$

# The UNION, INTERSECTION and MINUS Operations

- Two relations  $R(A_1, A_2, \dots, A_n)$  and  $S(B_1, B_2, \dots, B_n)$  are said to be **union compatible** (or **type compatible**) if two relations have the **same number of attributes and each corresponding pair of attributes has the same domain**.
- **UNION**: The result of this operation, denoted by  $R \cup S$ , is a relation that includes all tuples that are either in  $R$  or in  $S$  or in both  $R$  and  $S$ . Duplicate tuples is eliminated.
- **INTERSECTION**: The result of this operation, denoted by  $R \cap S$ , is a relation that includes all tuples that are in both  $R$  and  $S$ .
- **SET DIFFERENCE** (or **MINUS**): The result of this operation, denoted by  $R - S$ , is a relation that includes all tuples that are in  $R$  but not in  $S$ .

# STUDENT ⋃ INSTRUCTOR.

(a) STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(b)

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

## STUDENT $\cap$ INSTRUCTOR.

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

## STUDENT – INSTRUCTOR.

(d)

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

Fname	Lname
John	Smith
Ricardo	Brown
Francis	Johnson

## INSTRUCTOR – STUDENT.

# CARTESIAN PRODUCT (CROSS PRODUCT)

## Operation

- The CARTESIAN PRODUCT operation—also known as CROSS PRODUCT or CROSS JOIN—which is denoted by  $\times$  .
- For  $R \times S$  , the cartesian product operation defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S

## Student

Roll_no	Name
1	Beena
2	Ramu
3	Shilpa

## Instructor

Ins_no	Ins_name	subject
1	Shamna	Science
2	Anjali	Maths

## Student X Instructor

Roll_no	Name	Ins_no	Ins_name	Subject
1	Beena	1	Shamna	Science
1	Beena	2	Anjali	Maths
2	Ramu	1	Shamna	Science
2	Ramu	2	Anjali	Maths
3	Shilpa	1	Shamna	Science
3	Shilpa	2	Anjali	Maths

# The JOIN Operation

- The JOIN operation, denoted by  is used to combine *related tuples* from two relations into single “longer” tuples
- A Join combines two tuples from two relation only if the join condition is satisfied

# The JOIN Operation

- Equi join & Natural Join
- Left Outer join
- Right Outer Join

## Natural Join

- Natural join can only be performed if there is a common attribute between the relation.
  - The name & type of the attribute must be same

## Equi Join

- The most common use of JOIN involves join conditions with equality comparisons only. Such a JOIN, where the only comparison operator used is  $=$ , is called an EQUIJOIN

Example

P  $\bowtie$  Q

Customer			Order	
Cid	Cname	Age	Oid	Oname
101	Ajay	20	101	Pizza
102	Vijay	19	101	Noodles
103	Sita	21	103	Burger

Customer		$\bowtie_{Customer.cid=Order.oid}$	Order
Cid	Cname	Age	Oname
101	Ajay	20	Pizza
101	Ajay	20	Noodles
103	Sita	21	Burger



- Example : retrieve the name of the manager of each department

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-09	639 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

- Example : retrieve the name of the manager of each department

**EMPLOYEE**

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-09	639 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

**DEPARTMENT**

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

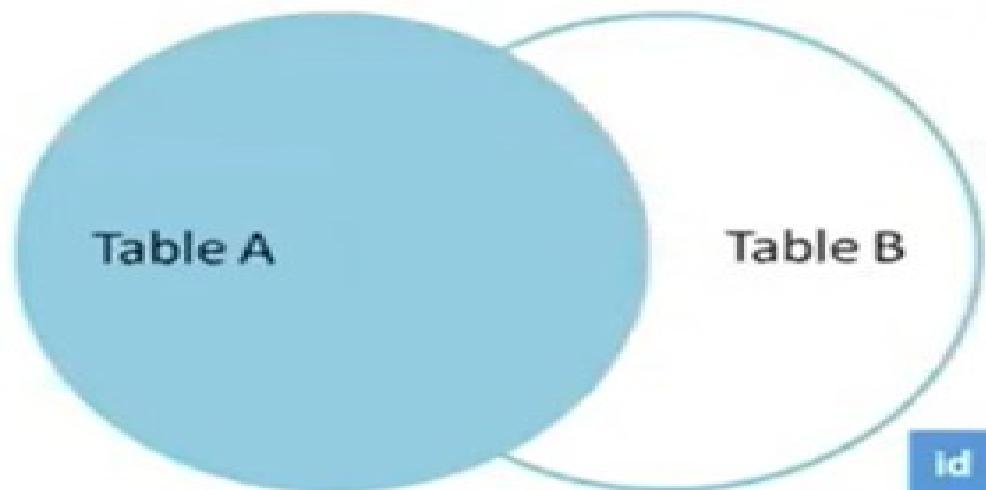
$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_ssn}=\text{Ssn}} \text{EMPLOYEE}$   
 $\text{RESULT} \leftarrow \pi_{\text{Dname}, \text{Lname}, \text{Fname}}(\text{DEPT\_MGR})$

We can also combine the relational algebra expression as follows

$$\pi_{Dname, Lname, Fname} ( \text{DEPARTMENT} \bowtie_{\text{Mar ssn}=Ssn} \text{EMPLOYEE} )$$

# LEFT JOIN ( $\bowtie$ )

- This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join.
- The rows for which there is no matching row on right side, the result-set will contain *null*.
- LEFT JOIN is also known as LEFT OUTER JOIN.



A		B	
<b>Id</b>	<b>Name</b>	<b>Name</b>	<b>Marks</b>
10	Jay	Rohan	20
20	Veer	Veer	18
30	John	John	14
		Sam	13

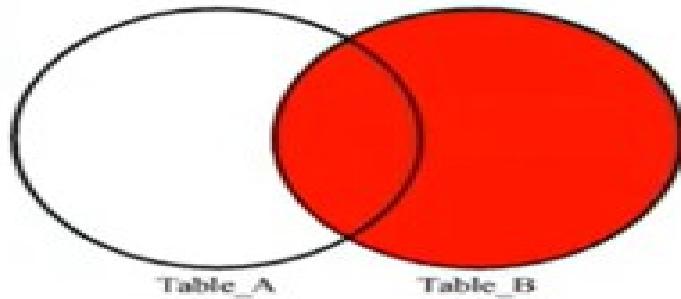
A  $\bowtie$  B

<b>Id</b>	<b>Name</b>	<b>Marks</b>
10	Jay	NULL
20	Veer	18
30	John	14



# RIGHT JOIN( $\bowtie$ )

- RIGHT JOIN is similar to LEFT JOIN.
- This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join.
- The rows for which there is no matching row on left side, the result-set will contain *null*.
- RIGHT JOIN is also known as RIGHT OUTER JOIN.



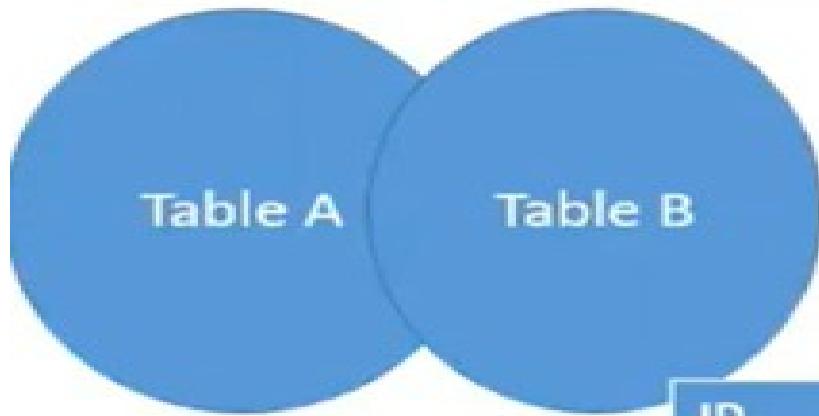
A  $\bowtie$  B

<b>id</b>	<b>Name</b>	<b>Marks</b>
Null	Rohan	20
20	Veer	18
30	John	14
Null	Sam	13

A		B	
<b>Id</b>	<b>Name</b>	<b>Name</b>	<b>Marks</b>
10	Jay	Rohan	20
20	Veer	Veer	18
30	John	John	14
		Sam	13

# FULL JOIN ( $\bowtie$ )

- FULL OUTER JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN.
- The result-set will contain all the rows from both the tables.
- The rows for which there is no matching, the result-set will contain *NULL* values.



A  $\bowtie$  B

ID	Name	Marks
10	Jay	NULL
20	Veer	18
30	John	14
NULL	Rohan	20
Null	Sam	13

A		B	
Id	Name	Name	Marks
10	Jay	Rohan	20
20	Veer	Veer	18
30	John	John	14
		Sam	13

Q1) Books (AccNo, ISBN, Title, Edition, Year)

Members (Memberid, MemberName, MemberType)

IssuedTo (AccNo, Memberid, DateOfIssue)

Write relational algebra expression for the  
following queries

(a) Accession Number(s) and Name(s) of third edition  
books published in 2018

(b) Names and date of issues of books taken by  
member name 'PRIVA'

(c) Names of books not taken by any member.

Q2) Write Relational Algebra expression for the queries that follow

Student (Rollno, Name, Age, Gender, Address, Advisorid)

Course (Courseid, cname, credits)

Professor (Profid, pname, phone)

Enrollment (Rollno, courseid, grade)

Primary Keys are underlined, Advisor is a FK

referring to the professor table Rollno and courseid in Enrollment are also FK referring to the primary key with same name.

- Names of the female student
- Names of the male students along with advisor name
- Rollno and name of the student who have not enrolled for any course

(a) suppliers (sid, sname, address)

parts (pid, pname, color)

Catalog (sid, pid, cost)

Write relational algebra for the following.

(a) Find the names of supplies who supply some red parts

(b) find the sids of supplier who supply some red or green part

(c) find the sids of supplier who supply some red and some green part

Albums (Album #, Album Name, Produced by, Years)

Songs (Song #, Song Start, duration, Album #)

Sung by (Artist Name, Song #)

Write Relational algebra expressions for the following.

(a) Name of albums produced by 'HMV' in the year 2018

(b) Names of the albums in which an artist with name 'AVANTHIKA' sung

(c) Names of albums in which all artists have sung song.

# Introduction to Structured Query Language



- ▶ Structured Query Language(SQL) was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s.
- ▶ This was initially called SEQUEL(Structured English QUery Language).
- ▶ The main objective of SQL is to update, store, manipulate and retrieve data stored in a relational database.



Donald D. Chamberlin



Raymond F. Boyce



# Data Definition Language

---

- ▶ The SQL data-definition language (DDL) allows the specification of information about relations, including:
  - ▶ The schema for each relation.
  - ▶ The domain of values associated with each attribute.
  - ▶ Integrity constraints



# Domain Types in SQL

---

- ▶ **char(n)**. Fixed length character string, with user-specified length n.
  - ▶ **varchar(n)**. Variable length character strings, with user-specified maximum length n.
  - ▶ **int**. Integer (a finite subset of the integers that is machine-dependent).
  - ▶ **smallint**. Small integer (a machine-dependent subset of the integer domain type).
  - ▶ **real, double precision**. Floating point and double-precision floating point numbers, with machine-dependent precision.
  - ▶ **float(n)**. Floating point number, with user-specified precision of at least n digits.
-



# DDL Commands

---

- ▶ DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema.
- ▶ It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.
- ▶ Examples of DDL commands:
  - ▶ CREATE - is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
  - ▶ DROP - is used to delete objects from the database.
  - ▶ ALTER-is used to alter the structure of the database.



# CREATE TABLE

---

- ▶ An SQL relation is defined using the create table command:

```
create table r (A1 D1, A2 D2, ..., An Dn,  
               (integrity-constraint1),  
               ...,  
               (integrity-constraintk))
```

- ▶ r is the name of the relation
- ▶ each A<sub>i</sub> is an attribute name in the schema of relation r
- ▶ D<sub>i</sub> is the data type of values in the domain of attribute A<sub>i</sub>
- ▶ Example:

```
create table instructor (  
    ID      char(5),  
    name    varchar(20),  
    dept_name varchar(20),  
    salary   numeric(8,2))
```



# Integrity Constraints in Create Table

---

- ▶ Types of Integrity Constraints in DBMS
  - ▶ Entity Integrity Constraints
  - ▶ Referential Integrity Constraints
  - ▶ Domain Constraints



# Integrity Constraints in Create Table

---

- ▶ primary key ( $A_1, \dots, A_n$ )
  - ▶ The attributes required to be nonnull and unique
- ▶ foreign key ( $A_m, \dots, A_n$ ) references s
  - ▶ The values of the attributes ( $A_m, \dots, A_n$ ) for any tuple in the relation must correspond to values of the primary key attributes of some tuple in relation s.
- ▶ not null
  - ▶ Specifies that the null value is not allowed for that attribute



# Create Table Example

```
▶ create table instructor (
    ID          char(5),
    name        varchar(20) not null,
    dept_name   varchar(20),
    salary      numeric(8,2),
    primary key (ID),
    foreign key (dept_name) references department);
```

# Create Table Example

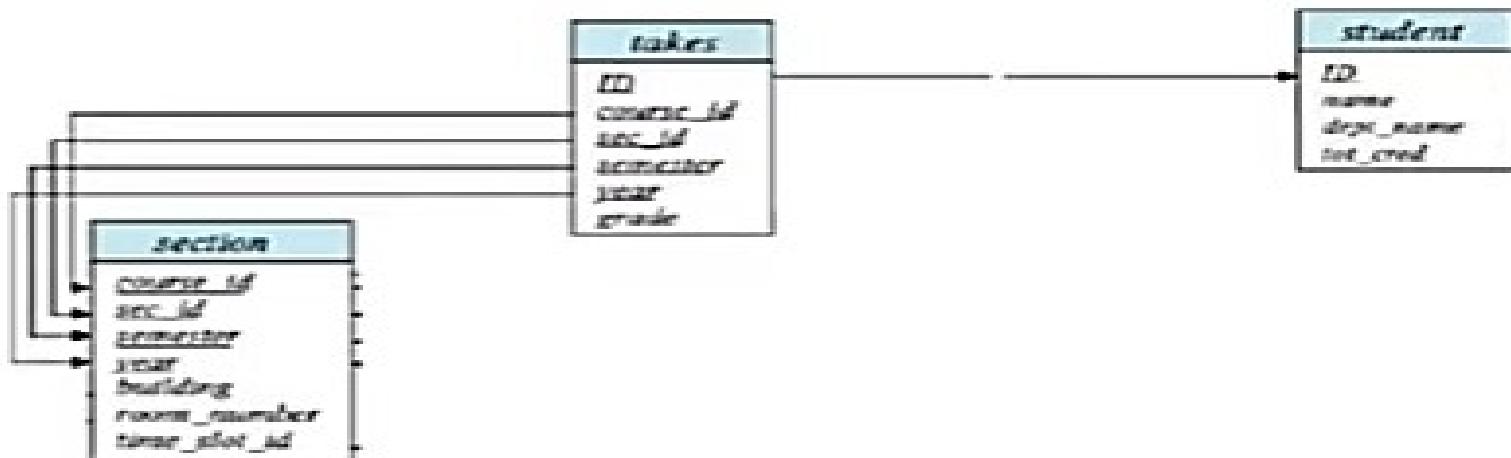
► **create table** takes (

ID	varchar(5),
course_id	varchar(8),
sec_id	varchar(8),
semester	varchar(6),
year	numeric(4,0),
grade	varchar(2),

primary key (ID, course\_id, sec\_id, semester, year) ,

foreign key (ID) references student,

foreign key (course\_id, sec\_id, semester, year) references section);





# SQL | DROP

---

- ▶ DROP is used to delete a whole database or just a table.
- ▶ The DROP statement destroys the objects like an existing database, table, index, or view.
- ▶ A DROP statement in SQL removes a component from a relational database management system (RDBMS).
- ▶ Syntax
  - ▶ `DROP object object_name`
- ▶ Examples:
  - ▶ `DROP TABLE table_name;`
  - ▶ `table_name:` Name of the table to be deleted.



# SQL | ALTER (ADD, DROP, MODIFY)

ROLL_NO	NAME
1	Ram
2	Abhi
3	Rahul
4	Tanu

To ADD 2 columns AGE and COURSE to table Student.

ALTER TABLE Student ADD (AGE number(3), COURSE varchar(40));

ROLL_NO	NAME	AGE	COURSE
1	Ram		
2	Abhi		
3	Rahul		
4	Tanu		



# SQL | ALTER (ADD, DROP, MODIFY)

---

## ► ALTER TABLE - DROP

- DROP COLUMN is used to drop column in a table. Deleting the unwanted columns from the table.

- Syntax:

```
ALTER TABLE table_name
```

```
  DROP COLUMN column_name;
```

# SQL | ALTER (ADD, DROP, MODIFY)

## ► ALTER TABLE-MODIFY

- It is used to modify the existing columns in a table. Multiple columns can also be modified at once.
- Syntax may vary slightly in different databases.
- Syntax

ALTER TABLE table\_name

MODIFY column\_name column\_type;

**ALTER TABLE *table\_name***

**RENAME COLUMN *old\_name* to *new\_name*;**

- Example

ALTER TABLE Student MODIFY COURSE varchar(20);

# Data Manipulation Language

---

- ▶ DML stands for Data Manipulation Language.
- ▶ It is a language used for selecting, inserting, deleting and updating data in a database.
- ▶ It is used to retrieve and manipulate data in a relational database.
- ▶ There are three basic constructs which allow database program and user to enter data and information are:
  - ▶ INSERT
  - ▶ UPDATE
  - ▶ DELETE



# SQL | INSERT

---

- ▶ The `INSERT INTO` statement is used to insert new records in a table.

|

- ▶ Syntax

- ▶ `INSERT INTO table_name VALUES (value1, value2, value3, ...);`



# SQL | INSERT | Example

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

▶ `INSERT INTO Customers VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway')`

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway



# SQL | UPDATE



- ▶ The UPDATE statement in SQL is used to update the data of an existing table in database.
- ▶ We can update single columns as well as multiple columns using UPDATE statement as per our requirement.
- ▶ Syntax

**UPDATE** table\_name

**SET** column1 = value1, column2 = value2, ...

**WHERE** condition;

- ▶ Note: The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!



# SQL | UPDATE | Example



CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

**UPDATE** Customers

**SET** ContactName = 'Alfred Schmidt', City= 'Frankfurt'

**WHERE** CustomerID = 1;



# SQL | UPDATE



## ▶ UPDATE Multiple Records

- ▶ It is the WHERE clause that determines how many records will be updated.

```
UPDATE Customers
```

```
SET ContactName='Juan'
```

```
WHERE Country='Mexico';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden



# SQL | DELETE

---

- ▶ The DELETE Statement in SQL is used to delete existing records from a table.
- ▶ We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.
  - ▶ Syntax
    - ▶ **DELETE FROM table\_name WHERE condition;**
  - ▶ The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!



# SQL | DELETE

---

- ▶ Delete All Records
  - ▶ It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:
- ▶ Syntax
  - ▶ **DELETE FROM** table\_name;
- ▶ Example
  - ▶ **DELETE FROM** customers;

Attribute	Datatype	Constraint
Roll No	Varchar(5)	Primary key
Name	Varchar(10)	Not Null
Subjectcode	Varchar(5)	References Subject
Age	Int	

1. Create the following table with constraints

Attribute	Datatype	Constraint
Roll No	Varchar(5)	Primary key
Name	Varchar(10)	Not Null
Subjectcode	Varchar(5)	References Subject
Age	Int	

1. Create the following table with constraints

**Create table** student (rollno varchar(5),name varchar(10) not null,subjectcode varchar(5) age int, **primary key**(rollno),  
**foreign key**(subjectcode) **references** subject);

Rollno	Name	Subjectcode	Age
1	Anil	CS203	21
2	Akash	CS206	22

2. Insert the following table student

3. Add a new column University  
varchar(10)

Rollno	Name	Subjectcode	Age
1	Anil	CS203	21
2	Akash	CS206	22

2. Insert the following table student

Insert into student values('1', 'Anil', 'CS203', 21)

Insert into student values('2', 'Akash', 'CS206', 22')

3. Add a new column University  
varchar(10)

Alter table student add(university varchar(10);

Rollno	Name	Subjectcode	Age	University
1	Anil	CS203	21	Kerala
2	Akash	CS206	22	KTU

#### 4. Update the university column of the table

**Update student set university='kerala' where rollno='1';**

**Update student set university='KTU' where rollno='2';**

#### 5. Delete the record of roll no 2

**Delete from student where rollno='2';**

#### 6. Delete all records from table student

**Delete from student;**

Rollno	Name	Subjectcode	Age	University
1	Anil	CS203	21	Kerala
2	Akash	CS206	22	KTU

4. Update the university column of the table

5. Delete the record of roll no 2

6. Delete all records from table student

- The SELECT-FROM-WHERE Structure of SQL Queries

SELECT <attributes>

FROM <table>

WHERE<condition>

Used to retrieve/select rows from the table based on a condition

QUERY Retrieve the birthdate and address of the employee(s) whose name is 8John B. Smith9

QUERY Retrieve the name and address of all employees who work for the 8Research9 department.

### Tables as Sets in SQL QUERY

Retrieve the salary of every employee and all distinct salary values.

SELECT ALL SALARY FROM EMPLOYEE;

SELECT DISTINCT SALARY□λ□ FROM EMPLOYEE;

- a) Write SQL DDL statements based on the following database schema (Assume suitable domain types):

Employee (eid, name, designation, salary, comp\_id)

Company (comp\_id, cname, address, turnover)

- b) Create the above mentioned tables assuming each company has many employees. Mention the primary key, foreign key and not null constraints.
- ii) Insert values into both the tables. Mention in which order insertions will be carried out.
- ii) Modify the table Employee to include a new column “years\_of\_exp”
- iv) Increment the salary of employees whose salary is less than Rs25000 by 5%

16 a) Consider the following relations which are part of the University database. (8)

Instructor (id, name, dept\_name, salary)

Course (course\_id, title, dept\_name, credits)

Section (course\_id,sec\_id,semester,year,building,room\_number,time\_slot\_id).

Write SQL queries for the given questions:

- i) Find the titles of courses in the IT department that have 4 credits.
  - ii) Find all instructors earning the highest salary.
  - iii) Increase the salary of each instructor in the IT department by 10%.
  - iv) Delete all courses that have never been offered.