Module 2 Syllabus

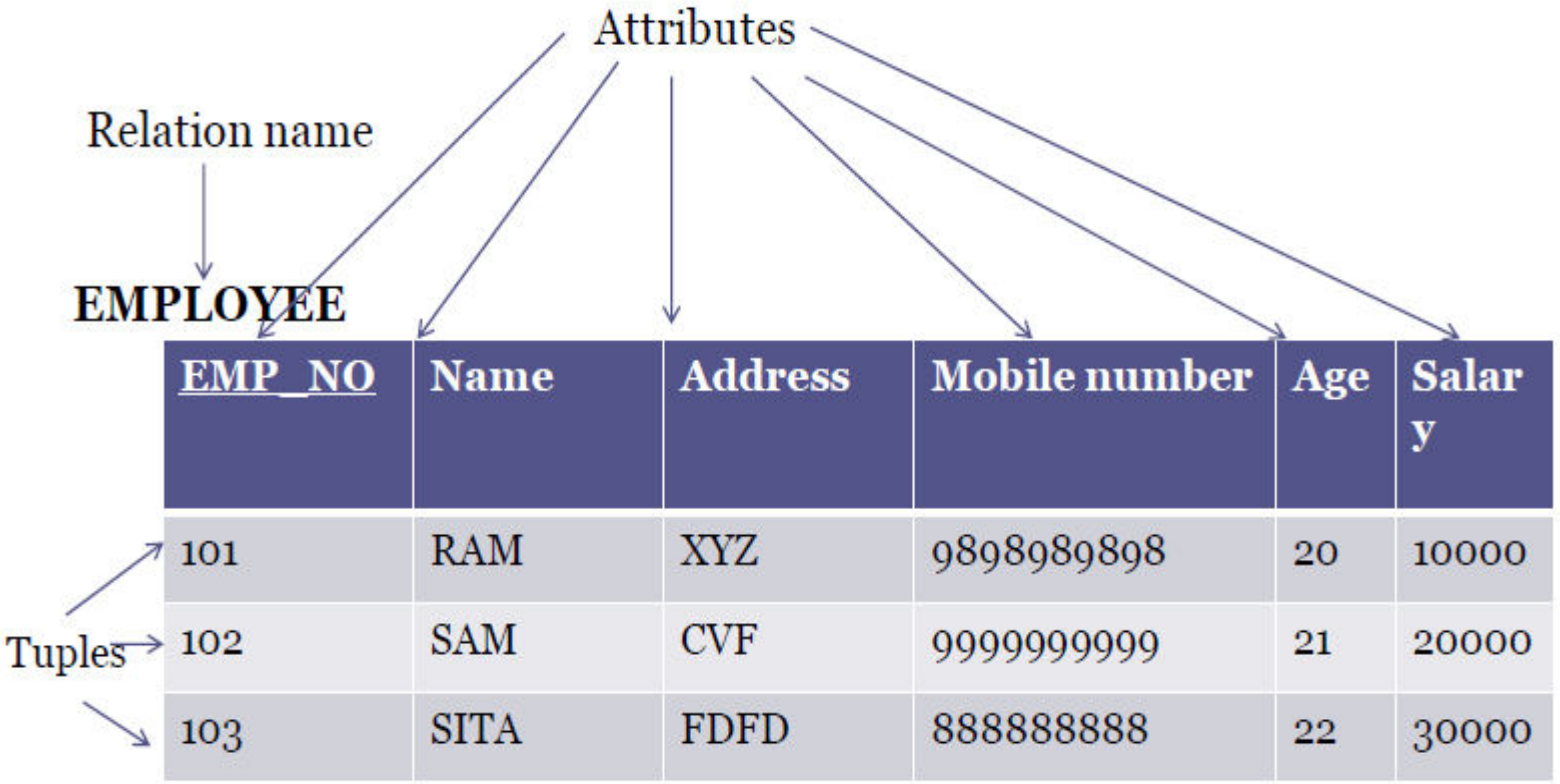
- Structure of Relational Databases – Integrity Constraints, Synthesizing ER diagram to relational schema
- Introduction to Relational Algebra - select, project, cartesian product operations, join - Equi-join, natural join. query examples,
- Introduction to Structured Query Language(SQL), Data Definition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

Database Management System
Module 2

Lect 1 : Relational data model

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



EMP_NO	Name	Address	Mobile number	Age	Salary
101	RAM	XYZ	9898989898	20	10000
102	SAM	CVF	9999999999	21	20000
103	SITA	FDFD	888888888	22	30000

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 :Data

Values : Jan, Feb, Mar.....

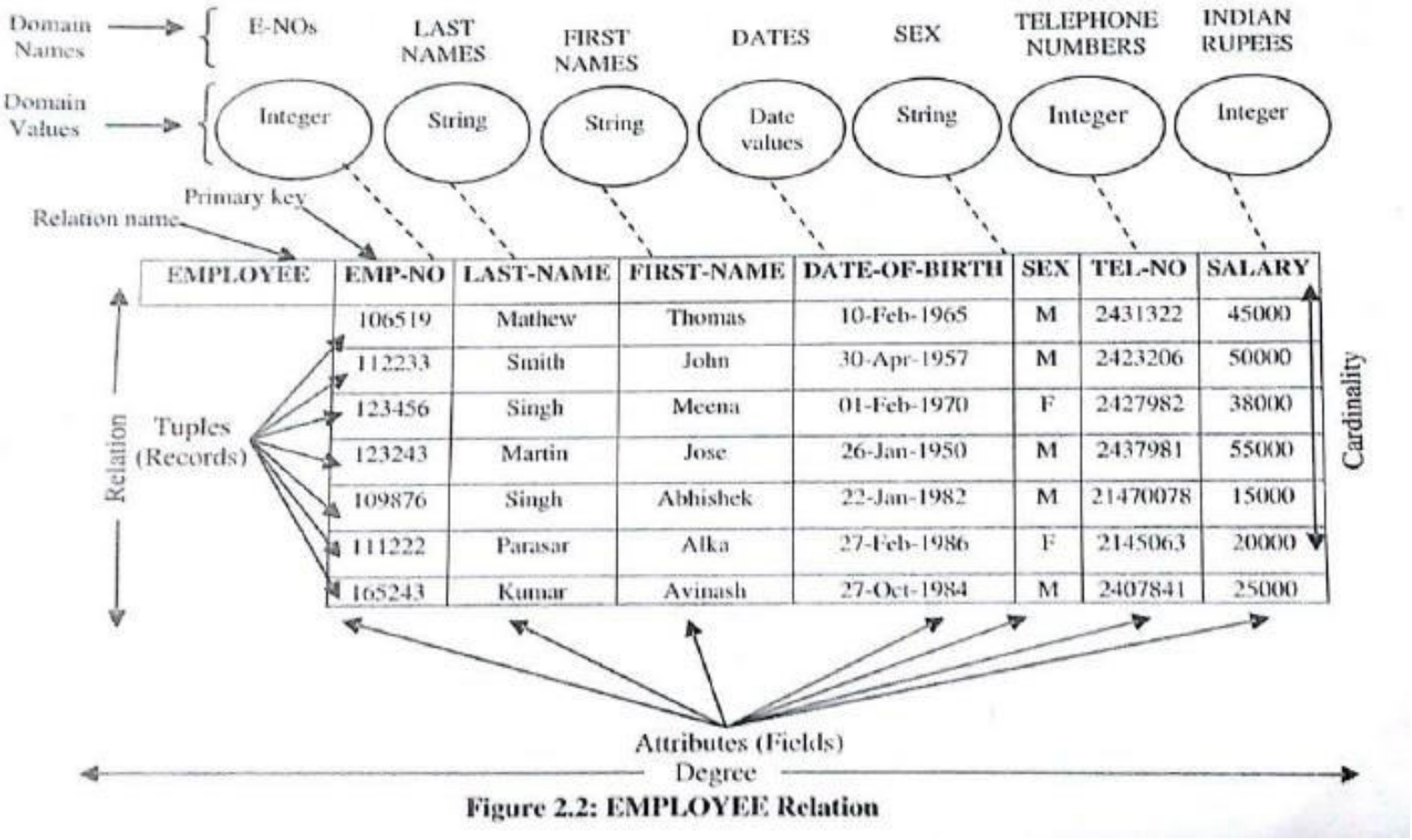
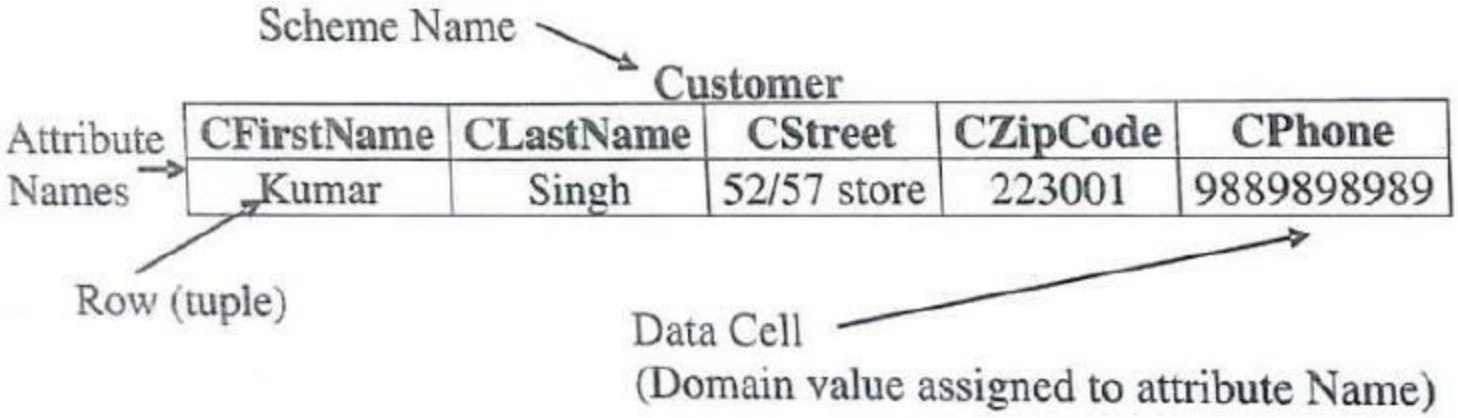


Figure 2.2: EMPLOYEE Relation

Tuples (rows)

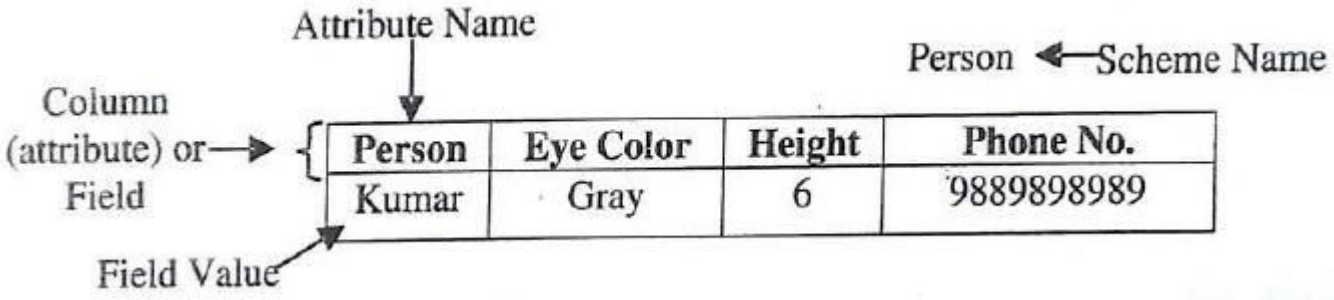
- A tuple is an ordered set of values
- Tuple is a portion of a table containing data that described only entity,
- Also known as record
- Each value is derived from an appropriate domain.



- **<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_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	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 (A1, A2, ...An)*
  - R is relation name and it has some attributes A1 to An
- Each attribute have some domain and it is represented by dom(Ai)
- 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 overthe domain of character strings of maximumlength 25
  - dom(Cust-name) is varchar(25)

- A relation state r(R) is a mathematical relation of degree n on the domains dom(A1), dom(A2)…, dom(An) which is a subset of Cartesian product(X) of domains that define R

$$r(R) \subseteq (dom(A1) X\ dom(A2)\ X\ \ldots . X\ dom(An))$$

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

Definition Summary

Informal Terms		Formal Terms
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Database Management System
Module 2

Lect 2 : Relational Integrity Constraint

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	TRC

- 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	TRC

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},

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

- 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.

EMPLOYEE

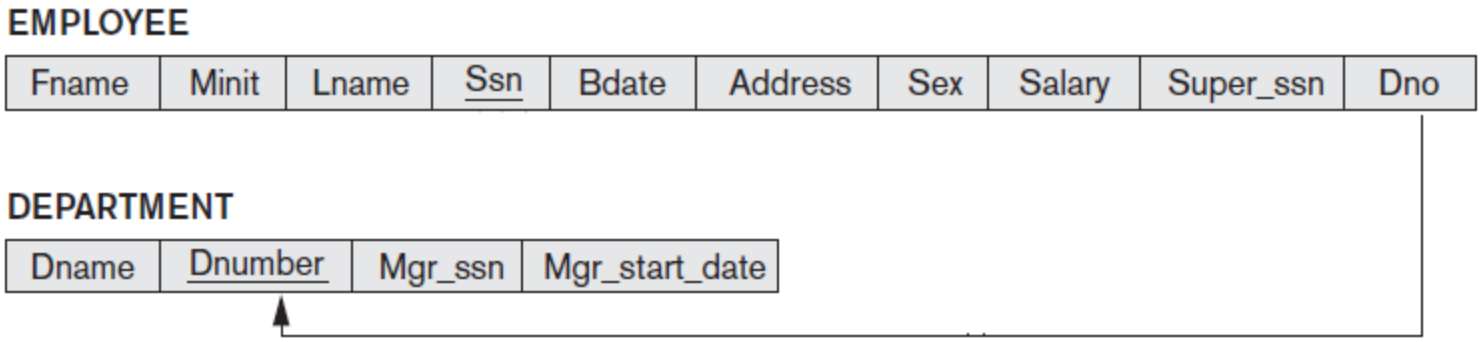
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	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

- 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.



Example 2

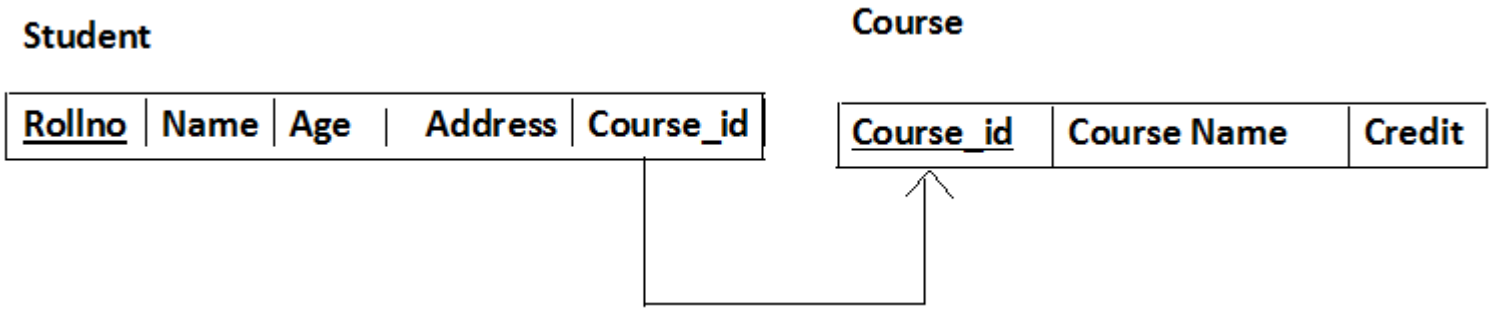
Student

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

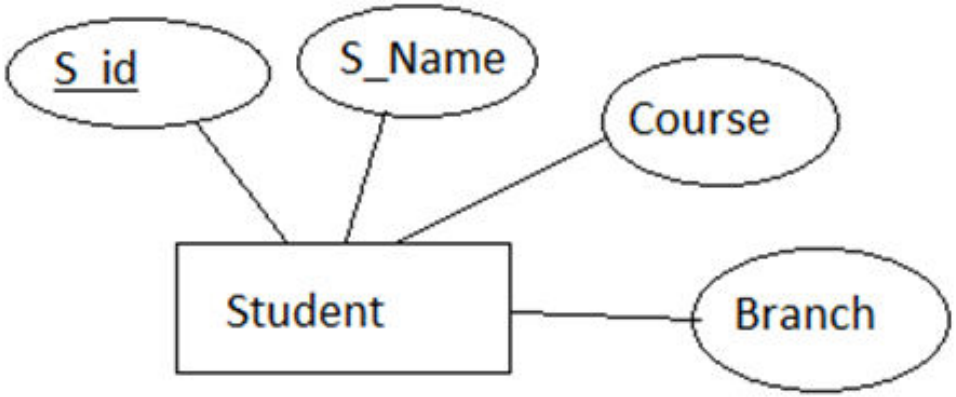


Database Management System
Module 2

Lect 3 : Converting ER diagram to Relational Schema

Converting ER diagram to relational Schema

1) Converting Strong Entity set into Relational Schema

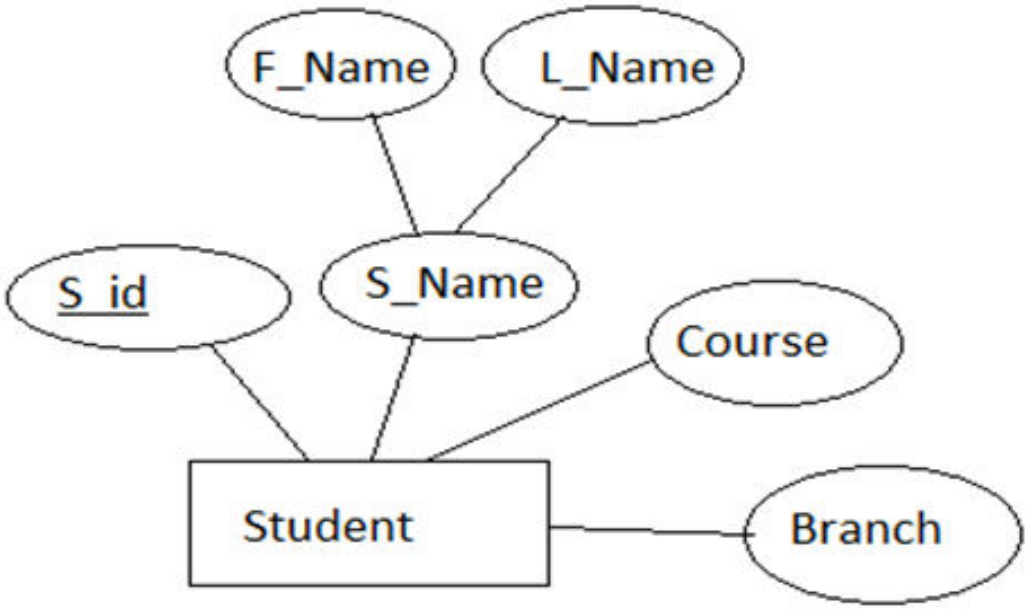


Student

S_id	S_Name	Course	Branch

2) Composite Attributes to Relational schema

Make F_Name,L_Name as individual attribute

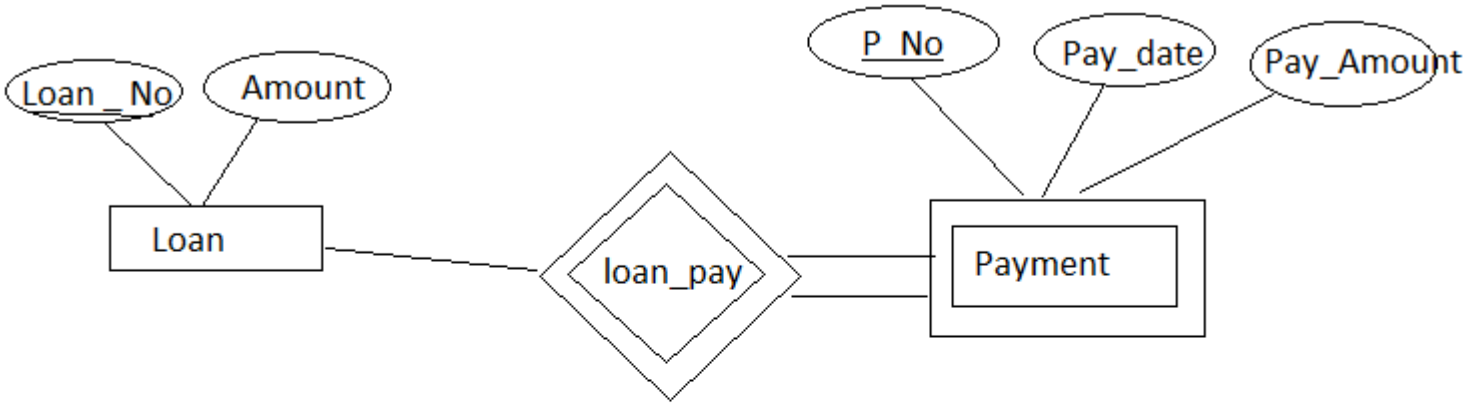


Student

S_id	F_Name	L_Name	Course	Branch

3) Representing weak entity set in relational schema

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



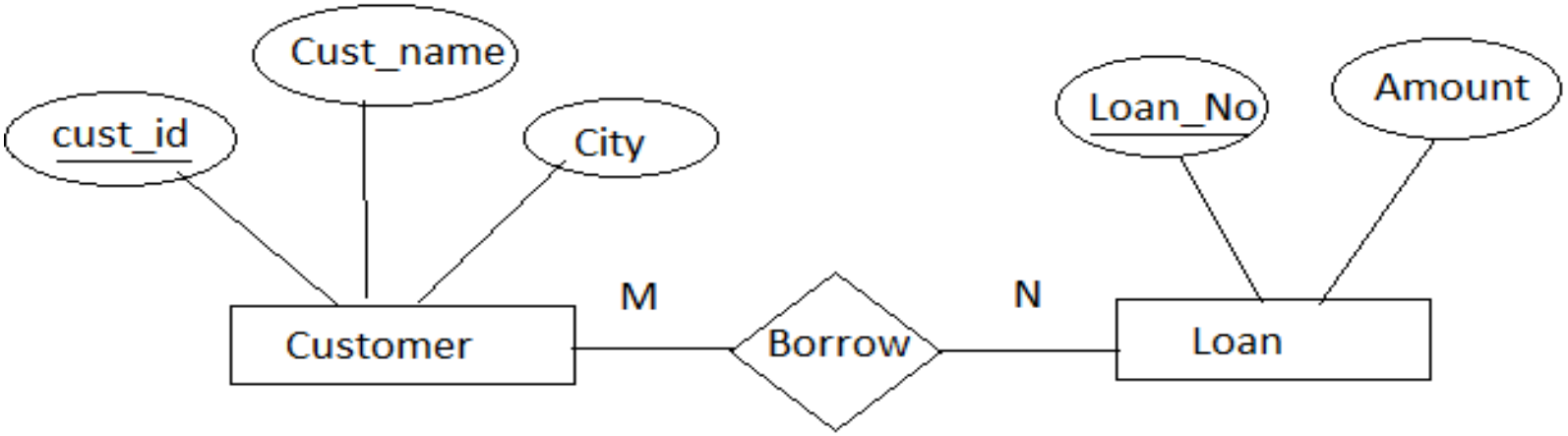
Loan

Loan_No	Amount

Payment

P_NO	Pay_date	Pay_amount	Loan_no

4) Representing Many- Many relationship in relational schema



Customer

Cust_id	Cust_name	City

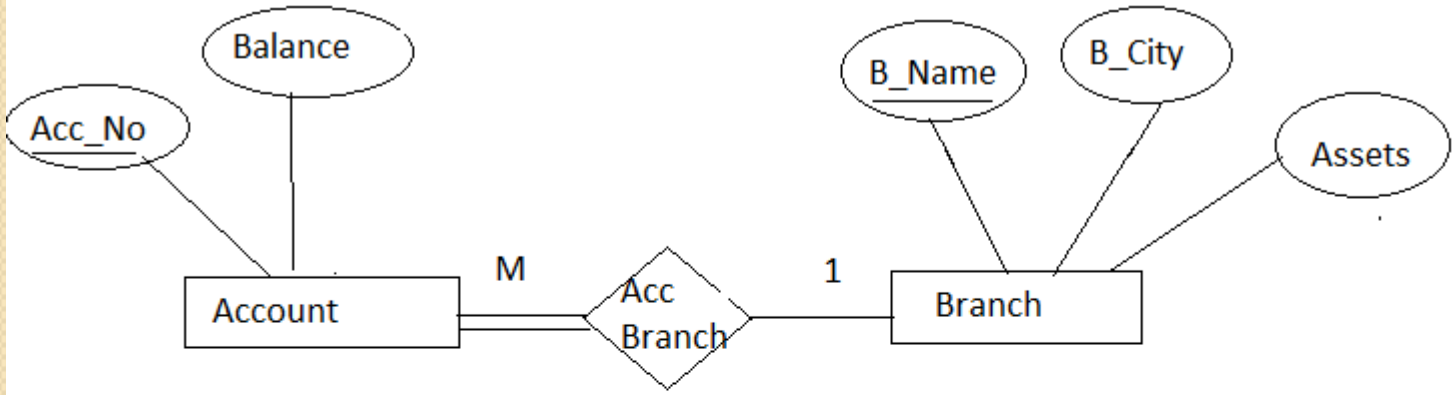
Loan

Loan_no	Amount

Borrow

Cust_id	Loan_no

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



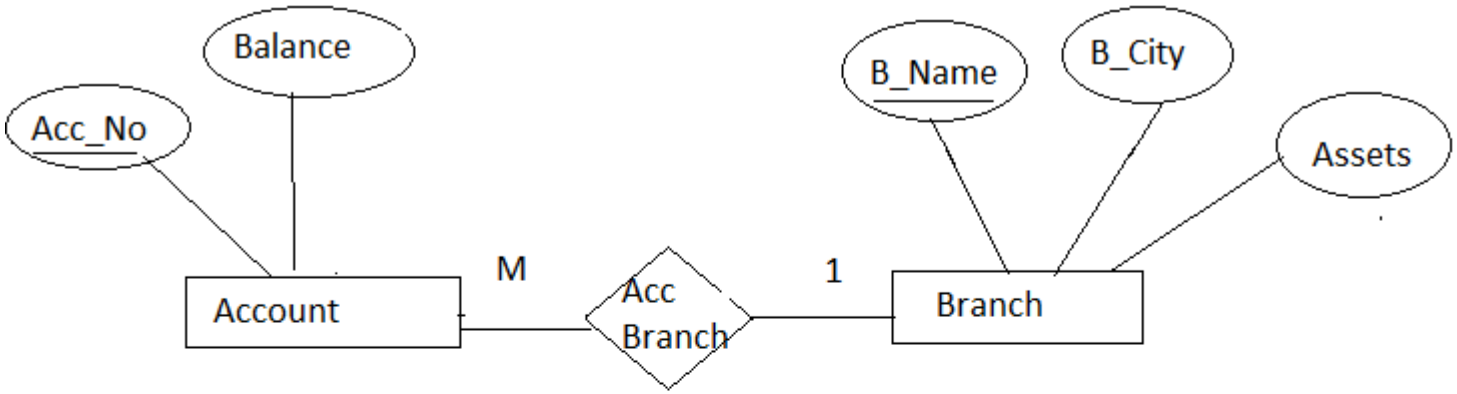
Account

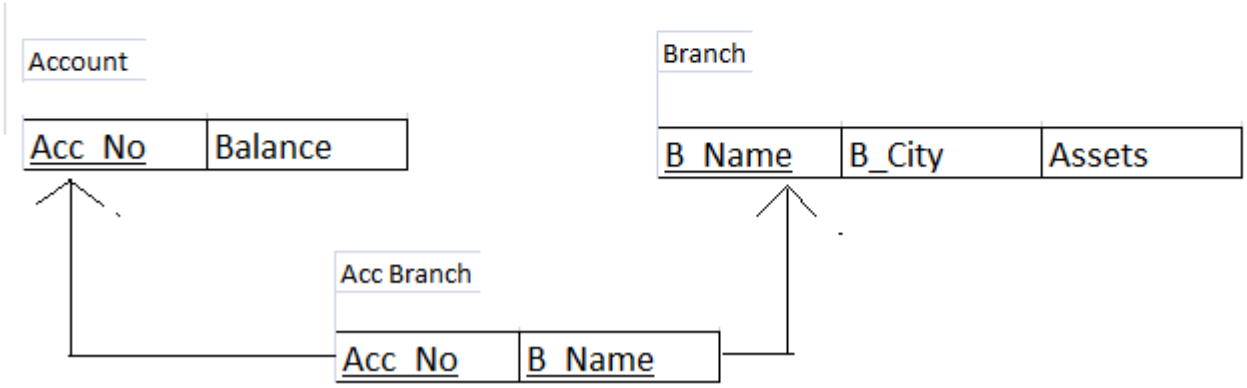
Acc_No	Balance	B_Name

Branch

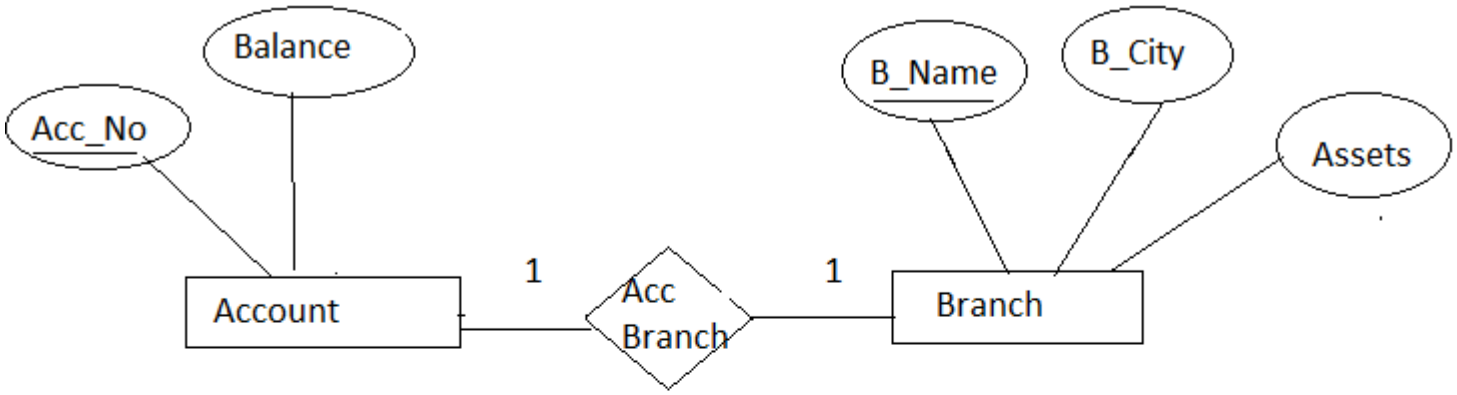
B_Name	B_City	Assets

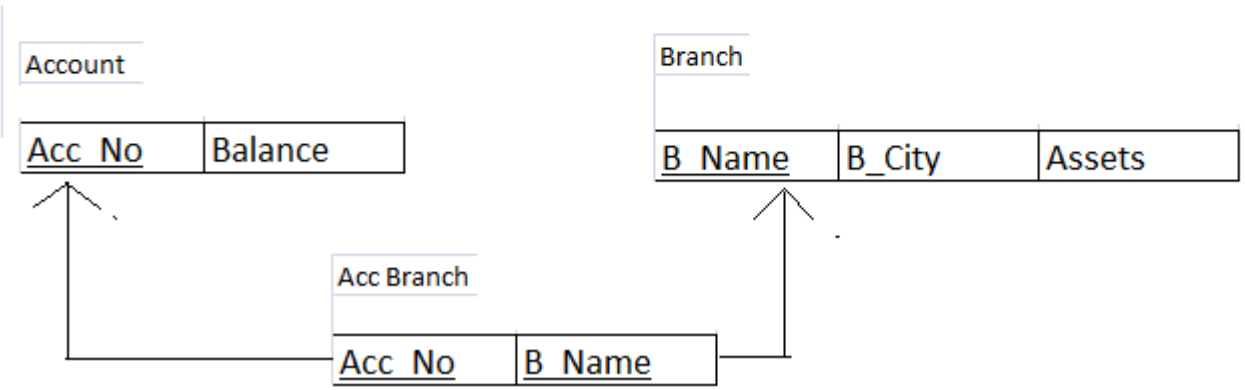
6) Representing Many- One or One- Many relationship in relational schema



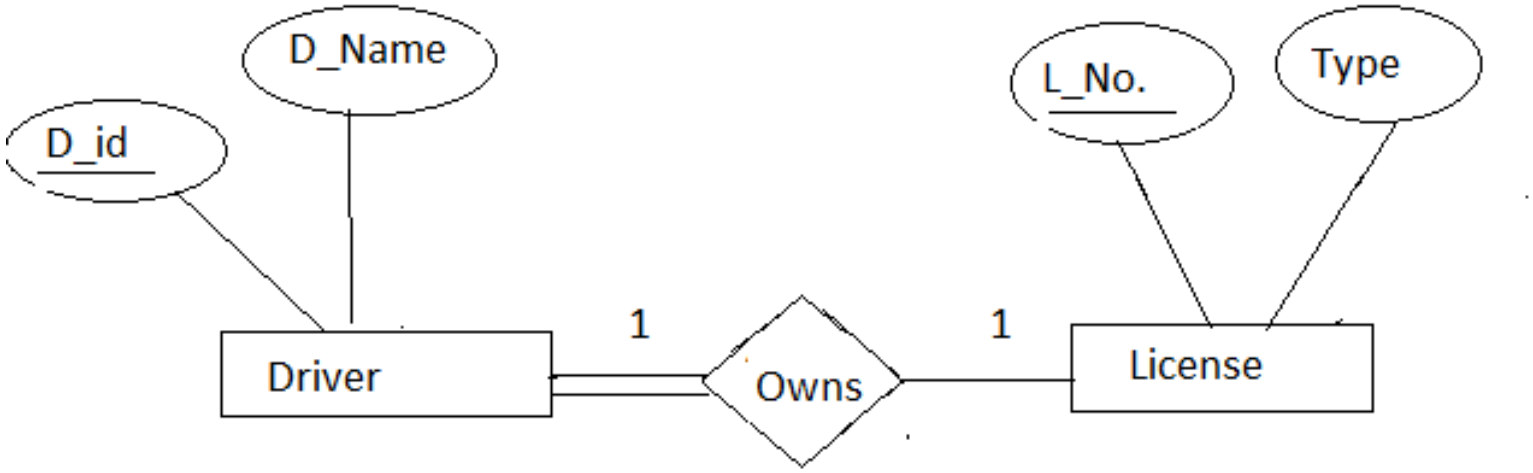


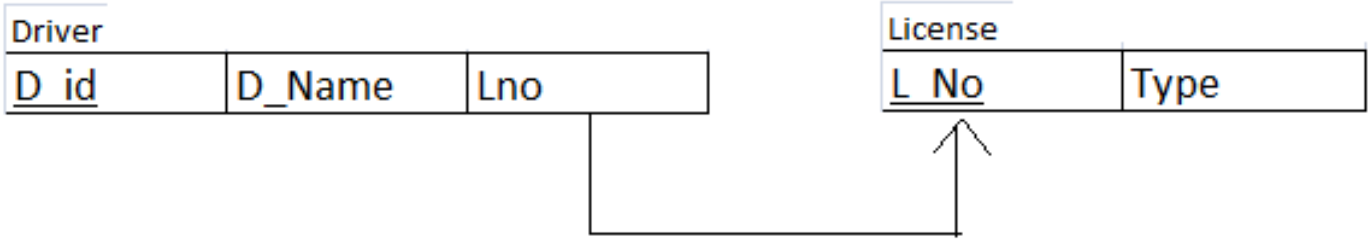
7) Representing One- One relationship in relational schema



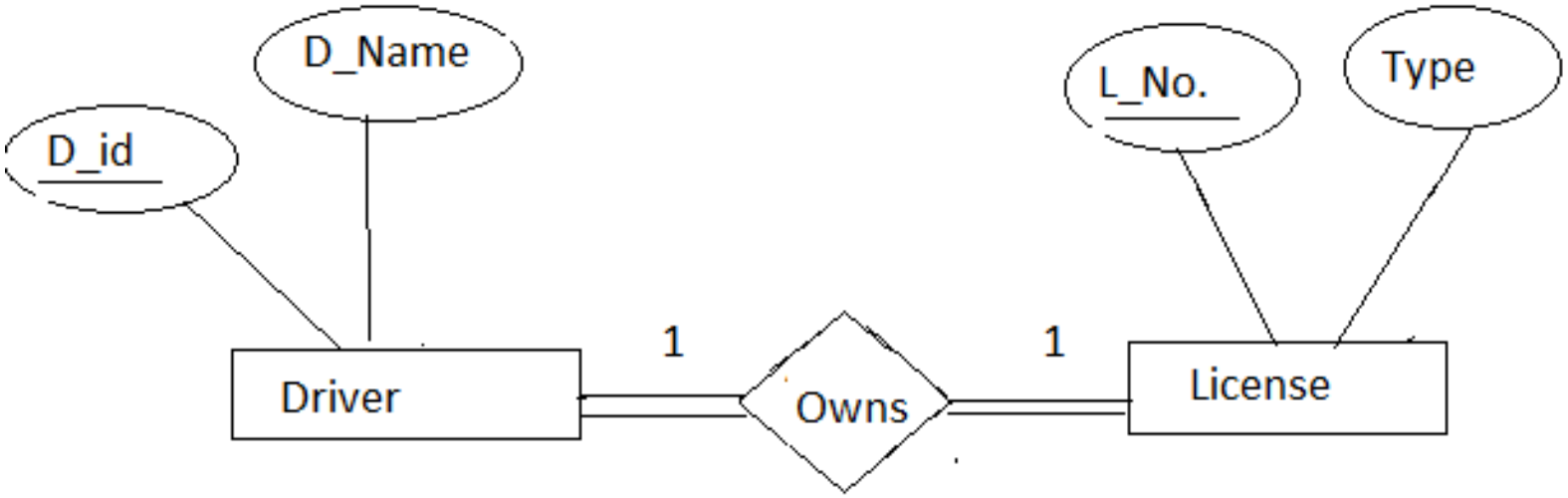


8) Representing One- One relationship with total participation in relational schema



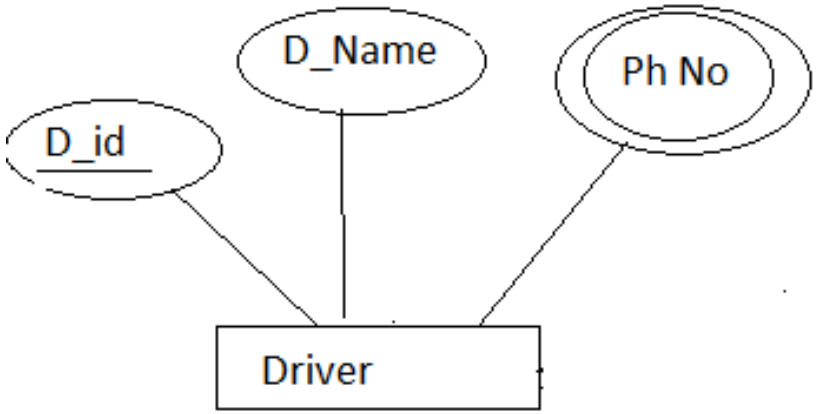


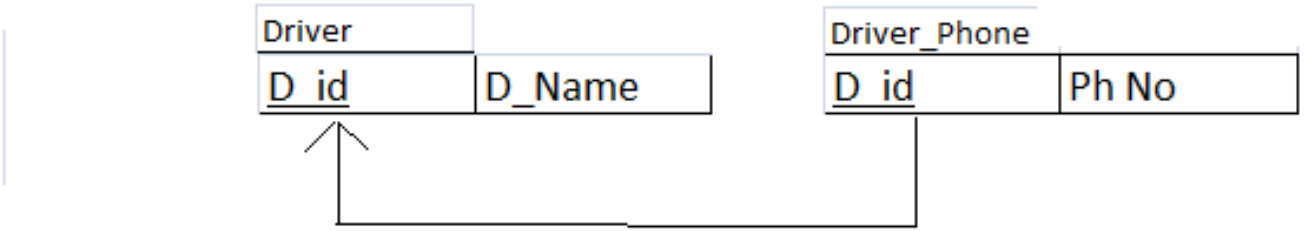
9) Representing One- One relationship with total participation at both ends in relational schema



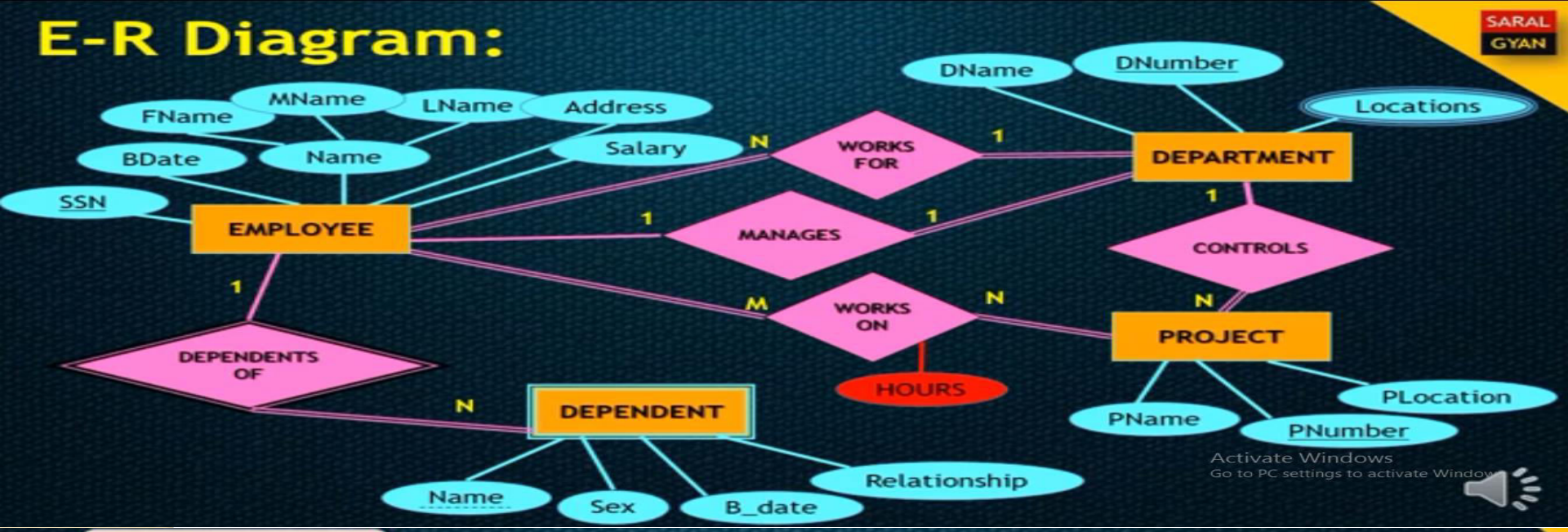
D_id	D_Name	L_No	Type

10) Representing Multivalued attributes in relational schema





E-R Diagram:



Employee

SSN	Bdate	FName	MName	LName	Address	Salary	DNumber

DEPARTMENT

DName	DNumber	SSN

DEPT_LOCATION

DNumber	Location

Project

PNumber	PName	PLocation	DNumber

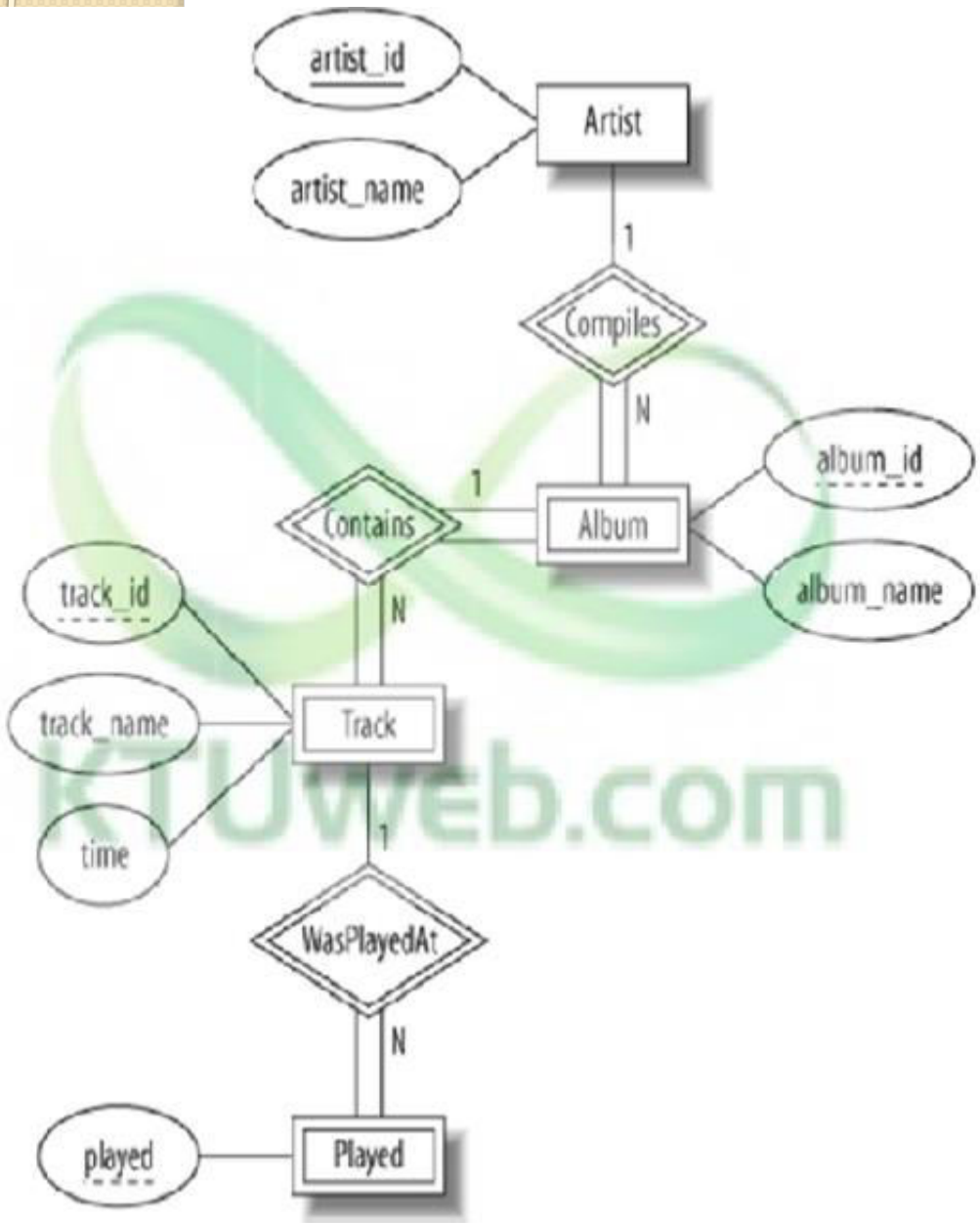
WORKS_ON

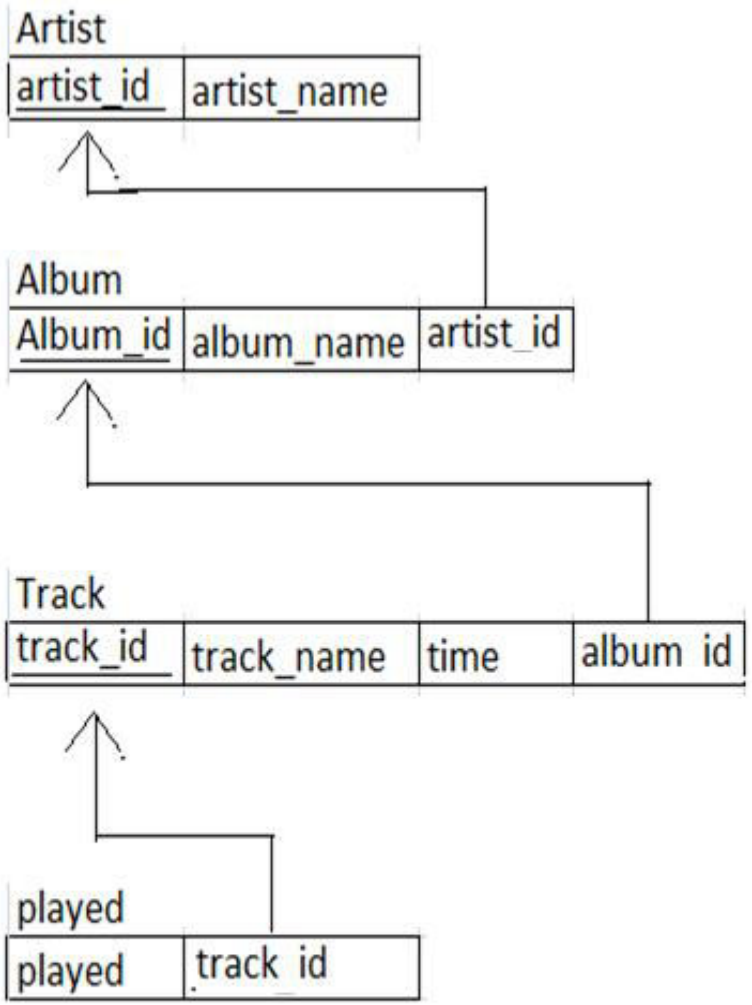
SSN	PNumber	Hours

Dependent

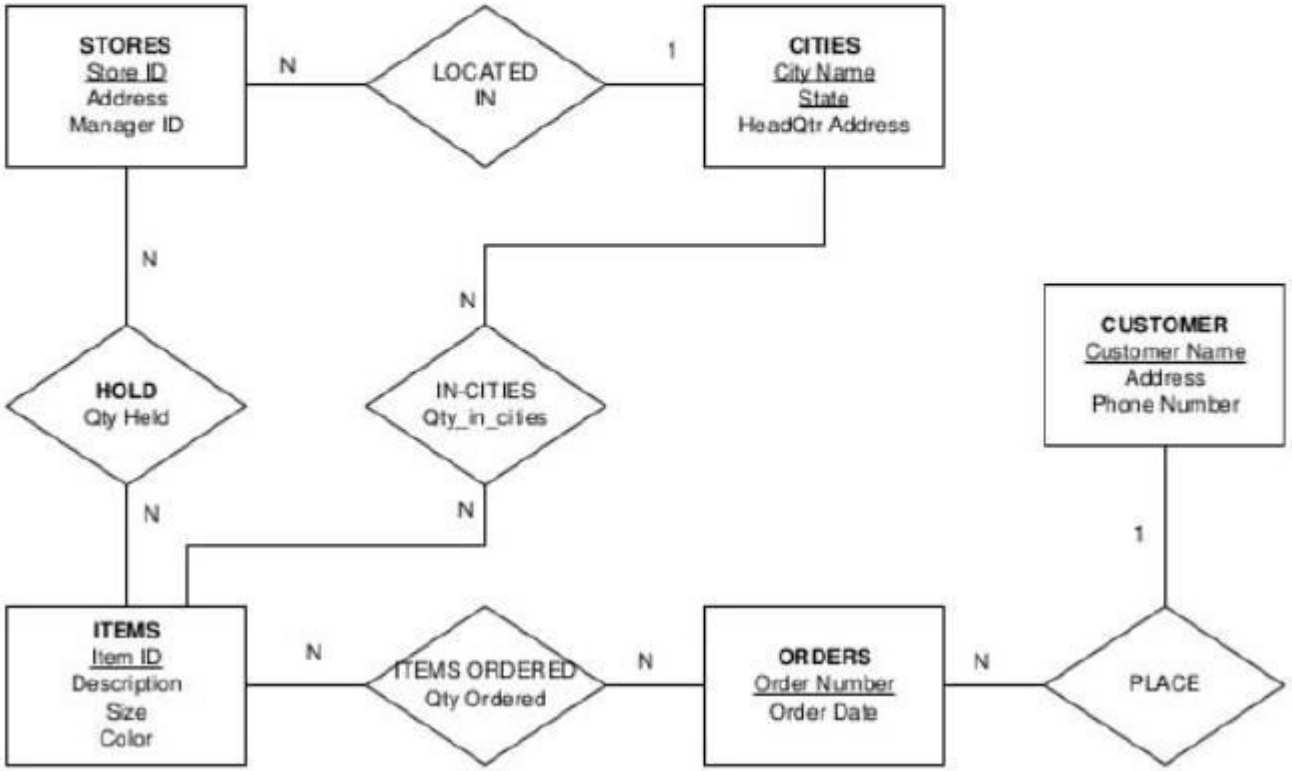
SSN	Dp_Name	Sex	B_Date	Relation

Foreign Key

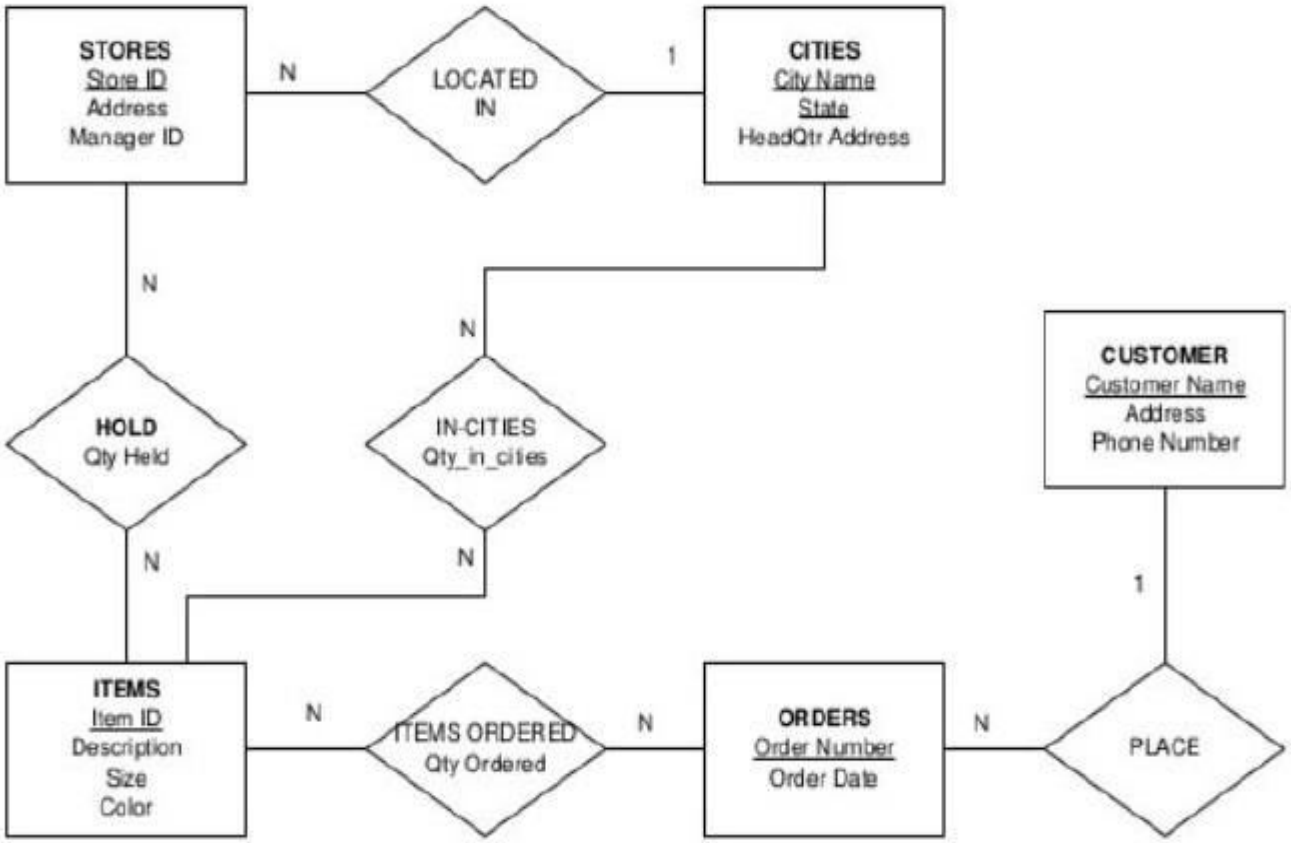

artist_id
artist_name
Artist
1
Compiles
N
album_id
1
Contains
Album
album_name
track_id
N
track_name
Track
time
1
WasPlayedAt
N
played
Played

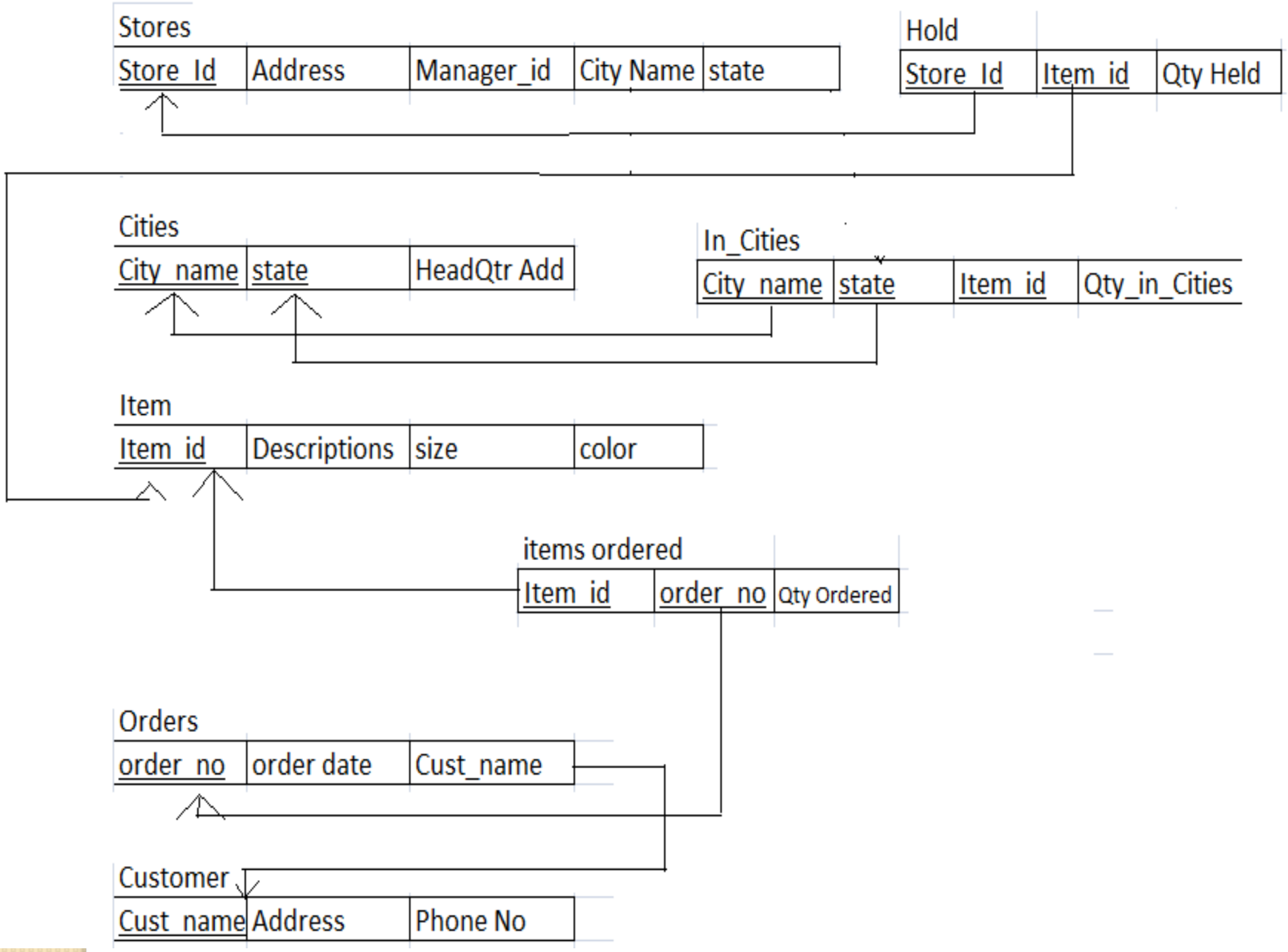

Artist
artist_id
artist_name
Album
Album_id
album_name
artist_id
Track
track_id
track_name
time
album_id
played
played
track_id

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.



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.




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

Database Management System
Module 2

Lect 3a : **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*

The SELECT Operation

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 select the EMPLOYEE tuples whose department is 4, or those whose salary is greater than $30,000

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

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:

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

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

$$\pi_{<\text{attribute list}>}(R)$$

(pi) is the symbol used to represent the PROJECT operation,

<attribute list> 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:

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

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

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

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

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

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

The UNION, INTERSECTION and MINUS Operations

- Two relations $R(A1, A2, ..., An)$ and $S(B1, B2, ..., Bn)$ 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.

(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

STUDENT ∪ INSTRUCTOR.

(b)

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

STUDENT ∩ INSTRUCTOR.

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

STUDENT – INSTRUCTOR.

(d)

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

INSTRUCTOR – STUDENT.

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

CARTESIAN PRODUCT (CROSS PRODUCT) Operation

- The CARTESIAN PRODUCT operation—also known as CROSS PRODUCT or CROSS JOIN—which is denoted by X .
- For R X 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

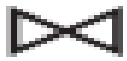
Instructor

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

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

Customer

Cid	Cname	Age
101	Ajay	20
102	Vijay	19
103	Sita	21
104	Gita	22

Order

Cid	Oname	Cost
101	Pizza	500
103	Noodles	300
108	Burger	99

Customer ⋈ Order

Cid	Cname	Age	Oname	Cost
101	Ajay	20	Pizza	500
103	Sita	21	Noodles	300

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

Cid	Cname	Age
101	Ajay	20
102	Vijay	19
103	Sita	21

Order

Oid	Oname
101	Pizza
101	Noodles
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	1955-12-08	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

$$DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE$$

$$RESULT \leftarrow \pi_{Dname,\ Lname,\ Fname}(DEPT\_MGR)$$

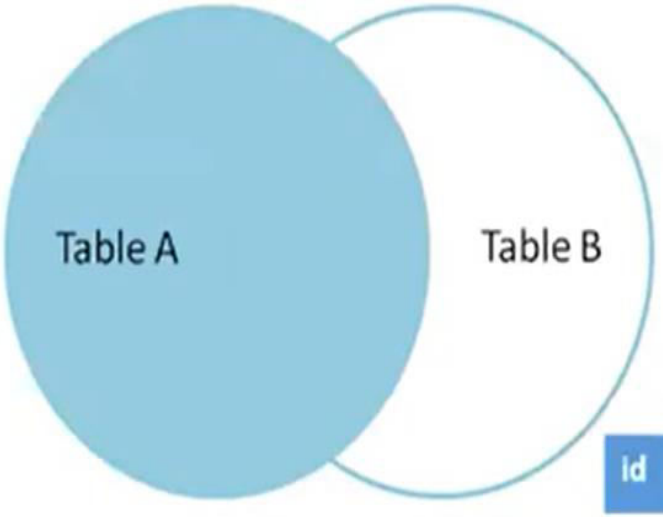
$$DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE$$
$$RESULT \leftarrow \pi_{Dname,\ Lname,\ Fname}(DEPT\_MGR)$$

We can also combine the relational algebra expression as follows

$$\pi_{Dname,\ Lname,\ Fname}(DEPT\_MGR) \quad \left(DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE \right)$$

LEFT JOIN (⟕)

- 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

Id	Name
10	Jay
20	Veer
30	John

B

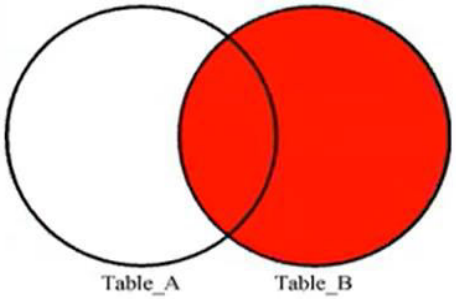
Name	Marks
Rohan	20
Veer	18
John	14
Sam	13

A ⟕ B

Id	Name	Marks
10	Jay	NULL
20	Veer	18
30	John	14

RIGHT JOIN(⋈)

- 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

Id	Name
10	Jay
20	Veer
30	John

B

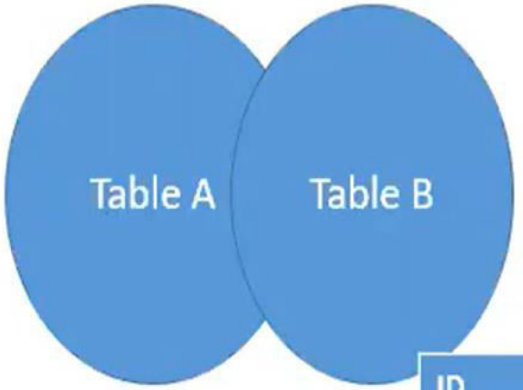
Name	Marks
Rohan	20
Veer	18
John	14
Sam	13

A ⋈ B

id	Name	Marks
Null	Rohan	20
20	Veer	18
30	John	14
Null	Sam	13

FULL JOIN (⋈)

- 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 ⋈ B

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

A

Id	Name
10	Jay
20	Veer
30	John

B

Name	Marks
Rohan	20
Veer	18
John	14
Sam	13

Prev Year University Questions on Relational Algebra

The relational schema for a library describing members, books and issue information is given below. Foreign keys have the same name as primary keys.

BOOKS(ACC-NO, ISBN, TITLE, EDITION, YEAR)

MEMBERS(MEMBERID, MEMBERNAME, MEMBERTYPE)

ISSUEDTO(ACC-NO, MEMBERID, DATE OF ISSUE)

Write relational algebra expressions for the following queries:

a) Accession Number(s) and Name(s) of third edition books published in 2018. (2)

b) Names and dates of issue of books taken by a member with name 'PRIYA'. (3)

c) Names of books *not* taken by any member. (4)

(a)

$$\Pi_{name,\ AccNo}\left(\sigma_{Edition='3'\ AND\ Year='2018'}(Book)\right)$$

(b)

$$A \leftarrow \sigma_{Membername='PRIYA'}(Members)$$

$$B \leftarrow A \bowtie_{A.memberid = Issuedto.memberid} Issuedto$$

$$C \leftarrow B \bowtie_{B.AccNo = Books.AccNo} Books$$

$$Result \leftarrow \Pi_{Title,\ date\ of\ issue}(C)$$

(c)

$$A \leftarrow \Pi_{AccNO}(Book)$$ { Acc No. of Entire books }

$$B \leftarrow \Pi_{AccNO}(ISSUED\ TO)$$ { Acc No of Issued books }

$$C \leftarrow A - B$$

$$\Pi_{title}\left(C \bowtie_{C.AccNo = Books.AccNo} Books\right)$$

Study the tables given below and write relational algebra expressions for the queries that follow. (9)

STUDENT(ROLLNO, NAME, AGE, GENDER, ADDRESS, ADVISOR)
COURSE(COURSEID, CNAME, CREDITS)
PROFESSOR(PROFID,PNAME, PHONE)
ENROLLMENT(ROLLNO, COURSEID, GRADE)

Primary keys are underlined. ADVISOR is a foreign key referring to PROFESSOR table. ROLLNO and COURSEID in ENROLLMENT are also foreign keys referring to THE primary keys with the same name.

(i) Names of female students
(ii) Names of male students along with adviser name
(iii) Roll Number and name of students who have not enrolled for any course.

Soln

(a) $\Pi_{name}\left(\sigma_{gender='Female'}(Student)\right)$

(b)
$A \leftarrow \sigma_{gender='male'}(Student)$

$B \leftarrow A \bowtie_{advisor = Profid} Professor$

$Result \leftarrow \Pi_{name,\ pname}(B)$

(c)
$A \leftarrow \Pi_{rollno}(student)$

$B \leftarrow \Pi_{rollno}(Enrollment)$

$C \leftarrow A - B$

$D \leftarrow C \bowtie_{C.rollno = Student.rollno} Student$

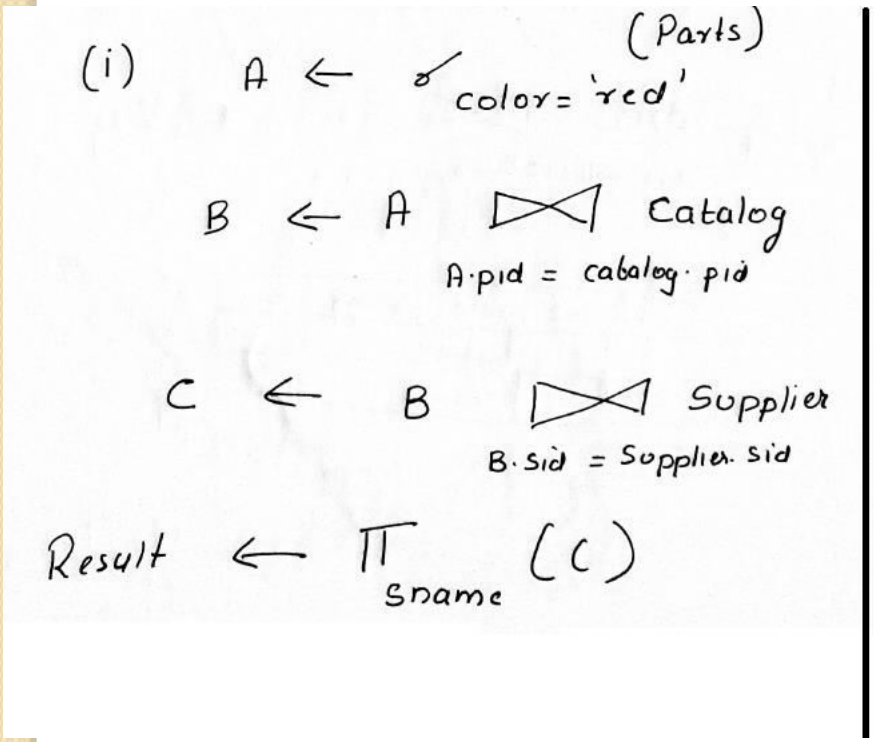
$Result \leftarrow \Pi_{rollno,\ name}(C)$

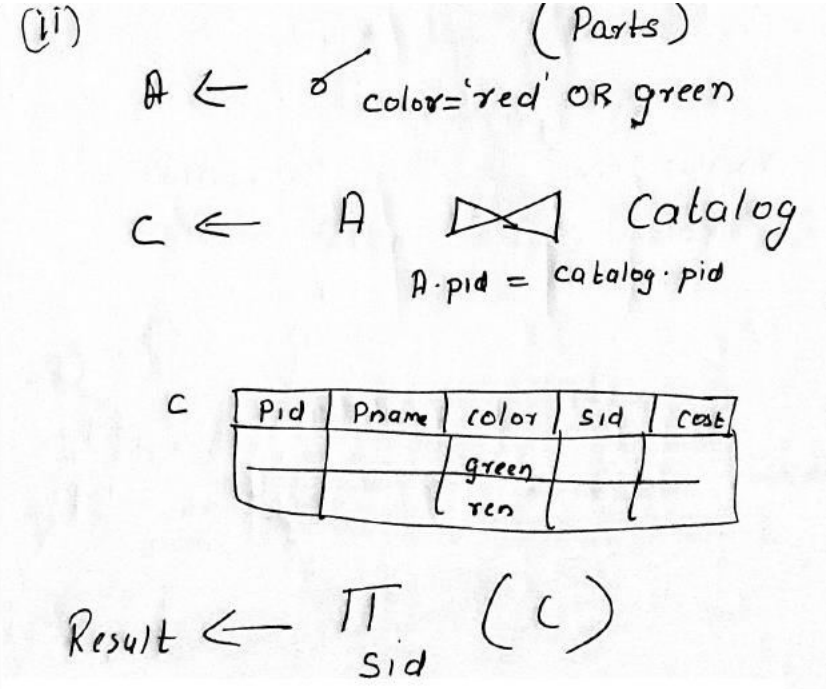
Consider the following database with primary keys underlined

Suppliers (sid, sname, address)
Parts (pid, pname, color)
Catalog (sid, pid, cost)

sid is the key for Suppliers, pid is the key for Parts, and sid and pid together form the key for Catalog. The Catalog relation lists the prices charged for parts by Suppliers.

Write relational algebra for the following queries: -
i) Find then names of suppliers who supply some red part
ii) Find the sid's of suppliers who supply some red or green part
iii) Find the sid's of suppliers who supply some red part and some green part.





(iii)

$A \leftarrow \sigma_{color = 'Red' \text{ AND } 'Green'}$ (Parts)

A

Pid	Pname	color
		Red
		Green

$B \leftarrow A \bowtie$ catalog

$A.pid = catalog.pid$

B

Pid	Pname	color	sid	cost

$Result \leftarrow \Pi_{sid}$ (B)

The relational database schema below represents certain information about albums, songs in the albums and singers of those songs. Foreign keys are given the *same* name as primary keys for easy identification. (9)

ALBUMS(ALBUM#, ALBUM-NAME, PRODUCED-BY, YEAR)
SONGS(SONG#, SONG-START, DURATION, ALBUM#)
SUNGBY(ARITISTNAME, SONG#)

In the context of the schema, write relational algebra expressions for the following queries:

(a) Names of albums produced by 'HMV' in the year 2018. (b) Names of albums in which an artist with name, 'AVANTHIKA' sung. (c) Names of albumsin which *all* the artists have sung songs.

(a)

$$\Pi_{\text{Album name}}\left(\sigma_{\text{year} = 2018 \text{ AND produced by} = \text{'HMV'}}(\text{Albums})\right)$$

(b)

$$A \leftarrow \sigma_{\text{Artist name} = \text{'AVANTHIKA'}}(\text{Sung by})$$

A ⇒

Artist Name	Sogn#
Avonthika	—

$$B \leftarrow A \bowtie_{A.\text{Song\#} = \text{Song.Song\#}} \text{Songs}$$

B

Song#	Song start	duration	Album#	Artist name

$$C \leftarrow B \bowtie_{B.\text{Album\#} = \text{Albums.Album\#}} \text{Albums}$$

$$\text{Result} \leftarrow \Pi_{\text{Album name}}(C)$$

(c)

$$A \leftarrow \text{Sung by} \bowtie_{\text{Sung by.Song\#} = \text{Songs.Song\#}} \text{Songs}$$

A

Song#	Song start	duration	Album#	Artist name

$$B \leftarrow A \bowtie_{A.\text{Album\#} = \text{Albums.Album\#}} \text{Albums}$$

B

Album#	Album name	Prod by	year	Song#	Song start	duration	Artist name

$$\text{Result} \leftarrow \Pi_{\text{albumname}}(B)$$

Q) Consider the query

"Select Name, Age from Student Where Gender = 'Male'" on the table Student (Rollno, Name, Age, Gender, Address). Give a relational algebra expression corresponding to the query

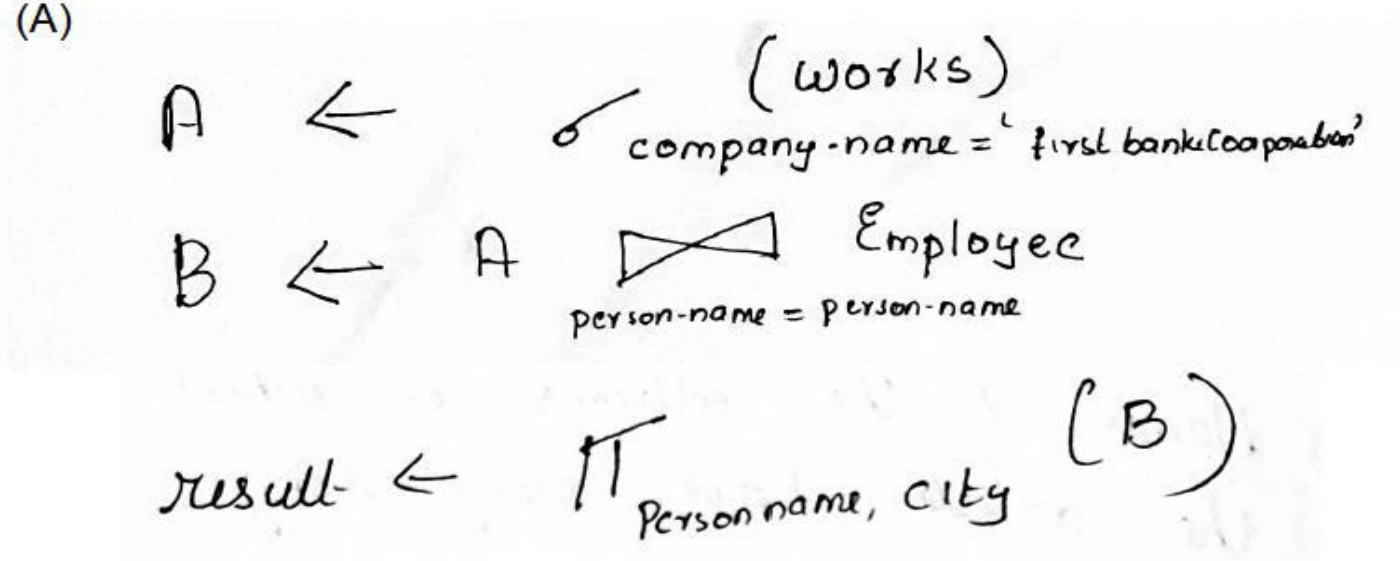
$$\Pi_{name,\ age}\left(\sigma_{gender='male'}(Student)\right)$$

Consider the schema given below.

employee (person-name, street, city)
works (person-name, company-name, salary)
company (company-name, city)
manages (person-name, manager-name)

Write relational algebra queries for the following questions
a. Find the names and cities of residence of all employees who work for First Bank Corporation.
b. Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000 per annum.
c. Find the names of all employees in this database who live in the same city as the company for which they work.

(A)



(B)

$A \leftarrow \sigma_{\text{company-name} = \text{'firstbank Coorperation'} \text{ AND } \text{Salary} > \$10,000}(\text{works})$

$B \leftarrow A \bowtie_{\text{person-name} = \text{person-name}} \text{Employee}$

$\text{Result} \leftarrow \Pi_{\text{personname, street, city}}(B)$

(C)

$A \leftarrow \text{Employee} \bowtie_{\text{city} = \text{city}} \text{Company}$

$\text{result} \leftarrow \Pi_{\text{person.name}}(A)$

Database Management System
Module 2

Lect 4 : Introduction to Structured Query Language

History

- 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-constraint$_1$),
  ...,
  (integrity-constraint$_k$))

  - r is the name of the relation
  - each A_i is an attribute name in the schema of relation r
  - D_i is the data type of values in the domain of attribute A_i
- Example:

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

Integrity Constraints in Create Table

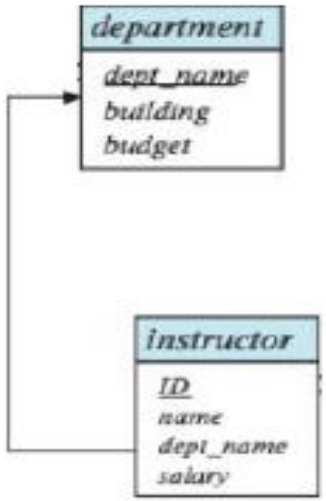
- Types of Integrity Constraints in DBMS
  - Entity Integrity Constraints
  - Referential Integrity Constraints
  - Domain Constraints

Integrity Constraints in Create Table

- primary key (A_1, ..., A_n)
  - The attributes required to be nonnull and unique
- foreign key (A_m, ..., A_n) references s
  - The values of the attributes (A_m, ..., 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

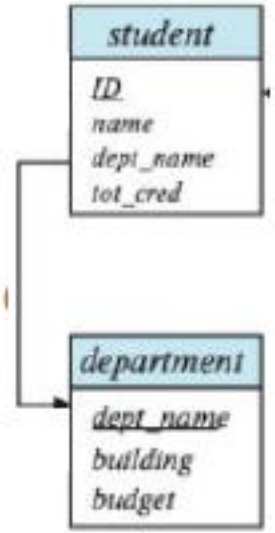
Create Table Example

- 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 student (
  ID varchar(5),
  name varchar(20) not null,
  dept_name varchar(20),
  tot_cred numeric(3,0),
  primary key (ID),
  foreign key (dept_name) references department);



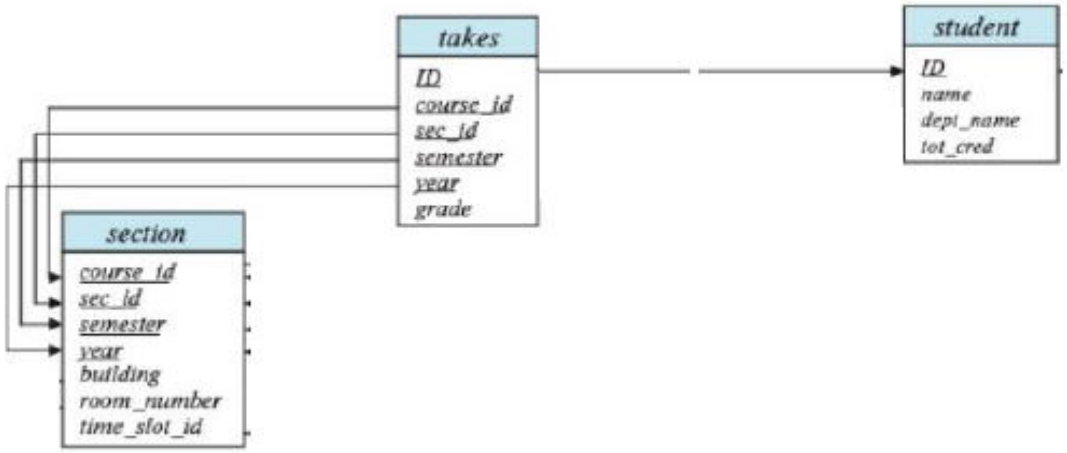
Create Table Example

- **create table** takes (

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

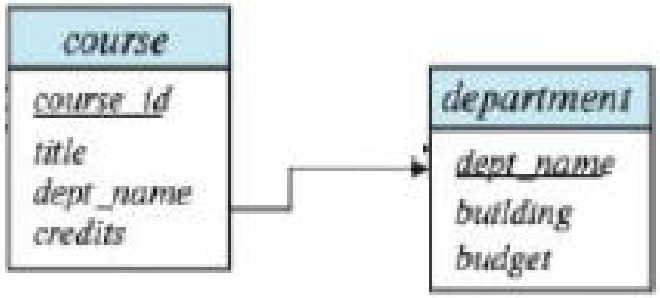
  primary key (ID, course_id, sec_id, semester, year) ,

  foreign key (ID) references student,
  foreign key (course_id, sec_id, semester, year) references section);



Create Table Example

- **create table** *course* (
  course_id **varchar**(8),
  title **varchar(**50),
  dept_name **varchar**(20),
  credits **numeric**(2,0),
  primary key *(course_id),*
  foreign key *(dept_name)* **references** *department*);



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;
  - 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 | 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

- `DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';`

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
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

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

```
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

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

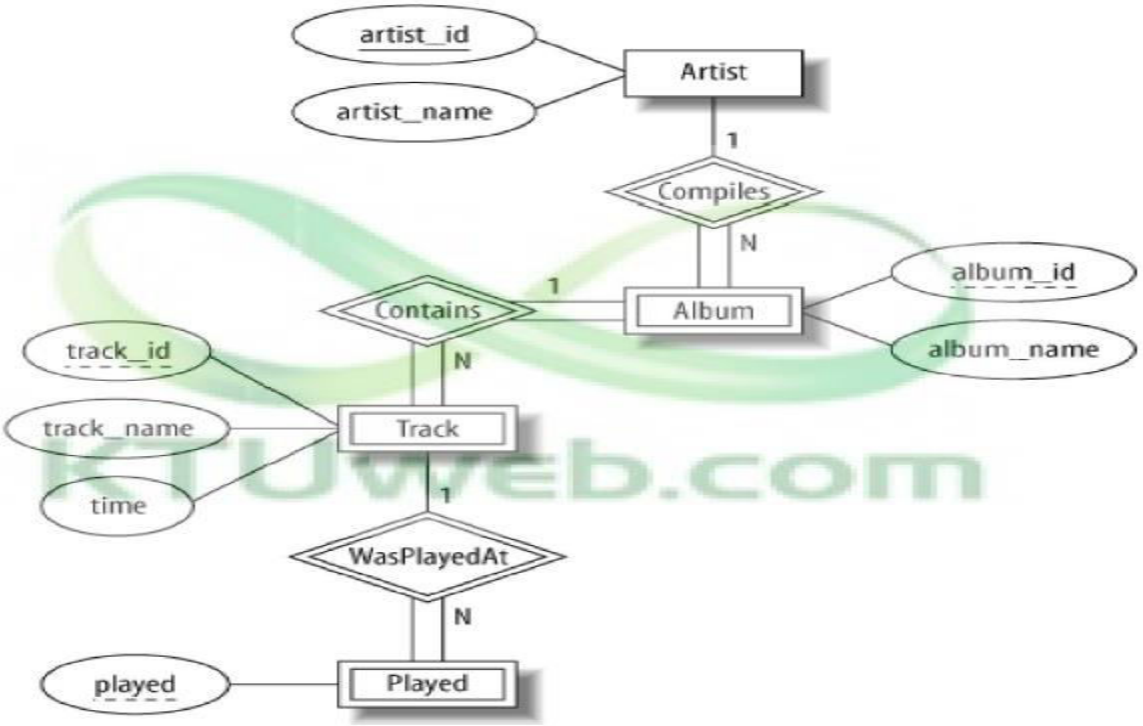
```
Delete from student where rollno='2';
```

6. Delete all records from table student

```
Delete from student;
```

University Question

Use the standard synthesis procedure to generate the set of relation corresponding to the ER diagram below. Identify primary and foreign key of the generated relations.

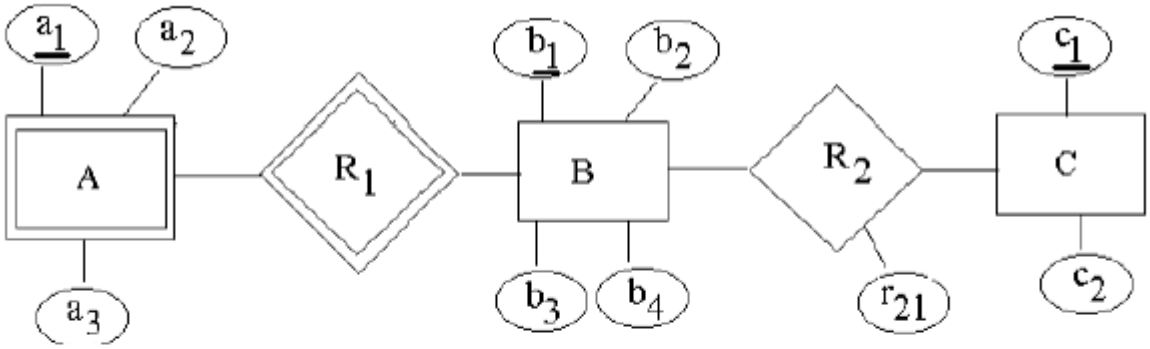


2. What is meant by referential integrity? How is it implemented using foreign key? Illustrate using a real example ?

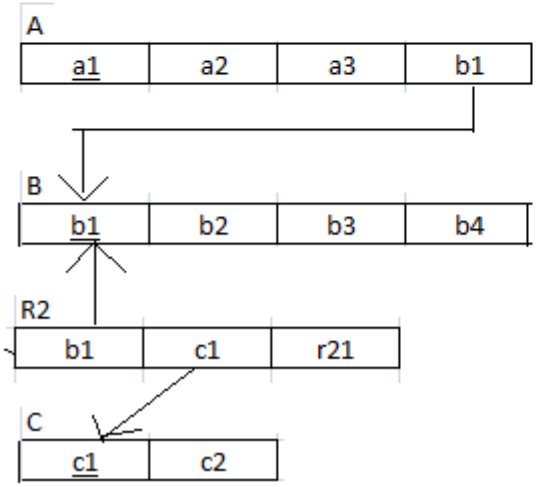
Hint : Refer the topic Referential Integrity

- 3.

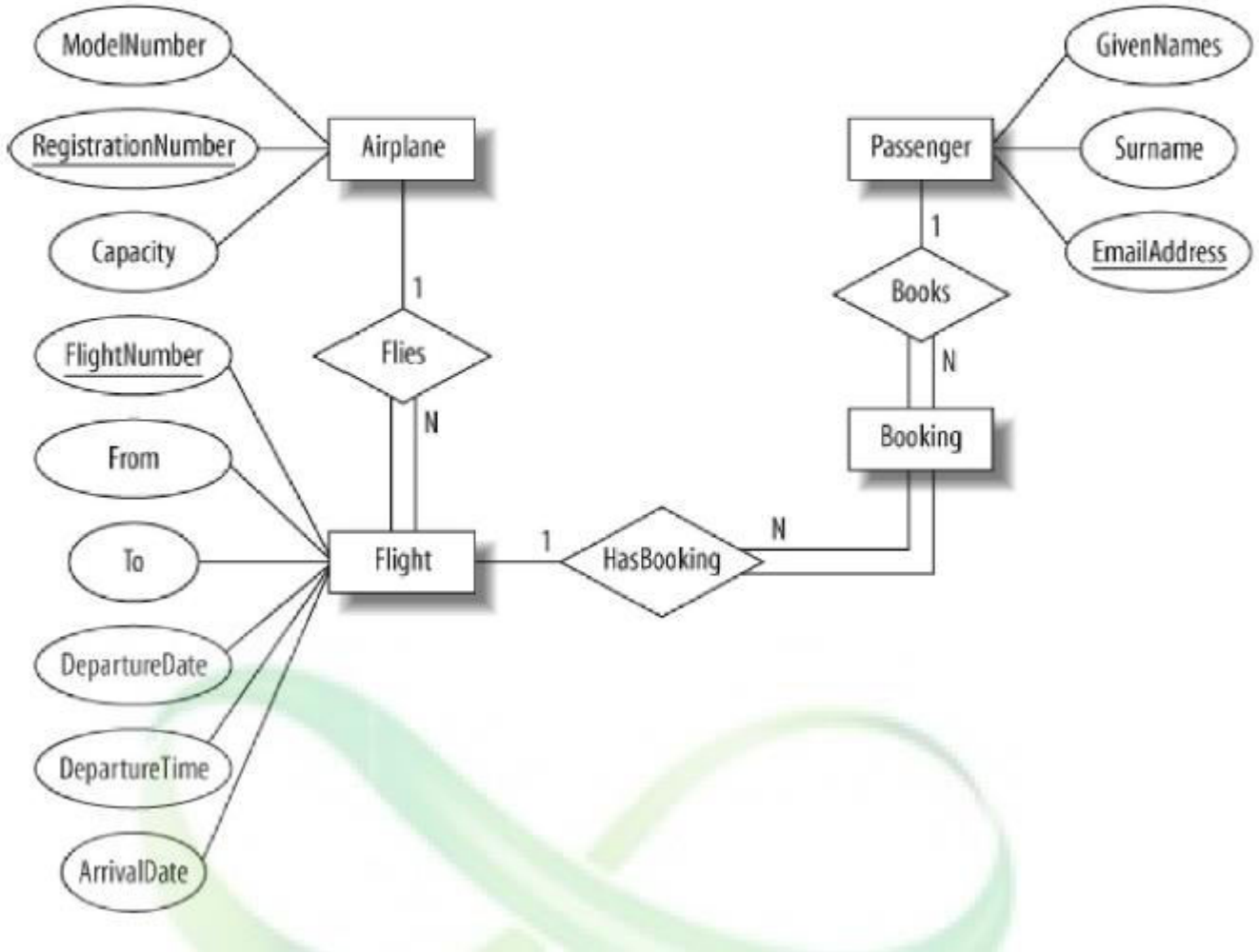
Consider the following ER diagram. Using this ER diagram create a relational database (primary keys are underlined).



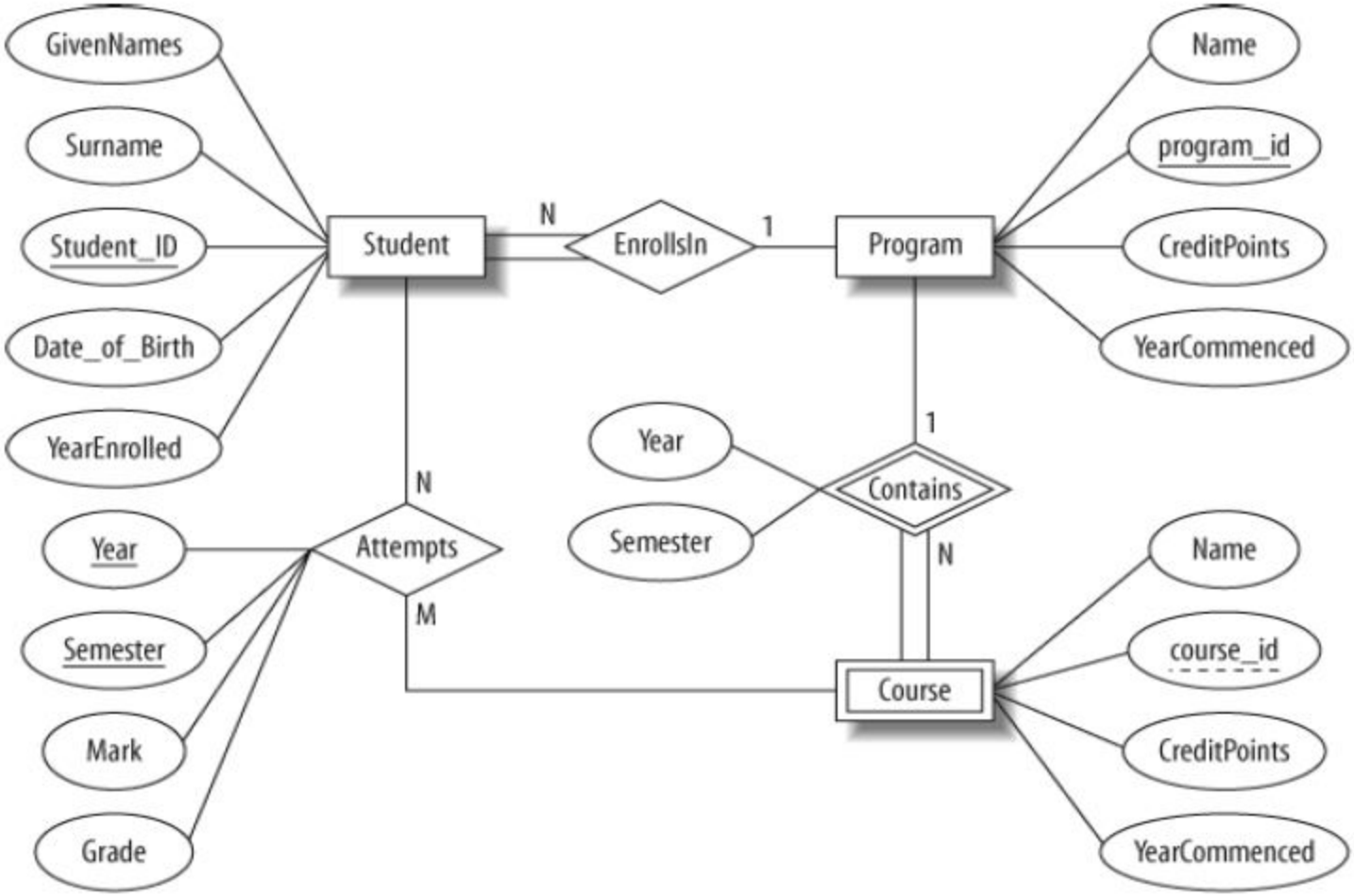
Soln:



a) Use the standard synthesis procedure to generate the set of relations corresponding to the ER diagram below. Identify primary and foreign keys of the relations



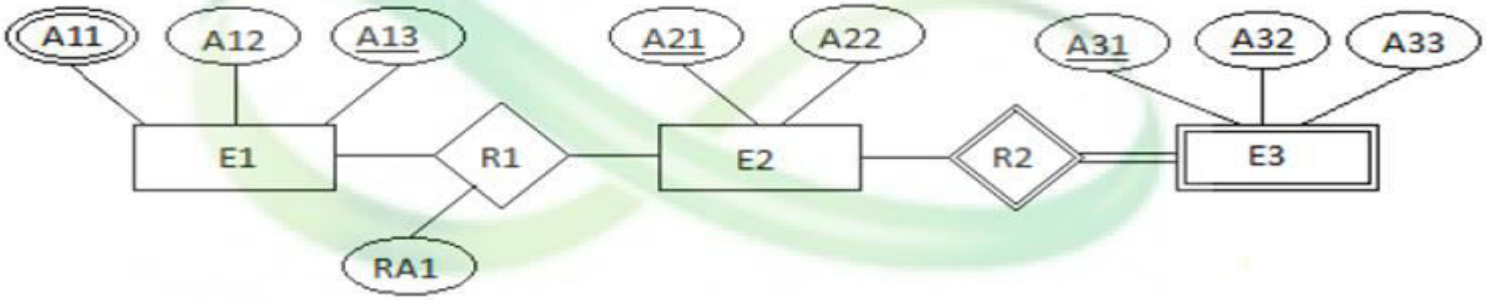
Covert the following ER Model to Relational Model



- What is entity integrity constraint? Why is it important?

Hint : Refer entity Integrity constraint

Using the following ER diagram, create a relation database. Give your assumptions.



Soln: