~~Unit → 1~~

Database Management system

Unit → 1

→ Introduction to Database

:→ Database is integrated collection of related information along with the details so that it is available to the several user for the diffⁿ application.

Database Name

student

Entity

column or field or attribute

Table Name : Computer science.

Roll number	Name	Address	Fees
1	Rahul	Delhi	10000
2	Raj	Mumbai	5000
3	Rohit	Kalkata	15000

Row or Record or Tuple

Note :-

- Database is in the form of table.
- One database may have one table or more than one table.
- One database may have one or more than one entity (Table Name)
- Attributes :→ Features of entity
  describe the

DBMS (Database Management System)

- ~~Database~~ is the software system that allows the access to the data~~base~~ of the database.

Database name

student → Entity

Table Name : Computer Science

attributes

Roll No.	Name	Address	Fees
1	Rahul	Delhi	10000
2	Raj	Mumbai	5000

Database

DBMS

Users

- Rahul Raj Rohit Scholarship.

File based System.

→ In file based system, we have to open indivisual file for Rahul Raj Rohit and scholarship. Then, we can check their details.

Roll no.	Name	Address	Fees	Scholarship
1	Rahul	Delhi	10000	5000
2	Raj	Mumbai	5000	3000
3	Rohit	Kolkata	15000	8000

Database Management System

DBMS

:- Database Management system.

- Database requires data and information.
- Data :-
  - It can be anything like name, place or number etc.
  - It refers to raw data or unprocessed data
- Information :-
  - It is organized or classified data so that it gives some meaningful values
  - It is a collection of data

Data	Information
• Data is raw facts and figures.	• Information is a processed form of data.
• Data doesnot help in decision making.	• Information helps in decision making.

* Eg :- University Database

:- Data about students, faculty, courses, research laboratories, registration etc.

It helps to keep an accurate track of the acddemics activities of the university

Atomicity → Either it is complete done or not done at all

Note:-
Before DBMS, organizations usually stored information in file based system but File based system has many disadvantages:-

- Difficulty in accessing data-

⇒ [Raj] [Rahul] [Rohit]

We have to access them indivisually and open the file for each of them (Raj, Rahul, Rohit)

If I choose file related to Rahul then it can only access details of Rahul only.

⇒ But if we use DBMS then we can access all the details of Raj, Rahul and Rohit at same time.

- Security problems → Hard to provide user access to some, but not all data
- Data inconsistent

→ If the file of a student modify by two different users then there may be inconsistency of their data.

- Anyone update only 10 lines and other may update 20 lines

- Data isolation

→ Each students has separate file. We keep all the details of students in separate file.

anything wh... either done completely or even not start at all.

and create inconsistent state

- Atomicity problem
  :→ The information which is entered to any system may fail at any time.
  → Failure occurs like ATM withdrawal.
  ~~and create inconsistency state~~
- Difficulty in concurrency control.
  :→ simultaneous execution of many at a time.
- Ingrity problem → Hard to add new constraint
- Data Redundancy or Duplicate of data
  :→ the same information may be duplicated in different files.
  - Multiple file formats.
- Difficulty in accessing data.
  - Need to write a new program to carry out each new task.

* Database Management System

:→ It is the software system that allows users to define, create and maintain a database and provides controlled access to the data.

- It is a collection of programs that enables user to store, modify and extract information from a database.

Features

- To add new information.
- To view or retrieve the stored information
- To modify or edit the existing data.
- To remove or delete the information.

Page No.
Date / /

syntax:-

```
CREATE TABLE table_name
(
  column_name1 data_type(size),
  column_name2 data_type(size),
  column_name3 data_type(size)
  );
```

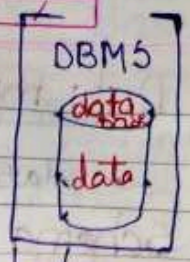
Eg :-

```
CREATE TABLE my_tab
(
  name varchar(30),
  roll number(4),
  address varchar(100)
  );
```

output:-

Name	Roll	Address

components of Database



5 major components :-

:- Hardware → It is the actual computer system used for keeping and accessing the database. It consists of secondary storage devices as hard disks.

:- Software → It is actual DBMS.

:- Data → It is the bridge b/w machine components and user components.

:- Users → There are number of users who can access or retrieve data on demand using the applications and the interfaces provided by DBMS.

:- Procedures → It refers to the instructions and the rules that govern the design and the use of database.

Applications of DBMS

- Airlines or railway :- Reservations, schedules, search flight/train between two particular station.
- Universities :- Registration, Grades, Library books due
- Sales :- customers, products, purchases, online tracking.
- Human resources :- employee record, salaries, tax.

Data model :- A set of concepts to describe the structure of a database.
- data
- data relationships
- data constraints.

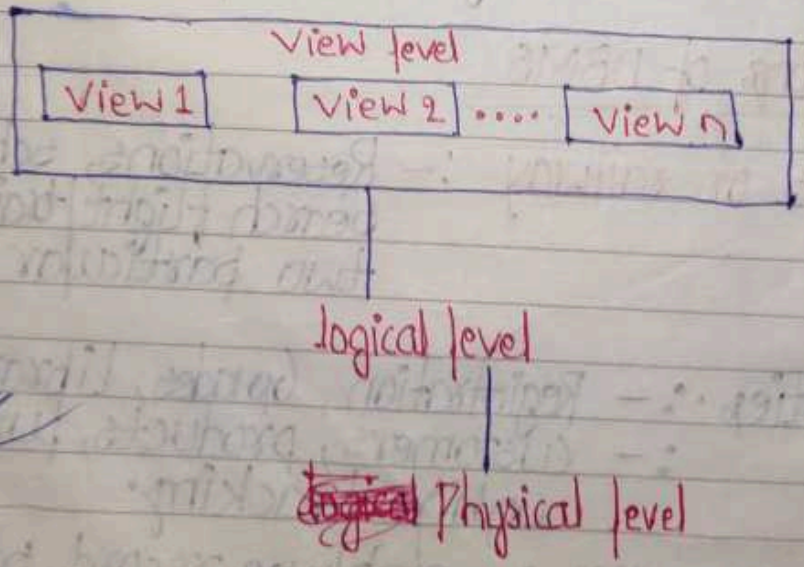
Schema :- The overall description of the database

Instance :- Data in the database at the particular instant of time.

Physical Data Independency
The ability to modify the physical schema without changing the logical schema

* Level of Abstraction

- Physical level :- It describes how a record is stored
- Logical level :- It describes what data stored in database and relationship among data.
- View level :- It describes only part of database. Application programs hide details of data types. View can also hide information for security purposes.



- Instances → At a particular time, what I do → the actual content of the database.
- schemas → plan to create or work on a database.

Logical schemas → structure of logical level

Database Languages

physical schema → structure of physical level

:→ Database languages are used to create and maintain database on computer.

(DDL)

* Data Definition Language :- It is a language that allows user to define data and their relationship to other types of data.

- CREATE
- ALTER
- DROP
- TRUNCATE
- RENAME

Physical data independence: the ability to modify the physical schema without changing the logical schema

- Specification notation for defining database schema

(DML)

* Data Manipulation Language :- It provides a set of operations to support the basic data manipulation operations.
It allows user to insert, update, delete and retrieve data from the database.

- DELETE — INSERT — SELECT — UPDATE

- It also known query language.

(DCL)

* Data Control Language :- DCL statements control access to data and the database.

- GRANT
- REVOKE
- COMMENT

- Transaction Control Language (TCL)
  - Commit

Entity Relationship

Page No. KRISHNA
Date / /

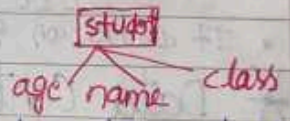
* ER MODEL → It is a visual representation of diffn data and their relations b/w them

:→ The ER model defines the conceptual view of a database.

→ **Entity** : It can be a real-world object, either animate or inanimate, that can be identifiable. or it has independent existence

eg In school database;
students, teachers, classes and course offered can be considered as entities.
- All these entities have some attributes or properties that give their identity.

⇒ **Attributes** : Entities are represented by means of their properties.
or It is a property of an entity, relationship.
eg In school database;
students entity may have name, class and age as attributes.



⇒ **Entity type** : It is collection of entity which has common attributes.

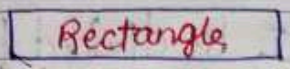
student → entity type or schema

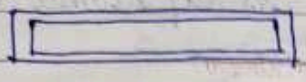
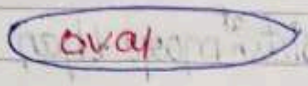
ID	Name	Age
e1	Ram	14
e2	Shyam	12
⋮	⋮	⋮
en	John	13

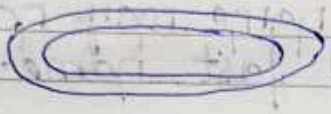
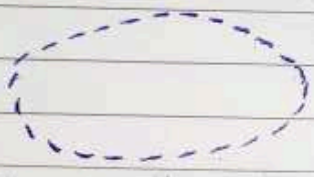
→ Attributes

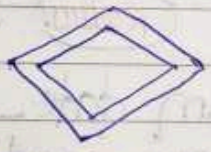
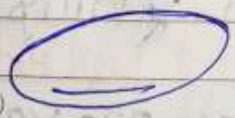
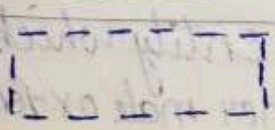
entity

Basic Terminology of ER MODEL or Element

- 

An object is in real world
Entity or strong Entity.
- 
Weak Entity.
- 

Attribute.
(Which describe entity)
- 
Multivalued Attribute.
-
Composite Attribute.
- 
Derived Attribute.
-

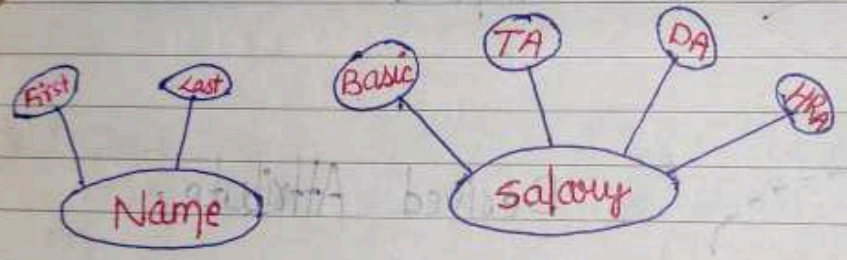
Relationship.
(Association among the entities)
- 
Weak Relationship.
- 
Key attribute (Primary key)
-
Partial key.
- 
Aggregation box

* Types of Attributes

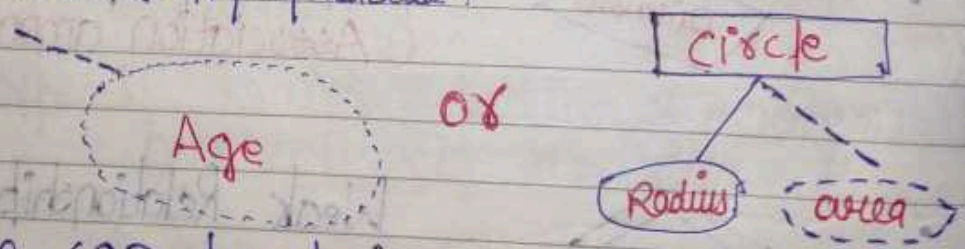
- **Simple attribute :-** It cannot be divided further.
  eg:- student's phone number is an atomic value of 10 digits. (mobile No)

- **Composite attribute :-** It consists of more than one simple attribute.
  eg :- student's complete name may have first_name and last_name.

  eg :- salary includes basic, TA, DA, HRA.



- **Derived attribute :-** The value of an attribute can be derived from other attributes present in database.

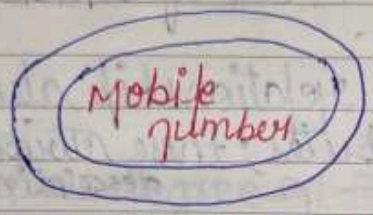


Note :- Age can be derived from date_of_birth
area can be derived from radius.

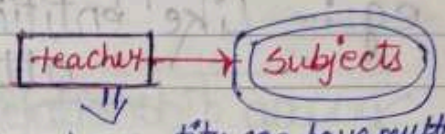
- **Single-value attribute :-** It takes one value per an entity.
  eg :- (Gender)
  Gender is an entity which can hold either male or female.

- **Multivalued attribute** :- It may takes more than one values.

eg:- A person may have more than one phone number or email_address.



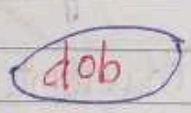
or



teacher entity can have multiple subject values.

- **Stored attribute** :- It doesnot take any updation in database.

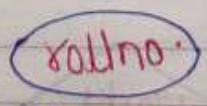
eg:- Date_of_birth doesn't take any updation in future.



- **Key** :- It is an attribute or collection of attributes that uniquely identify an entity among entity set

or which uniquely identify in the entity set.

eg:- roll_number of student makes him/her identifiable among students.



- **Super key** :- A set of attributes (one or more) that collectively identifies an entity.

→ It describes how entities interact.

- **Relationship** :- Association among the entities ~~or specifies~~

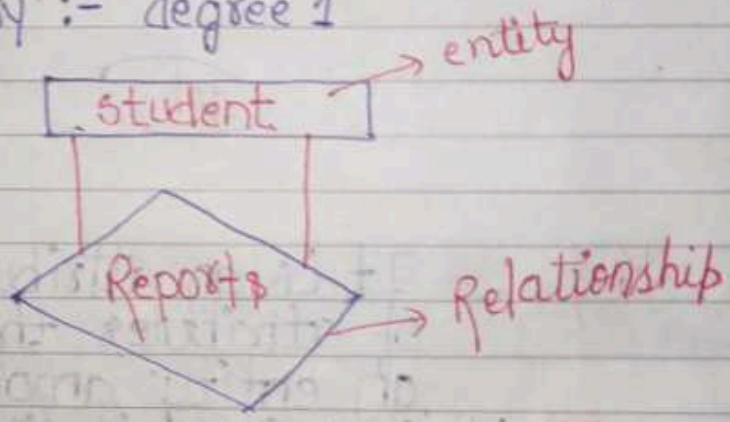
eg :- student enrolls in a course, where Enrolls called relationship.

- **Relationship set** :- A set of relationships of similar type.

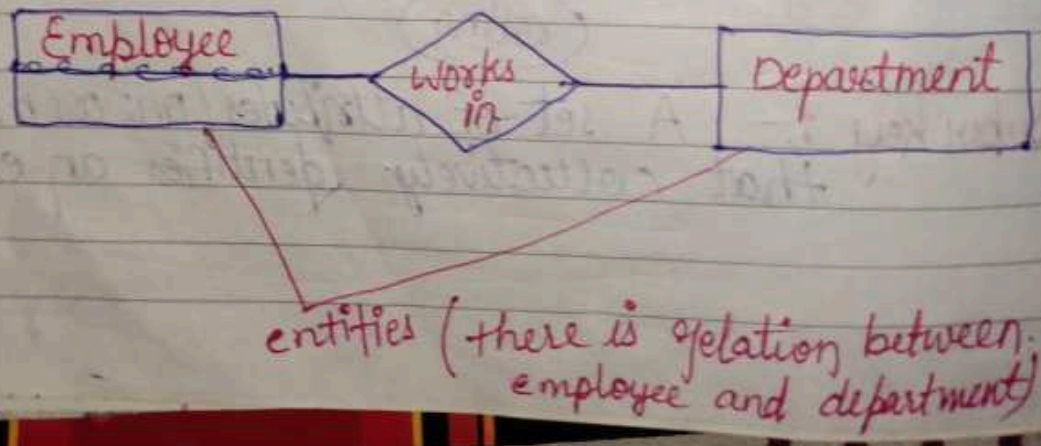
eg :- Like entities, relationship also have an attributes; these attributes called descriptive attributes.

- **Degree of relationship** :- specifies the no. of entity sets participates in relation ship set.

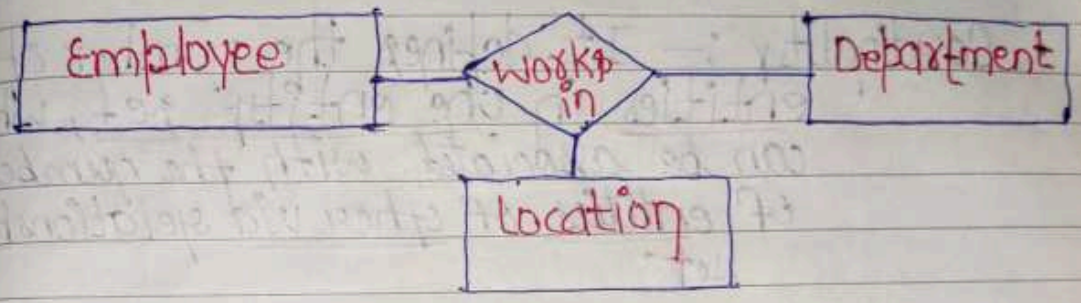
i) Unary :- degree 1



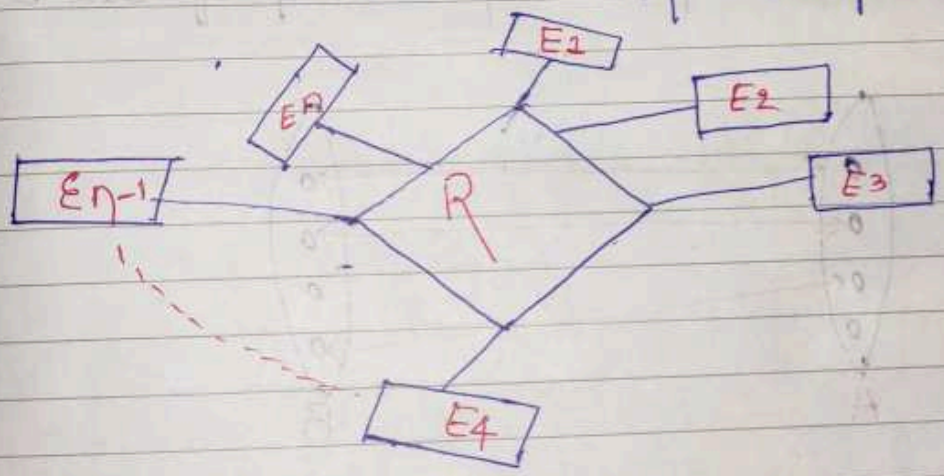
ii) Binary :- degree 2 → Relationship between two entity sets.



iii) Terniary :- degree 3 → Relationship among three entity sets.



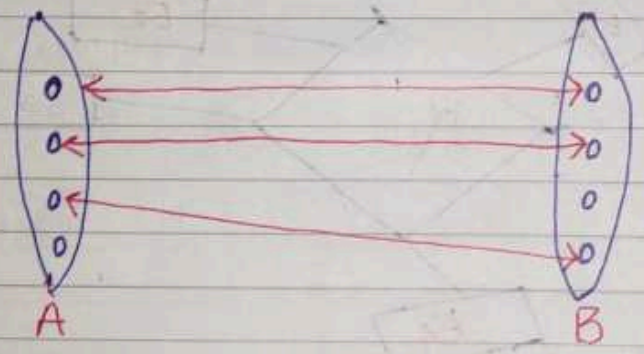
iv) n-ary :- degree n → Relationship among n entity sets.



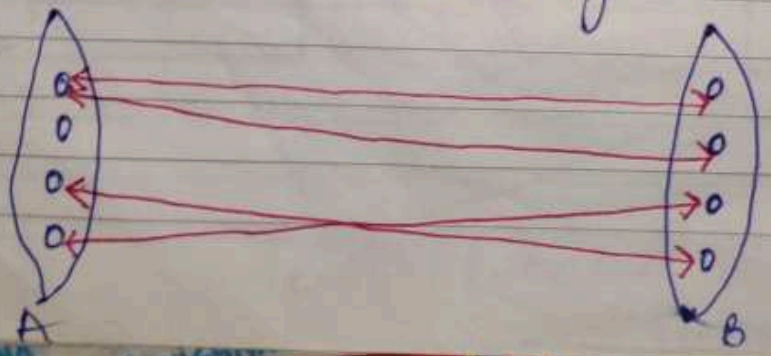
* Mapping

- Cardinality :- It defines the number of entities in one entity set, which can be associated with the number of entities of other via relationship set.

- One-to-One :- One entity from entity set A can be associated with at most one entity of set B & vice versa



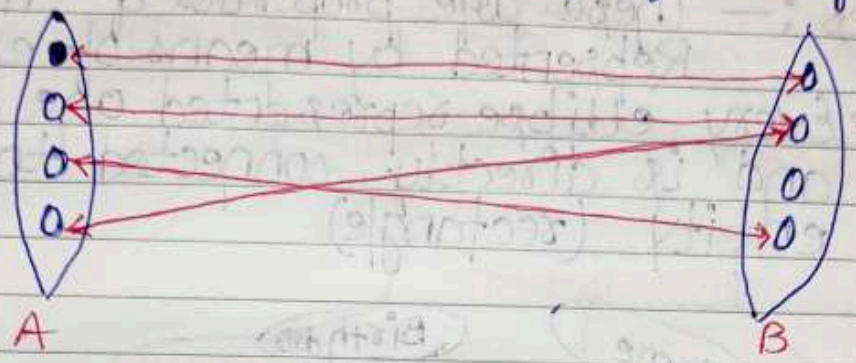
- One-to-many :- One entity from entity set A can be associated with more than ~~more~~ one entities of entity set B however an entity from set B, can be associated with at most one entity.



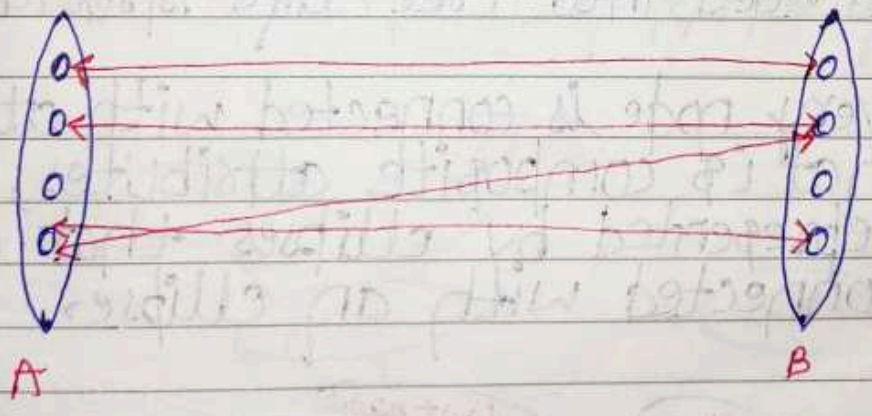
eg:- Email Account —— has —— Contacts

In email system, one account may have multiple contacts.

- Many-to-one :- More than one entity from set A can be associated with at most one entity of entity set B.



- Many-to-Many :- ~~more~~ One entity from A can be associated with ~~one~~ more than one entity from B and vice-versa.



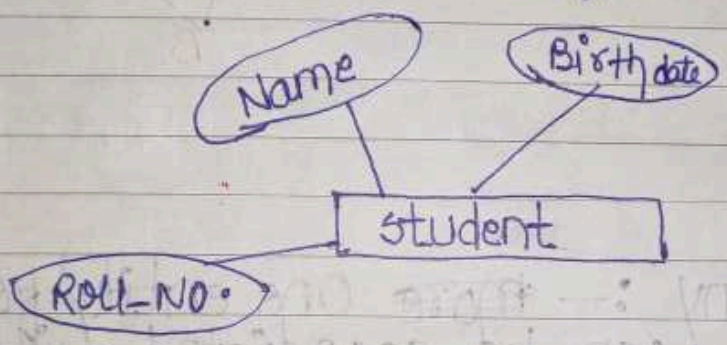
ER Representation

Page No. KRISHNA
Date / /

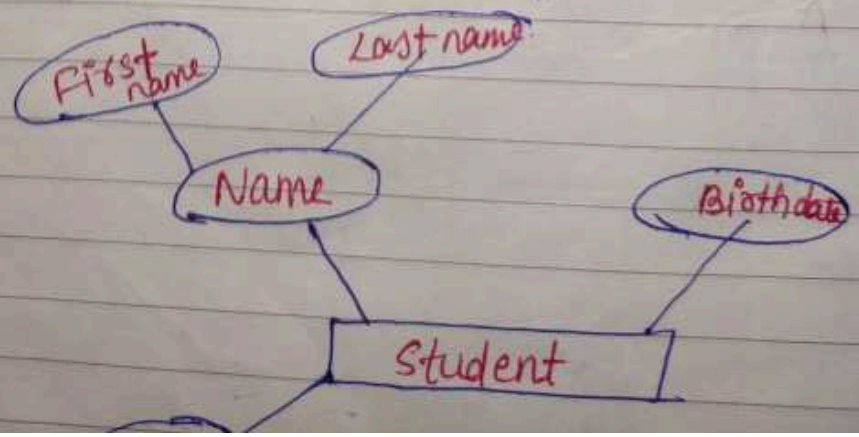
- **Entity :-** Represented by means of rectangles.

[student] [Teacher] [Projects]

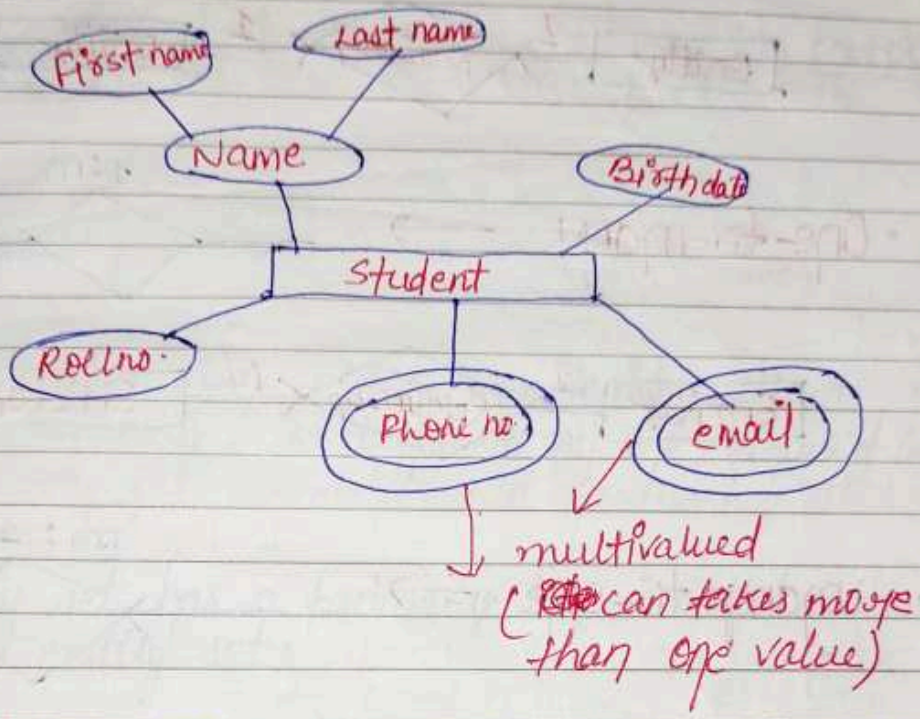
- **Attributes :-** These are properties of entity. Represented by means of ellipses
  - ⇒ Every ellipse represented one attribute and is directly connected to its entity (rectangle)



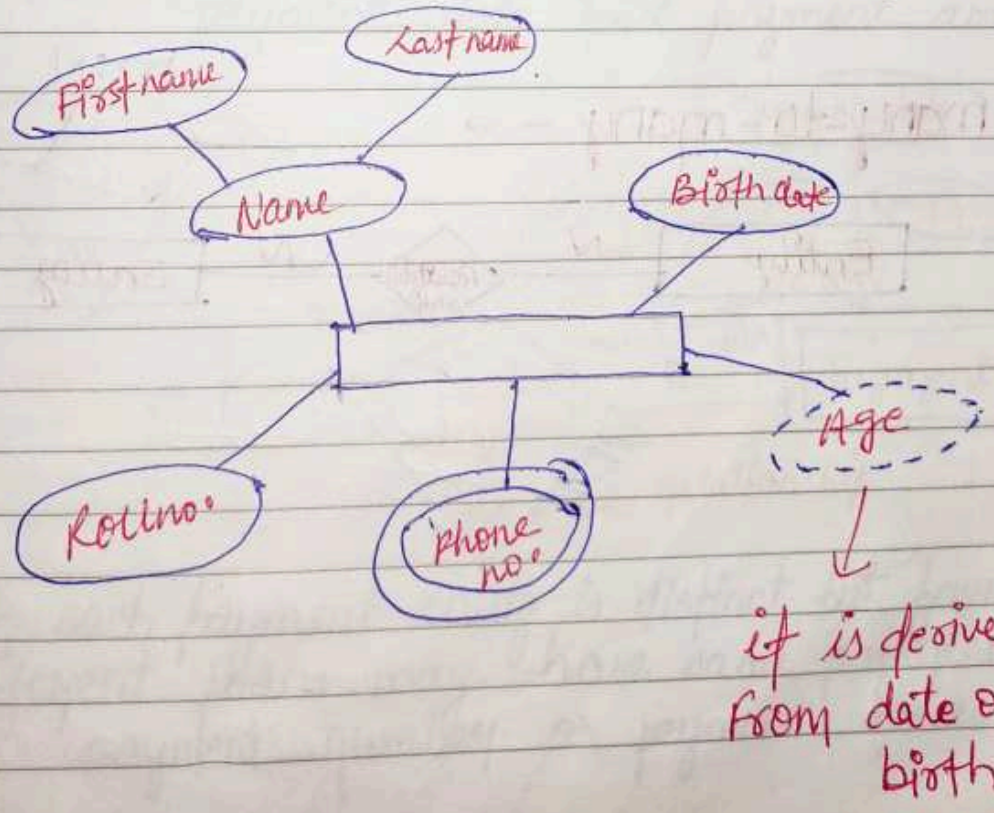
- **Composite attributes :-** They are further divided into tree like structure.
  - ⇒ Every node is connected with attribute. That is composite attributes, are represented by ellipses that are connected with an ellipse.



• Multivalued :— These attributes are depicted by double ellipse.



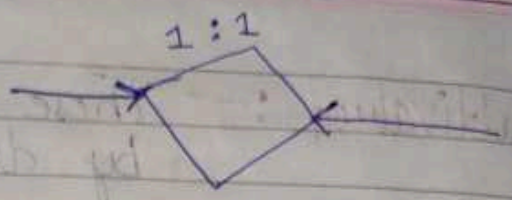
• Derived :— These can depicted by dashed ellipse.

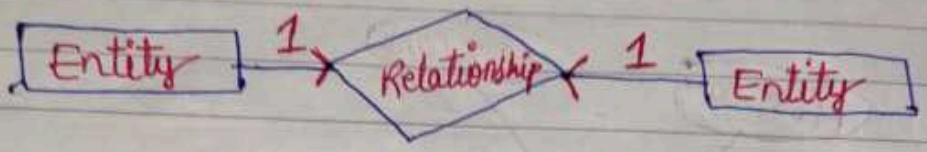


ER Representation of Relationship

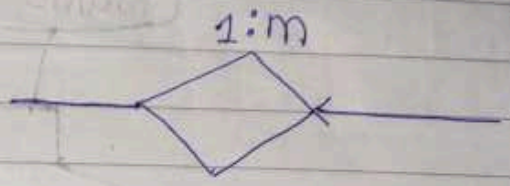
Page No. KRISHNA
Date / /

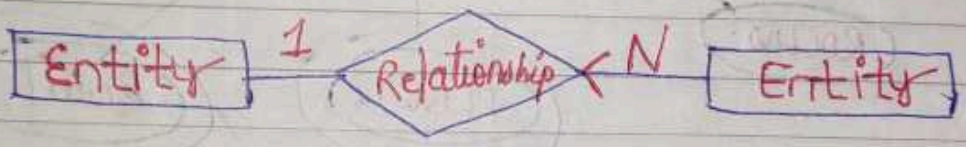
- One-to-one →



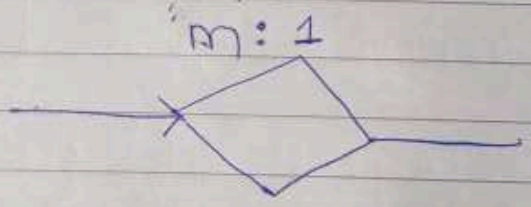


- One-to-many →

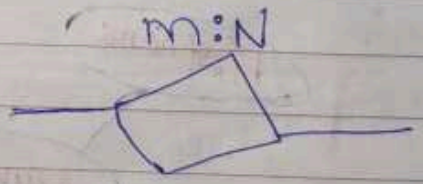


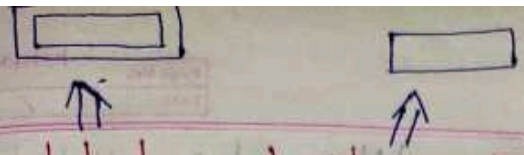


- many-to-one →



- many-to-many →

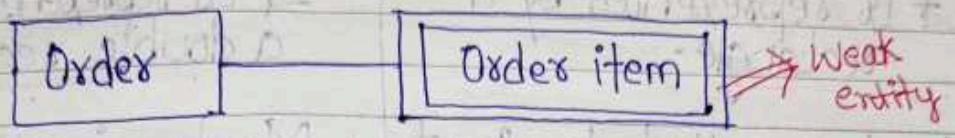




Weak and Strong Entity

- Weak entity is an entity that depends on the existence of other entity.

Or, we can say that it is an entity which cannot be identified by its own attributes.

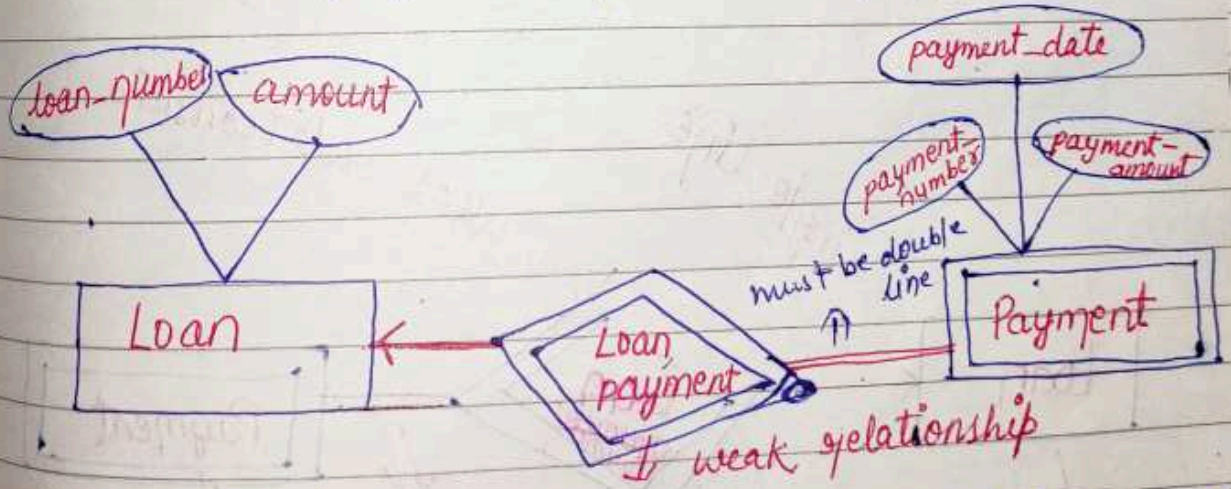


Here, the order item will be meaningless without an order so it depends on the existence of Order

⇒ Weak entity set which doesnot have a key attribute

- An entity set has a primary key is called as Strong entity set.

⇒ Consider an entity set payment Payment which has three attributes :- payment_number, payment_date and payment_amount

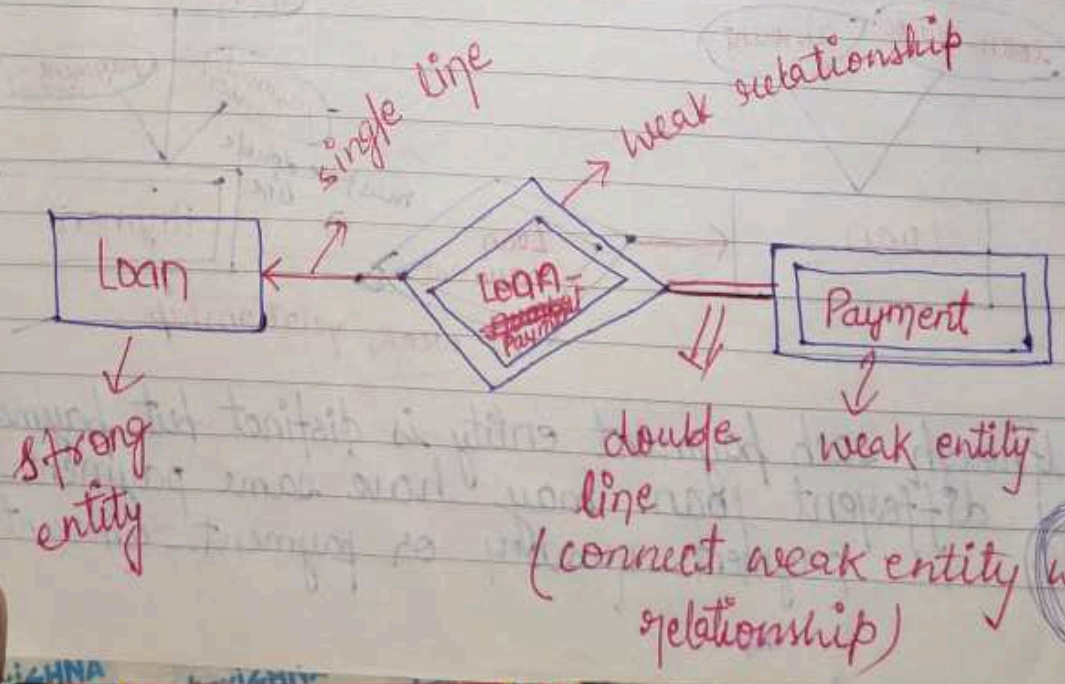


Although each payment entity is distinct but payment for different loans may have same payment_date payment_number or payment_amount.

Note :-

- A member of strong entity set is called dominant entity and member of weak entity set is called subordinate entity.

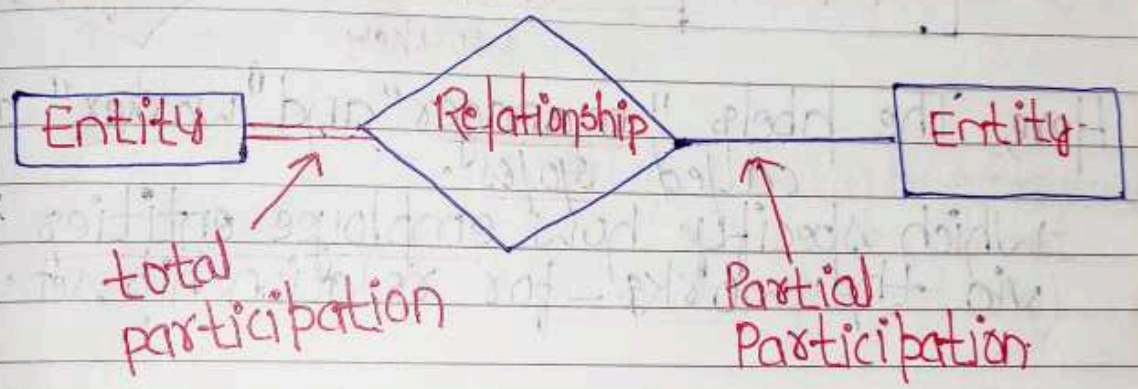
~~Wea~~ Strong entity set	Weak entity set
• It is represented by a rectangle.	• It is represented by a double rectange.
• Member called dominant.	• Member is called subordinate.
• Relationship b/w two strong entity is represented by diamond symbol.	• Relationship b/w one strong and one weak set is represented by double diamond symbol.
• Line connecting strong entity with relationship is single.	• line connecting weak entity set with relationship is double.



Total and Partial Participation

- **Total Participation** :- Each entity is involved in the relationship.
  Total participation is represented by double line

- **Partial Participation** :- Not all entities are involved in the relationship.
  Partial participation is represented by single line

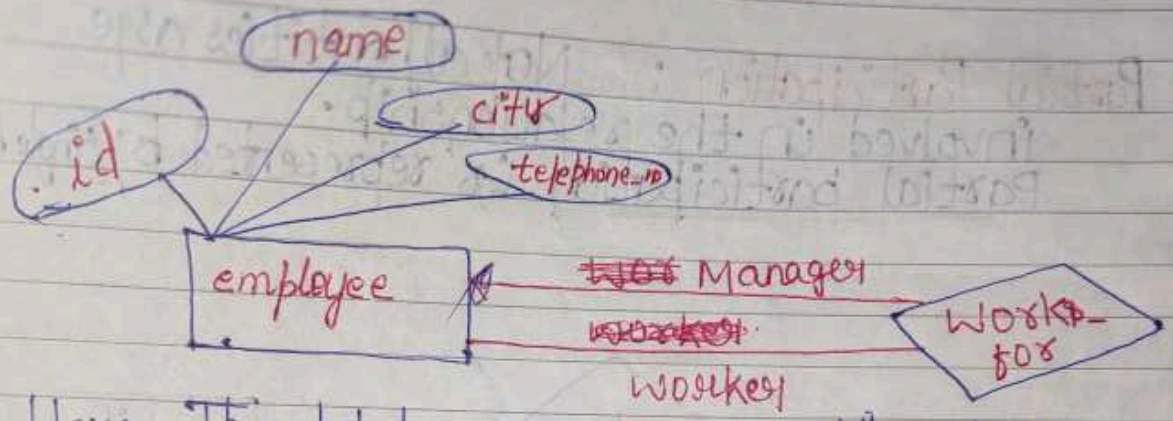


eg :- Employee head of department.
- Not all employees become a head, but department will always be headed by one employee.

so, employee participated partially in relationship

Roles

- Roles are ~~role~~ type which specify how employee entities interact in relationship set.



Here, The labels "manager" and "worker" are called roles.
Which specify how employee entities interact via the works-for relationship set.

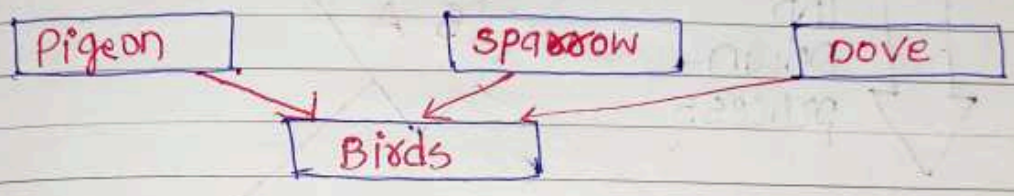
- Roles are indicated in E-R diagrams by labelling the lines that connect diamonds to rectangles.

Generalization

→ diamond.

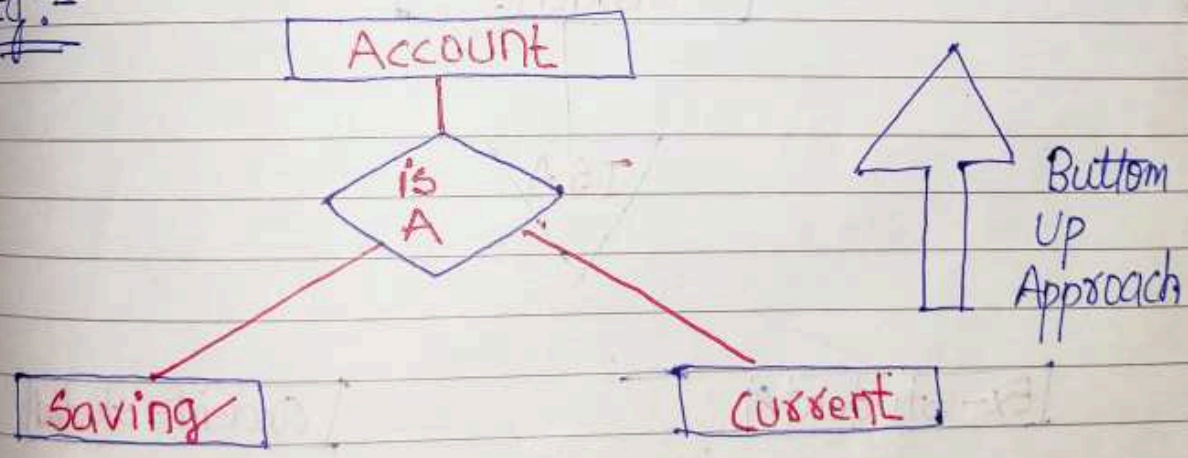
:→ The process of generalizing entities, where the generalized entities contain the properties of all the generalized entities.

- In this, a number of entities are brought together into one generalized entity based on similar characteristics.

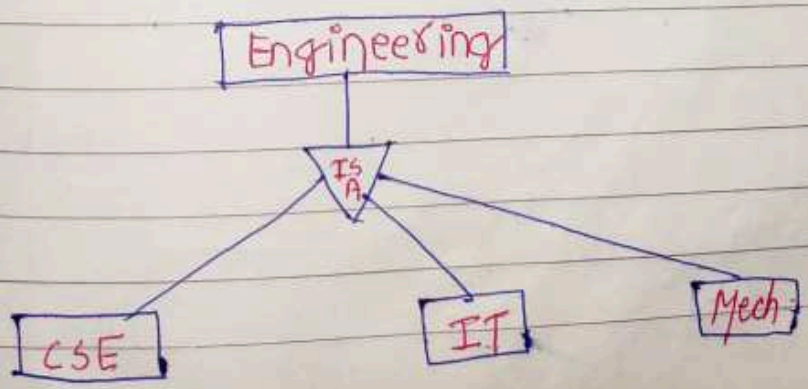


Here, Pigeon, sparrow, Dove can all be generalized as birds (all have almost same features).

Eg:-



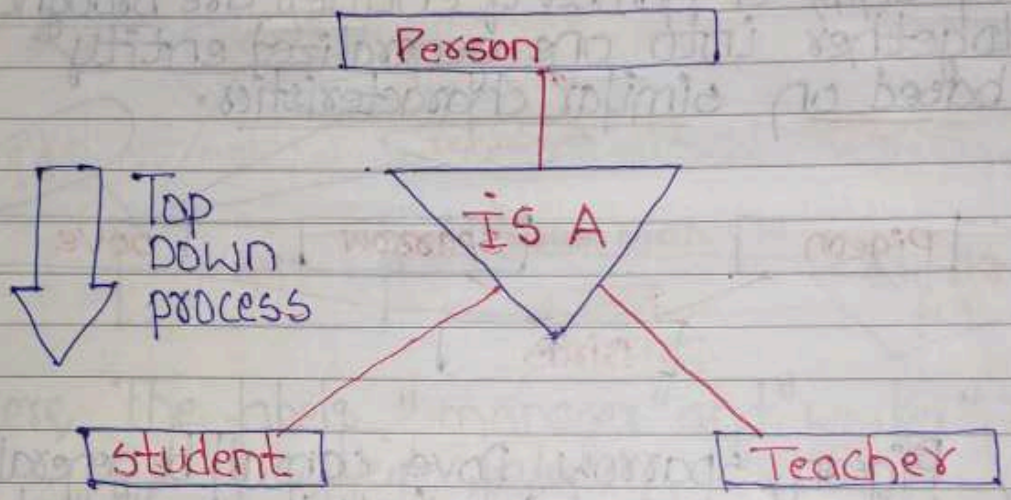
Eg:-



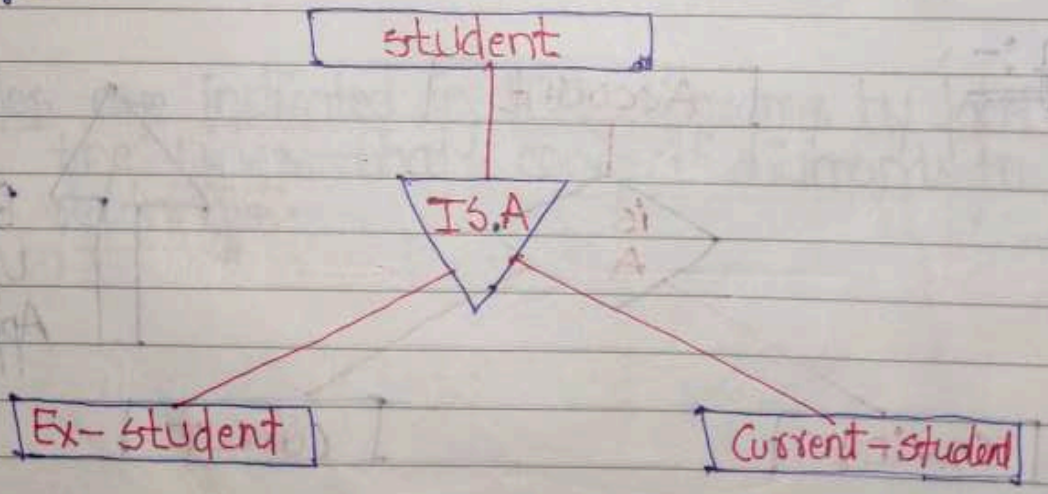
Specialization

- The group of entities is divided into sub-group based on their special characteristic.

eg:- In school, person may be divided as teacher or student.

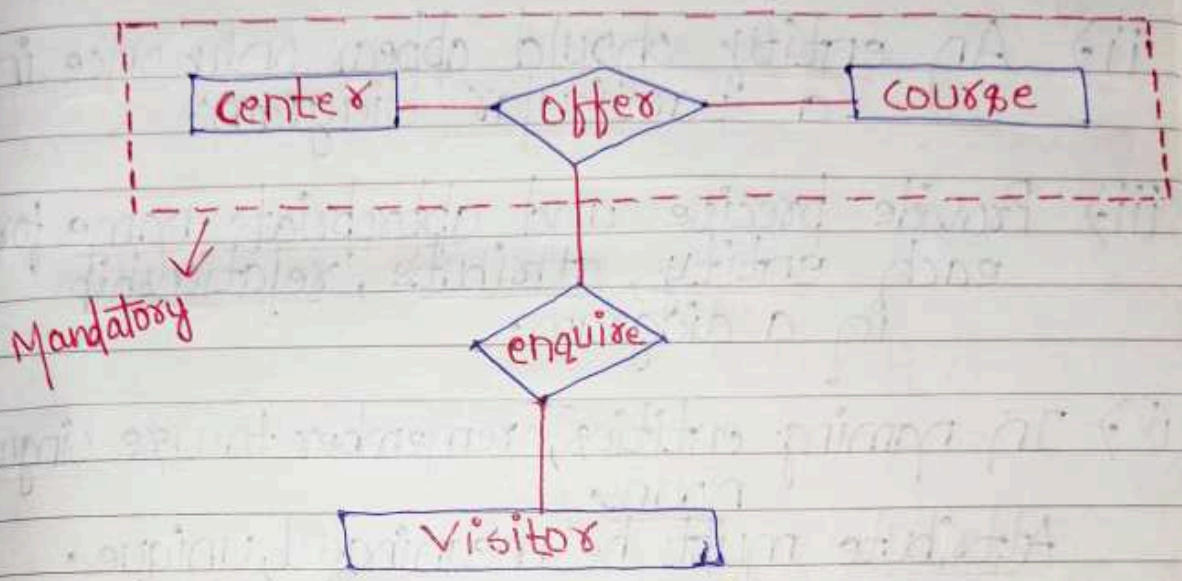


eg:-



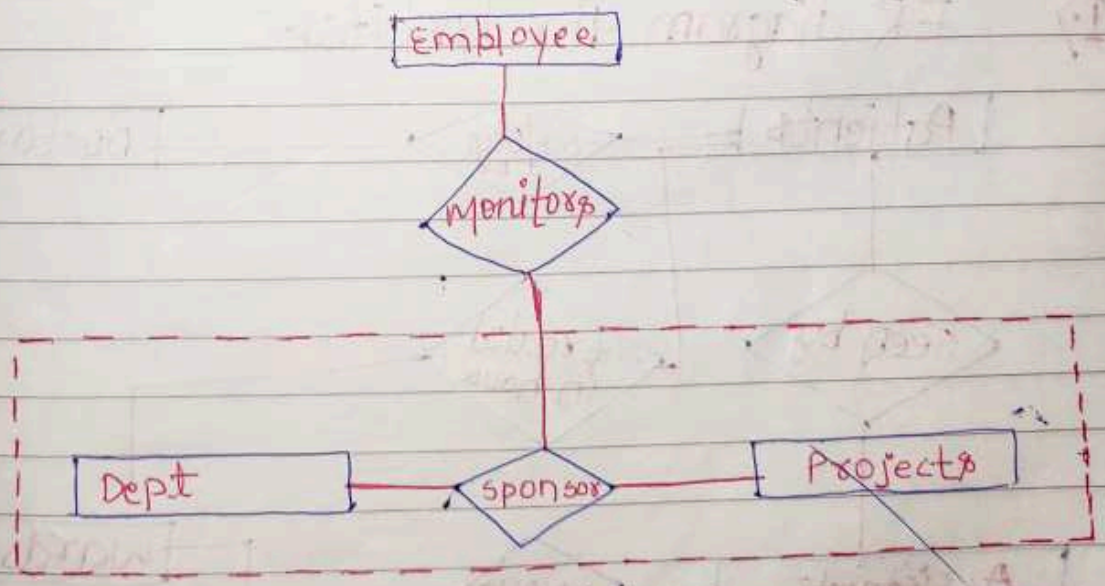
Aggregation

- A process when relation between two entity is treated as a single entity.



Here, the relation b/w center and course, is acting as an entity in relation with visitor.

eg:-



* Rules to draw ER diagram

i.) Identify all the relevant entities in a given system.

ii.) An entity should appear only once in a particular diagram.

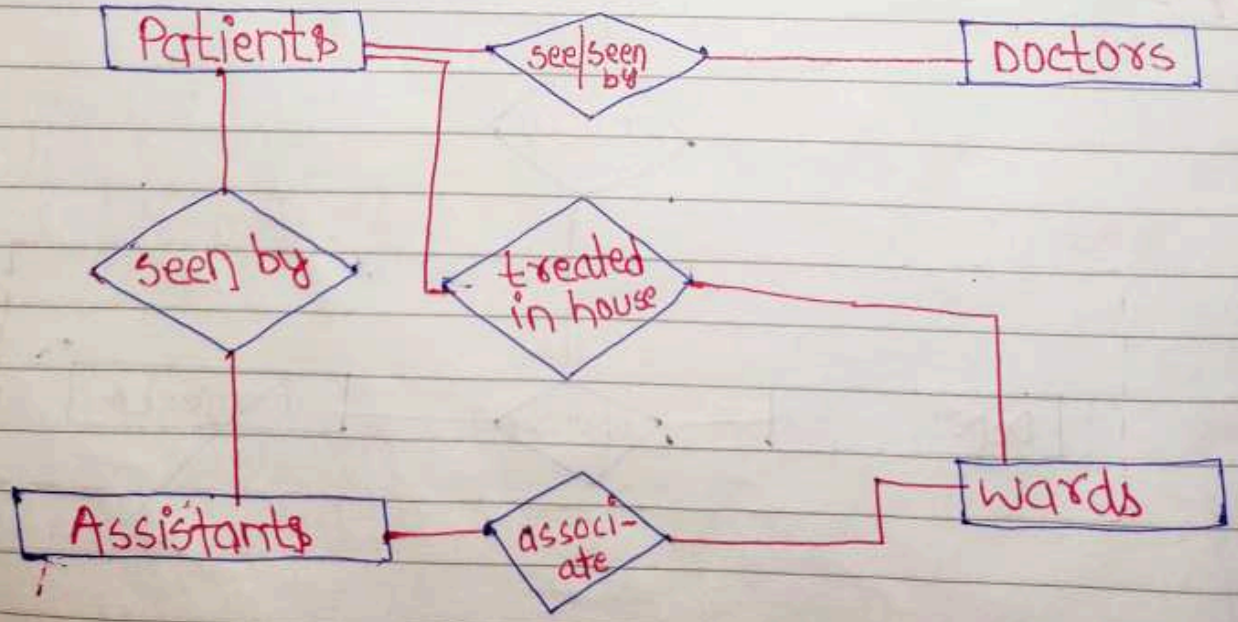
iii.) Provide precise and appropriate name for each entity, attribute, relationship in a diagram.

iv.) In naming entities, remember to use singular nouns.
Attribute must be meaningful, unique.

v.) Remove unnecessary relationships b/w entities.

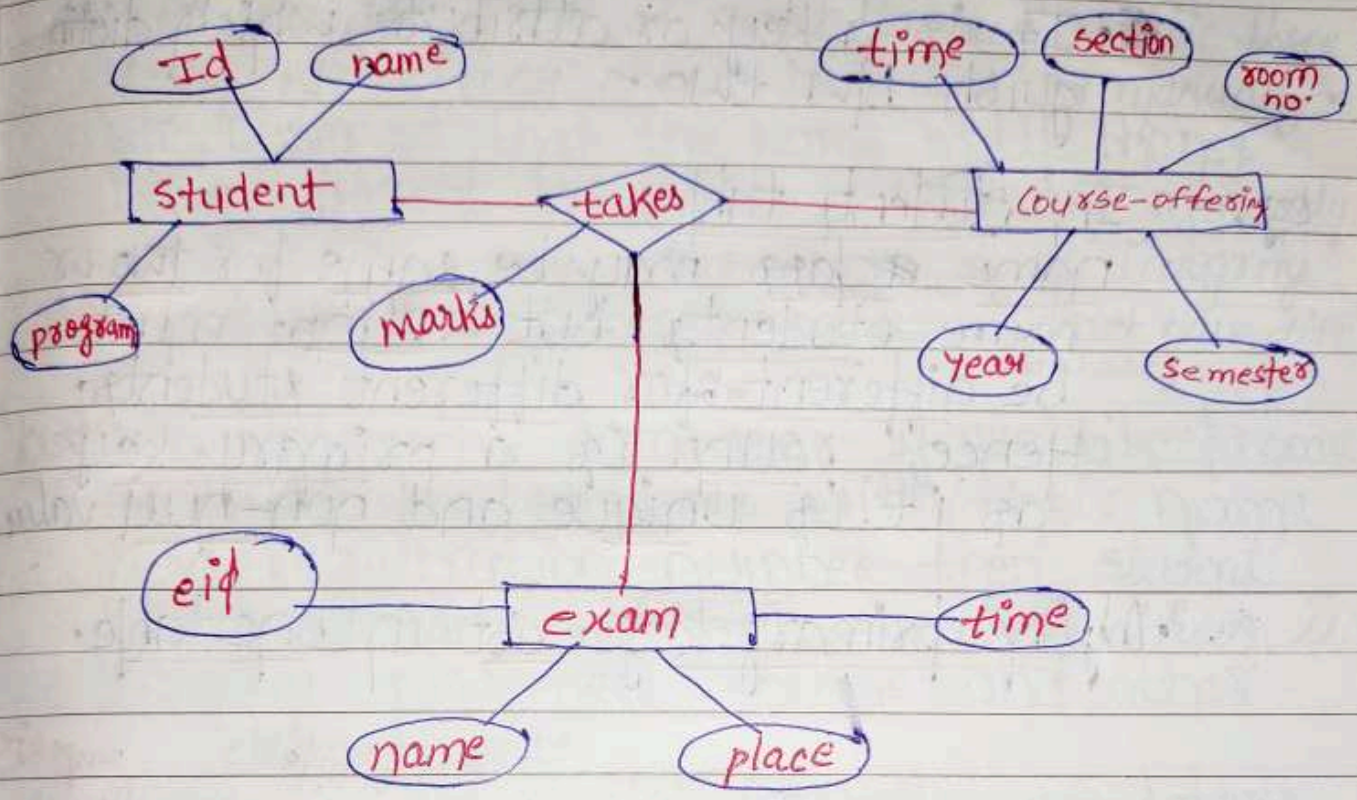
Examples

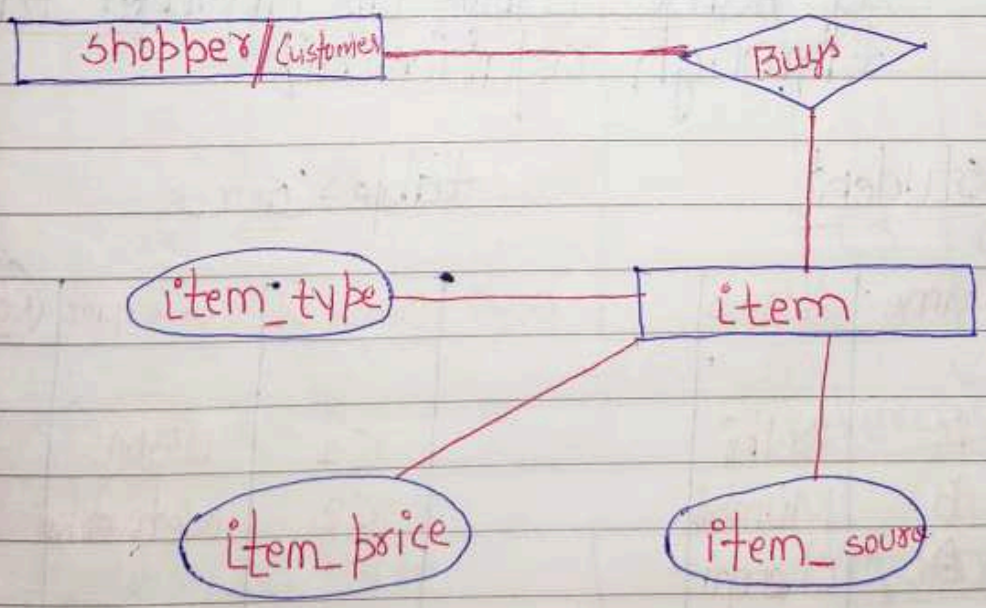
1.) ER diagram for hospital.



Page No.
Date / /

2. ER diagram for student database





Page No.
Date / /

• **Primary key** :→ It is unique as well as not null value.

- It is a key or attribute which distinguish b/w two.

eg:- In students table,
name, class may be same for two or more students but rollno. must be different or unique for different students.
Hence, rollno is a primary key as it is unique and non-null value

• Only One primary key exist in one table.

• **Foreign key** :→ It is the attribute (simple) or set of attribute (composite) that references to primary key of same table or another table through relationship.

• table→ student

Rollno (primary key)	Name	Address
1	A	Delhi
2	B	Mumbai
3	A	Chennai

Base table or referenced table
(Table which has primary key)

table→Course

Course id	Course name	Rollno (Foreign key)
C1	DBMS	1
C2	Networks	2

Referencing table
Table which has

→ same value

Page No.
Date / /

Referential Integrity

⇒ Let us suppose, If we want to buy mobile phone then we either check it online or manually. We observed that the same mobile phones have different prices in online and manually. so, we can say that there is no integrity here. (It is because amazon, flipkart have diff database)

But; In University database, If we like to search a particular students with their name and registration number then student will be same either we search them in library, placement, course and other department.

So, There is referential integrity mandatory in database.

primary key

Table : Student

eg:-

Rollno.	Name	Address
1	A	Delhi
2	B	Mumbai
3	A	chennai
4	D	Goa

Base table or referenced table.

Table : Course

foreign key

Course Id	Course Name	Rollno.
C1	DBMS	1
C2	Networks	2
C3	C++	7

Referencing table

- Referenced table

→ Insertion (No violence)
→ Deletion (may be violence)
→ Updation (may be violence)

Referencing table

→ Insertion → May cause Violence
→ Delete → Will not cause Violence
→ Updation → May cause Violence

Solution

on delete cascade

table then it also delete

• Candidate key :- It is a minimal set of attributes which uniquely identifies a tuple in a relation.

• Super key :- A super key is a combination of all possible attributes which can uniquely identify two tuples in a table.

• super set of any candidate key is a super key.

candidate key attributes

eg:-

• Candidate key
→ RollNo.

RollNo.	Name	Age

• Super key will be collection of tuples of one candidate key and any other attributes.

eg:- (RollNo.) → It is candidate key as well as super key.
(RollNo., name)
(RollNo., age)
(RollNo., name, age)

Super key (It contains Candidate key and other attributes)

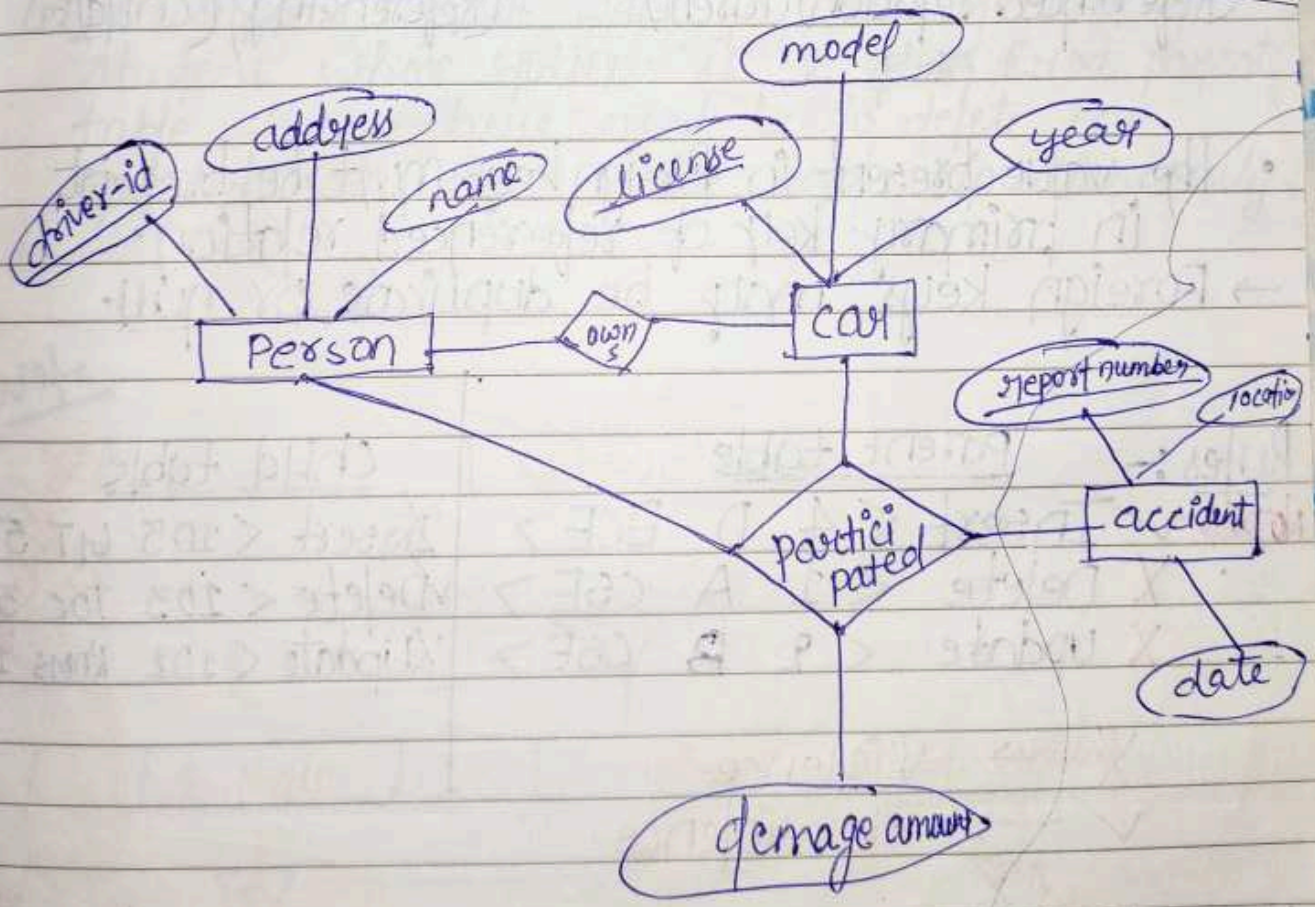
(name, age) → It is not a super key because it doesnot contain candidate key.

Note :- Every candidate key is called super key but every super key neednot be a candidate key.

Ques:- ~~R(A1, A2, A3,~~ $R(A_1, A_2, A_3, A_4, \ldots, A_n)$
then how many super keys are possible?

(I) If A_1 is candidate key.
(II) If A_1, A_2 are candidate keys.

sol:-



Concept on foreign key

:→ It is based on Referential Integrity Constraint

table :- Student

R.No. (Primary key)	Name	Branch
1	A	CSE
2	B	IT
3	C	CSE

(Referenced relation)(Parent)

table :- Registration

C.No	C.Name	R.No. (Foreign key)
101	DBMS	1
102	CD	1
103	TOC	3
104	PL	null

duplicate

(Referencing)(child).

- The value present in foreign key must be present in primary key of referenced relation.

→ Foreign keys may be duplicate or null.

Rules :-

Parent table	Child table
✓ Insert < 4 D ECE >	✗ Insert < 105 GT 5 >
✗ Delete < 1 A CSE >	✓ Delete < 103 TOC 3 >
✗ update < 2 B CSE >	✗ Update < 102 RDMS 1 >

No update

✗ → Violence

✓ → No violence

Here, Deletion and updation from referenced relation and insertion and updation from child table/referencing relation may cause violence foreign key constraint!

solution :- Foreign key with Ondelete Cascade

:- In this process, if data from parent key is deleted or updated then the related data from child table also be deleted / updated cascadely / simultaneously.

eg:-

Parent table

RNO	Name	Age
1	A	17
~~2~~	~~B~~	~~15~~
3	A	18

Parent key

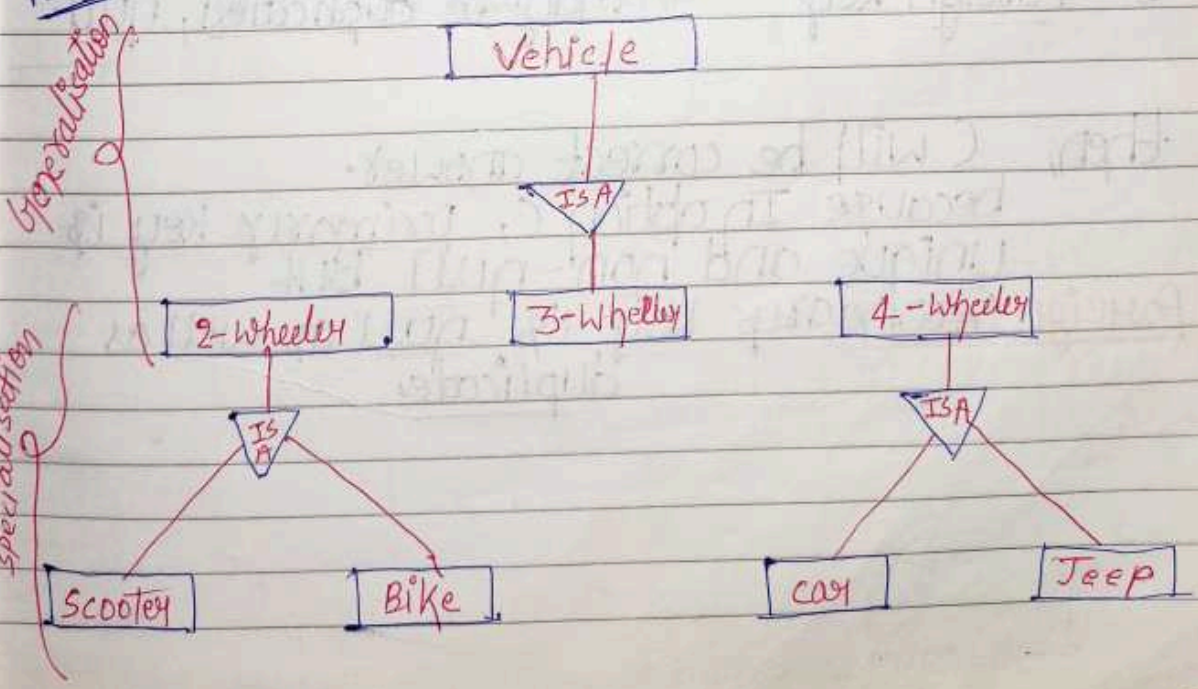
Child table

Class	Course	RNO
101	CSE	1
~~102~~	~~CD~~	~~2~~
103	TOC	3

Foreign key

let, us suppose, if we want to delete record of student whose rollno. is 2 ~~then~~ from parent table then there need be to delete the related data from child table simultaneously.

Note



Page No.
Date / /

Ques :- Consider a relation R(A,B) where A is primary key and B is foreign key referencing the same relation then which of the following row sequence can be inserted successfully into R.

a) (1, null) (2, 1) (2, 2) (3, 2)

b) (null, 1) (1, 2) (2, 3) (3, 4)

c) (1, null) (2, 1) (3, 2) (4, 2)

d) all.

→ On keeping the concept of primary key and foreign key as :-

A = Primary key → must be unique, non-null, non-duplicate.

B = Foreign key → May be duplicated, null.

then, C will be correct answer.
because In option C, primary key is unique and non-null but Foreign ~~secondary~~ key is null as well as duplicate.

Page No.
Date / /

Ques Consider a relation R(A, B) where A is a primary key and B is a foreign key referencing A with on delete cascade. consider the following relation instance on R.

R(A B)
(2 4)
(3 4)
(4 3)
(5 2)
(7 2)
(9 5)
(6 4)

What are the tuples that must be additional deleted to preserve referential integrity constraint when the tuple (2,4) deleted is.

a) (3,4) (4,3)
b) (3,4) (4,3) (6,4)
c) (5,2) (7,2)
d) (5,2) (7,2) (9,5)

First which tuple is deleted.

→

* ER model to Relational model

i) Entity set.
:→

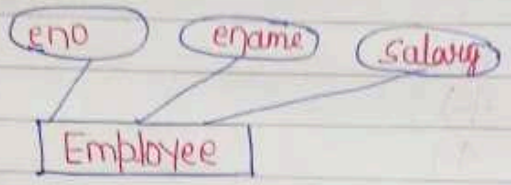


Table:- Employee

eno	ename	salary

- An entity set is mapped with a relation.
- The key attribute of an entity becomes primary key of the relation.

ii) Entity set with composite attributes

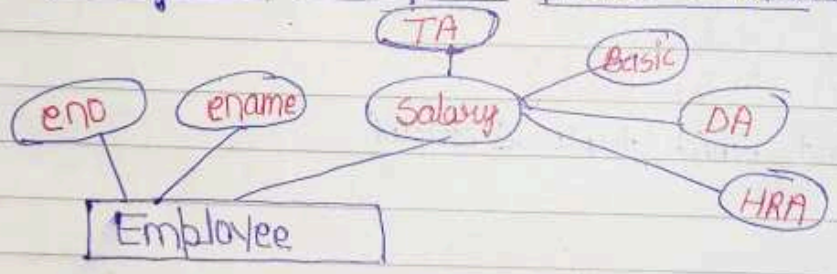
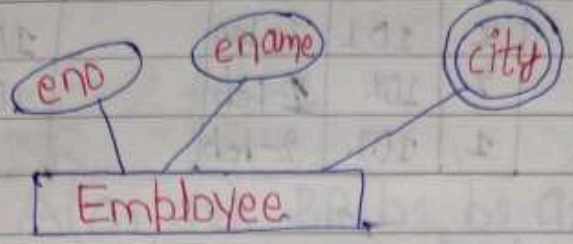


Table :- Employee

eno	ename	Basic	TA	DA	HRA
1	A	5000	3000	5000	1200

- The salary attribute of a relation includes the simple attributes Basic, TA, DA, HRA etc.

iii) Entity set with multivalued attribute



Employee

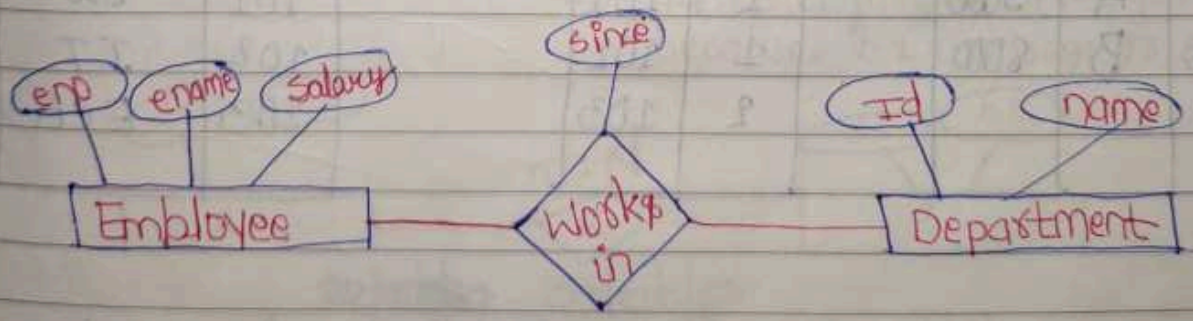
e.no	ename
1	A
2	B

Employee-address

e.no	city
1	Hyd
1	Bang
2	Bang
2	Pune

- If an entity contains multivalued attributes. It is represented with two relations one with all simple attributes and the other with key and multivalued attributes.

iv) Translating relationship set into a table.



Employee — Foreign key — Works in — Foreign key — Department

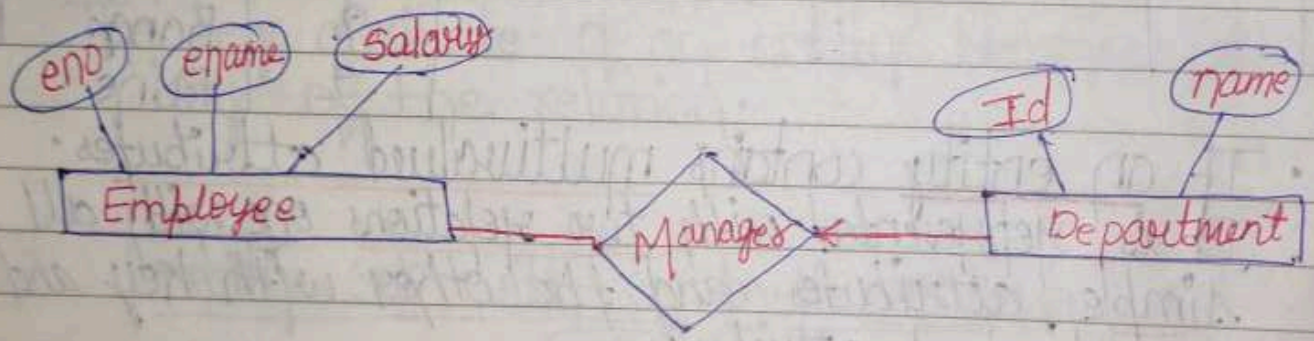
eno	ename	Salary
1	A	5000
2	B	9000

eno	Id	Since
1	101	1-Jan
2	102	1-Feb
1	102	2-Feb

Id	name
101	CSE
102	IT

v.) Relationship set with key constraint

:→ Each department is required to have almost one employee as a manager.



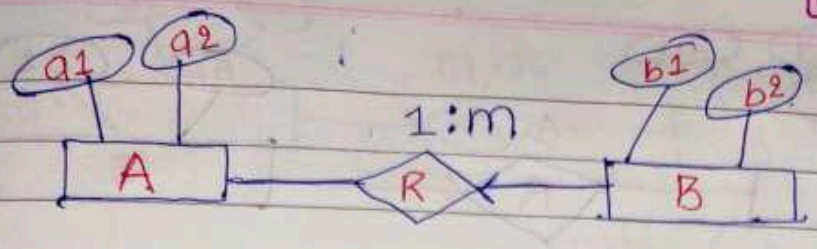
Employee — Foreign key — Manager — Foreign key — Department

eno	ename	Salary
1	A	5000
2	B	8000

eno	Id
1	101
1	102
2	103

Id	name
101	CSE
102	IT
103	ECE

i.)

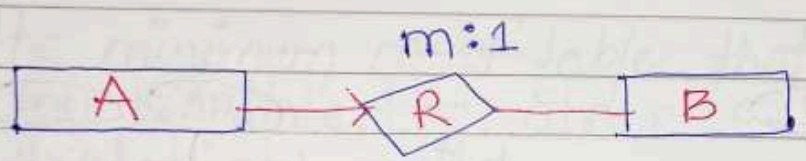


⇒ A, BR 2 tables

A(a1, a2), BR(b1, b2, a1) Foreign key

↳ It is foreign key of B because a1 belongs to A.

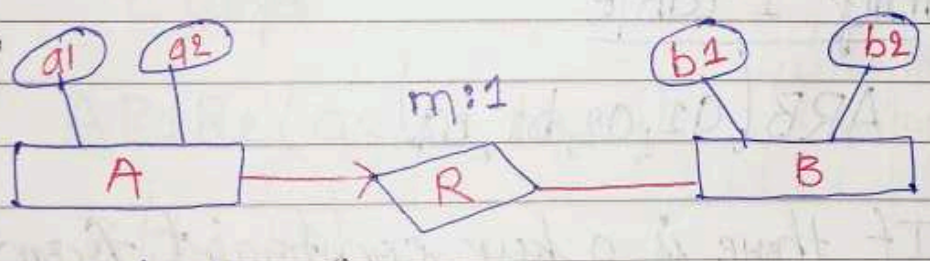
ii.)



2 tables

⇒ (AR, B).

iii.)

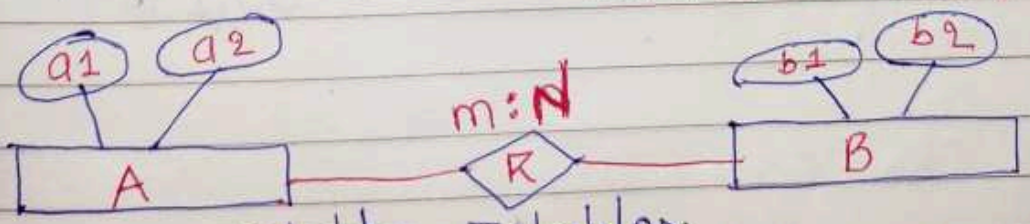


2 tables

⇒ AR(a1, a2, b1) ; B(b1, b2)

↳ It is a foreign key for A because b1 belongs to B.

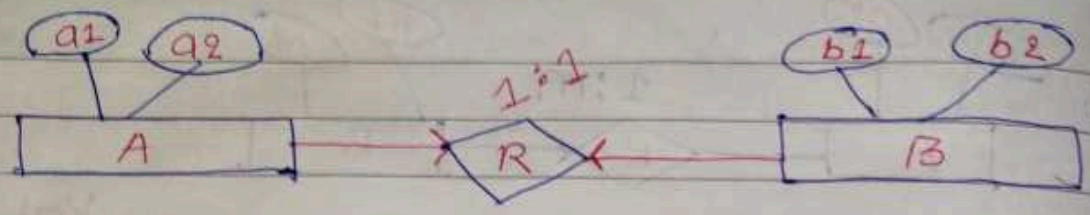
iv.)



~~2 tables~~ 3 tables.

i.e., A(a1, a2) R(a1, b1) B(b1, b2)

v.)

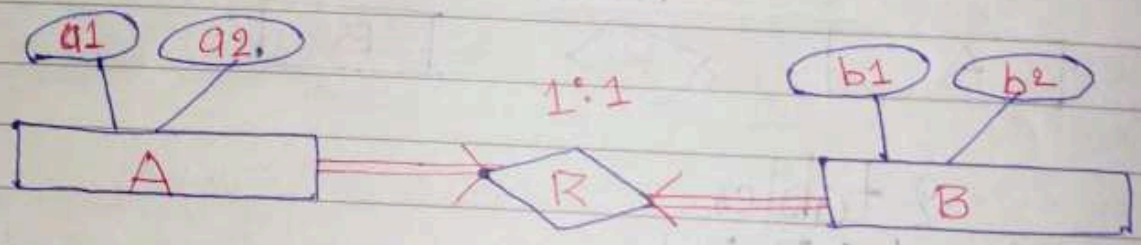


2 tables.

i.e; AR(a1, a2, b1) , B(b1, b2) OR

A(a1, a2) , BR(b1, b2, a1)

vi.)

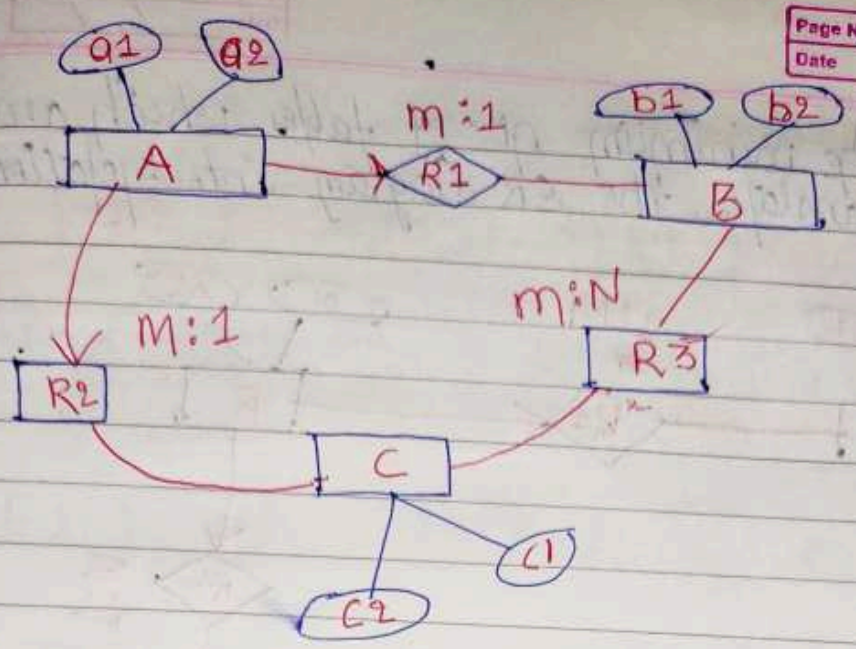


Only 1 table

i.e; ARB(a1, a2, b1, b2)

Note :- If there is a key constraint from both the sides of an entity set with total participation then we represent that binary relationship using single table

Ques



Find the minimum no. of tables that one possible when you translate, the above E-R diagram into relational model.

Sol:- 4 tables

i) AR1R2 (a1, a2, b1, c1)

Foreign key because b1 belongs to B & c1 belongs to C.

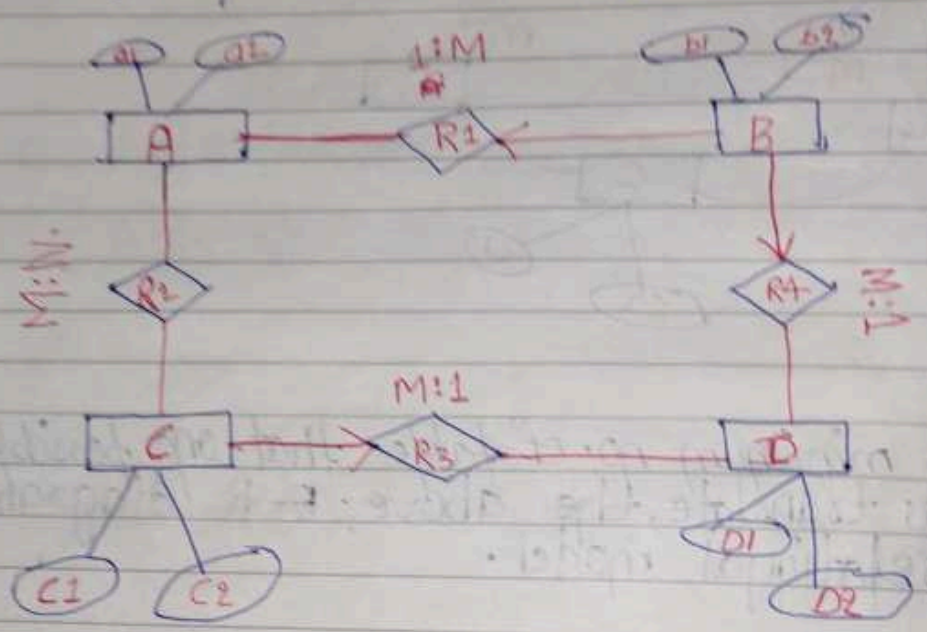
ii) B (b1, b2)

iii) R3 (b1, c1)

Foreign key

iv) C (c1, c2).

Ques Find the minimum no. of tables which are required to translate the ER diagram into relational model



Ans:- 5 tables

i) BR1R4R5 (b1, b2, a1, c1, D1). Foreign key since a1 belongs to A, D1 belongs to D, c1 belong to C

ii) A(a1, a2).

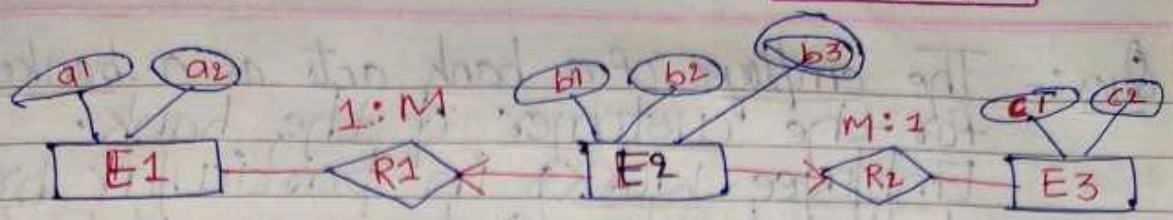
iii) R2(a1, c1). Foreign key

iv) D(D1, D2)

v) CR3(c1, c2, D1). Foreign key.

Page No.
Date / /

Ques

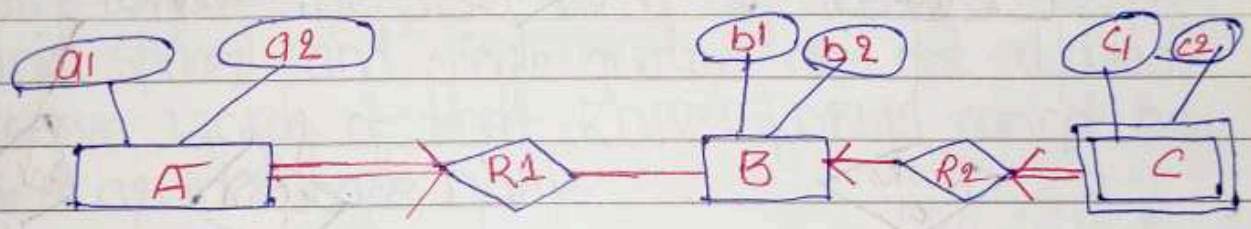


Find total no. of table (minimum).

Sol:- E1(a1, a2) E2(b1, b2, b3, a1, c1) E3(c1, c2)

so, minm no. of tables = 3.

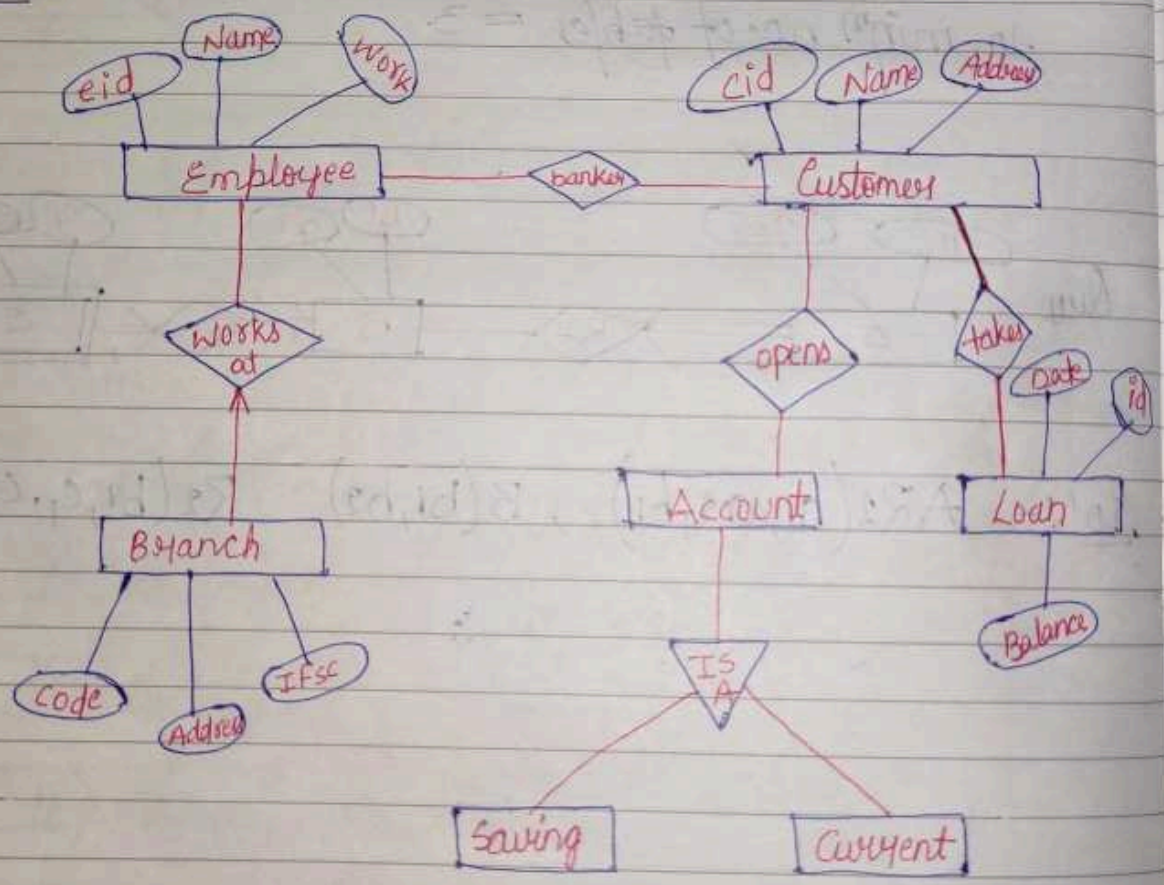
Ques



Sol:- AR1(a1, a2, b1), B(b1, b2) R2(b1, c1, c2).

Ques :- The employee of a bank acts as a banker for the customer of the bank. Employee works in particular branch of the bank. Customer can take loan from the bank. At the same time, a customer can open either saving or current account.

Sol :-



Here, Entity

Employee
Customer
Branch
Account
Loan

Attributes

eid, Name, work.
cid, Name, Address.
code, Address, IFSC
saving, current as type
date, id, balance.

Practical

• CREATE DATABASE

:→ It is used to create database.

Syntax :- CREATE DATABASE database_name;

eg :- CREATE DATABASE my_db;

Guidelines for database name

- Name should be start with an alphabet.
- Blank spaces and single quotes are not allowed.
- Reserve words of that RDBMS/DBMS cannot be used as database name.

Steps :-

- Open mysql command line client - Unicode.

MySQL Command line client

we can write in small letter also.

```
mysql > CREATE DATABASE my_db;
Query OK, 1 row affected (0.00 sec)
```

database name

It signify that database is created.

Page No.
Date / /

This is used to tell your RDBMS/DBMS that you want to use this database.

syntax :- USE database_name;
eg :- USE my_db;

steps :- Open mysql command line client

```
mysql > CREATE DATABASE my_db;
Query OK, 1 row affected (0.00 sec)

mysql > USE my_db;   + enter
Database changed
```

↓ Feedback

It is saved in this database.

- CREATE TABLE

syntax :-

```
CREATE TABLE table_name
(
 column_name1 data_type(size),
 column_name2 data_type (size),
 column_name3 data_type (size)
);
```

or

```
CREATE TABLE table_name
(
 column_name 1 datatype(size) [constraints],
 column_name 2 datatype(size) [constraints]
);
```

NOT NULL
PRIMARY KEY

Keyword to create table

database name

eg:- CREATE TABLE my_tab
(
name varchar (30),
roll number (4),
address varchar (100)
);

field or column name

datatypes (varchar, number).

semicolon indicates that the above statement is ended.

Guildlines for create Table

- Tablename should start with an alphabet.
- Blank spaces and single quotes are not allowed.

- DESC

:→ This is used to describe your table. DESC only describes structure of table, not the information (rows) inside table.
- DESC is short form of Describe.

syntax :→ DESC table_name;

eg:- DESC my_tab;

Page No. Date / /

Notes

~~show databases/tables~~

SHOW DATABASES | TABLES

- SHOW DATABASES — This command is used to view all the databases name:

syntax :- SHOW DATABASES;

- SHOW TABLES — This command is used to view all the tables of current database

syntax :- SHOW TABLES;

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| db       |
| my_db    |
+----------+
2 rows in set (0.00 sec)

mysql> USE my_db;

mysql> SHOW TABLES;
+----------+
| my_tab   |
+----------+
```

It displays all the database created

It will open my database named my_db

It shows all the tables present in my current database :- USE my_db

Data Types

- INT or INTEGER

:- It holds whole number between -32768 and ~~32768~~ 32767 either it is negative or positive.
- It cannot hold any decimal value.
- The maximum number of digits may be specified in parenthesis. by size

syntax :- column_name INT(size);
eg :- roll INT(5);

→ size
→ data types
→ column_name

- DEC or DECIMAL → column_name DECIMAL(T,D)
  → where, T = Total digits
  Range = 1-65

:- It holds fixed point numbers. eg → 2368.981 Range = 7
- Size is the total number of digits and p is the number of digit after decimal point.
- The decimal pt and -ve sign are not counted in size.

syntax :- column_name DECIMAL(size, p)

eg :- price DECIMAL(4,2) → no. of digit after decimal
→ size

⇓

56.89 or -12.34

→ -ve sign and decimal pt is not counted

counting of no. of digit starts from left of dec. mark

No. of Digit	No. of Bytes
0	0
1-2	1
3-4	2
5-6	3
7-9	4

Eg:- 1699.81 = 2+1 = 3 byte

no. of digit 4 1 2

no. of bytes 2 1 1

Page No. Date / / KRISHNA

Eg:- 14564345 66999.81

No. of digit 9 4 2

No. of bytes 4 2 1

total bytes = 4+2+1 = 7

• CHAR OR CHARACTER

:→ It holds a fixed length string (can contains letters, numbers, special character etc).

• The fixed size is specified with in parenthesis

syntax :- column_name CHAR(20);

eg :- name CHAR(20); → size

→ data type

→ column_name

• VARCHAR

:- It holds a variable length, string (can contains letters, numbers, and special character).

• The maximum size is written in parenthesis

• It can store upto 255 characters.

Note :- If we put a greater value than 255 then it will be converted to a TEXT Type.

syntax :- column_name VARCHAR(size);

eg :- name VARCHAR(50); → size

→ data type

→ column_name

• TEXT

:→ It

syntax

eg

• DATE

:- It

syntax

eg

• TIM

:- It

syntax

eg

• TEXT

:→ It holds a string with a maximum length of 65,535 characters.

syntax :- column_name TEXT;
eg :- address TEXT;

→ data types.

→ column_name

• DATE

:- It displays Date values in yyyy-mm-dd format.

syntax :- column_name DATE;
eg :- age DATE;
For DOB (Date of birth)

• TIMESTAMP

:- It displays date and time.

syntax :- column_name TIMESTAMP;

eg :- login_dt TIMESTAMP;

Page No.
Date / /

- CREATE TABLE named as saket

```
mysql> USE my_db;
Database changed

mysql> SHOW TABLES;
```

my_tab { table exists

```
mysql> CREATE TABLE saket
    -> (
    -> student_id INT(5),
    -> name ~~varchar~~ VARCHAR(50),
    -> address TEXT,
    -> dob DATE,
    -> fee DEC(10,2)
    -> );
```

→ No. of digit after decimal

↳ size

Query OK, 0 rows affected.

```
mysql> DESC saket;
```

Field	Type	Null	Key	Default
student_id	int(5)	Yes		NULL
name	varchar(50)	YES		NULL
address	text	YES		NULL
dob	date	YES		NULL

DML

Page No.
Date / /

⇒ Database Manipulation Languages

:- These commands are applied on data stored in the table.

- Different DML commands are :-

Statement	Description
SELECT INSERT UPDATE DELETE MERGE	Retrieves data from the database, enter new rows, changes existing rows, and removes unwanted rows from tables in the database resp.

SELECT

:- It is also considered as a Data Query Language (DQL).

- To select data or retrieve data from database we use SELECT command.
- Table name is used with FROM clause.
- Define condition with WHERE clause.
- To select all columns:

— SELECT * FROM Persons WHERE P_ID <= 5;

- To select specific columns

— Select P_ID, Pname FROM Persons WHERE P_ID <= 5;

Page No.
Date / /

1. To select all columns from the table

syntax :- SELECT * FROM table_name;

all

2. to select particular columns from the table

syntax :- SELECT column1, column2, ...
FROM table_name;

particular table

eg :- SELECT name, city, pin
FROM bholi;

3. To SELECT with LIMIT

:→ It is used to specify some number of records to display.

syntax :- SELECT column_name1, column_name2
or [SELECT *
FROM table_name
LIMIT record_number.

eg :- → SELECT * FROM bholi
→ LIMIT 5;

→ It displays data or records of table upto 5.

syntax :- [SELECT * FROM bholi
→ LIMIT 2, 5;

no. of records

→ It displays data or records of table from index no. 2 to 6.

index no.

syntax :- SELECT * FROM bholi
LIMIT 3, 7;

no. of records.

index no. 3

• INSERT INTO

a) Insert with specifying column name

:- The INSERT INTO statement is used to insert new records/rows/tuple in a table.

syntax :- INSERT INTO table_name (col_name1, col_2)...
VALUES (value 1, 'value 2', 'value 3', ...);

table_name

These are all column present in table

If inserts value into table

INT data type

VARCHAR data type so, value is within ''

Eg :- INSERT INTO my_tab (stu_id, name, address)
VALUES (05, 'Anu', 'Delhi');

Rules :- column and value datatype should be same.

- Any value that goes into a VARCHAR, CHAR, DATE or TEXT column has single quotes around it.
- No need of quotes for numeric values for INT, DEC.

Note • To see all the information inserted into table by using :-

SELECT * FROM table_name;

b) Insert into table without specifying column name

syntax :-

```
INSERT INTO table_name
VALUES (value1, 'value2', 'value3', value4);
```

⇒ Sequence is very important in this case.

Note :- To see all the information inserted into table by using :-

```
SELECT * FROM table_name;
```

c) changing the order of column

bholi

stu_id	name	city	pin

syntax :- INSERT INTO table_name (column2, column1, column3, ...)
VALUES ('value2', value1, value3);

see above operation

```
SELECT * FROM table_name;
```

d) Insert into table with specifying columns

syntax:-

```
INSERT INTO table_name(column1, column2,
                       column3)
VALUES (value 1, 'value2', 'value3');
```

Note :- if there are 4 columns which are present in table but we want to insert data in only 3 columns.

eg :-

- table → bhali

stu_id	name	address	pin

```
→ INSERT INTO bhali (stu_id, name, pin)
→ VALUES ( 1, 'saket', 211019);

Query OK

mysql > SELECT * FROM bhali;
```

stu_id	name	address	pin
1	saket	NULL	211019

We donot pass any values for column address of the table 'bhali'

e.) Insert multiple record at one time

syntax :-

```
INSERT INTO table_name ( column 1, column 2
                         column 3, col4)
VALUES (value 1, 'value 2', 'value 3',
        value 4),
       (value 1, 'value 2', 'value 3', value4),
       (value 1, 'value 2', 'value 3', 'value4);
```

It is used to add multiple record within the column of table at one time.

to see ~~the~~ insertion / data

```
SELECT * FROM table_name.
```

juzzz.com

Single quotes Problem

Eg :- INSERT INTO table_name (column1, column2, to column3)
VALUES (142, "K.K's company", 'Delhi');

Due to this single quote, it shows error because system doesnot fetch detail due to multiple quotes.

There are two ways to overcome this problem,

i.) USE backslash
:- 'K.K\'s company'.
→ backslash

ii.) Use two times single quotes.
:- 'K.K''s Company'.
→ single quotes

Eg :- INSERT INTO bholi (id, c_name, city)
VALUES (142, 'K.K\'s company', 'Delhi');
→ table_name

Or,
INSERT INTO bholi (id, c_name, city)
VALUES (142, 'K.K''s company', 'Delhi');

Page No.
Date / /

WHERE Clause :- It is used to filter specific data.

:→ WHERE is used to search for a specific data.

- It exacts only those records which satisfy specified condition

syntax :- a) Specific data from all column

```
SELECT * FROM table_name
WHERE column_name operator 'value';
```

eg :-
```
SELECT * FROM bhali → table_name
WHERE stu_id = 9;
```
→ value, → operator, → column_name

sto_id	name	city	pin
9	Saket	Sasaram	821115

b) specific data from specific column (multiple column may possible)

syntax :-
```
SELECT column_name FROM table_name
WHERE column_name = 'value'.
```
→ operator

eg :-
```
SELECT name FROM bhali → table_name
WHERE name = 'Anu'
```
→ value, → column_name

name
Anu

Note :- Value can be text or numeric. If it is text then put it into single quote

operator

operator	Description
=	Equal
<> or !=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
* LIKE	Search for a pattern
IN	To specify mul. values for column

- Equal

```
SELECT * FROM bhali
WHERE stu_id = 9;
```

table_name
value
column_name
operator

- Not Equal

```
SELECT * FROM bhali
WHERE stu_id != 9;
```

- Greater than

```
SELECT * FROM bhali
WHERE stu_id > 9;
```

operator

or WHERE name>'R';

- Less than

```
SELECT * FROM bhali
WHERE stu_id < 9;
```

• SELECT * FROM bholi
WHERE name > 'J' → It displays all the name which starting with letters which are greater than J.

Page No. KRISHNA
Date

• BETWEEN

:→ It selects values within a range.

```
SELECT * FROM bholi
WHERE column_name BETWEEN value1 AND value2
```

a) Between text

Syntax:-
```
SELECT *
FROM bholi
WHERE name BETWEEN 'J'
AND 'S'
```
↳ operator

• BETWEEN DATE

Syntax :-
```
SELECT * FROM bholi
WHERE column_name
BETWEEN 'yyyy/mm/dd' AND
'yyyy/mm/dd';
```

• IN

:→ It specify multiple values in a WHERE clause or displays

Syntax:-
```
SELECT * FROM table_name
WHERE column_name IN ('value1','value2');
```

eg:-
```
SELECT * FROM bholi
WHERE name IN ('Anu','sonu');
```

stu_id	name	city	pin
2	Anu	Delhi	111025
3	sonu	Mumbai	546879
4	Anu	Kolkata	568974

⇓ It displays all the records in which name is Anu & sonu.

• LIKE:- It is used to search for a specified in a column.

Syntax :-
```
SELECT * FROM table_name
WHERE column_name
LIKE 'pattern';
```

eg:-
```
SELECT * FROM bholi
WHERE name LIKE '%nu';
```

Wildcards :- It is used to search data within a table

% → zero or more characters

_ → One single character

[ ] → Ranges of character to match.

[^] → Matches only a character

% → zero or more char
- 'Geek%' → All start with Geek
- '%shows' → Ending with shows
- '%sh%' → contain with sh

_ One single character
- 'show_' → starting with show
- '_eek' → any character then eek

KRISHNA
Page No.
Date / /

NULL Value :- It is diff^n from zero value.

:→ NULL values represent missing unknown data
- A field with NULL value is one that has been left blank during creation.

a.) **IS NULL**
↳ It is used to test for empty values.

:→ This is used to select only the records with NULL Values in the column.

Syntax :-

```
SELECT * FROM table_name
WHERE column_name IS NULL;
```

↓ It displays records with NULL Value.

eg:- table :- bholi

stu_id	name	city	pin
1	Saket	Prayagraj	211019
2	Ankit	NULL	821115
3	Sumit	Sasaram	821115

- mysql > SELECT * FROM bholi
  → WHERE city='NULL'; ✗ It shows an error

- mysql > SELECT * FROM bholi
  → WHERE city IS NULL; ✓ correct way to display NULL value

table :- bholi

stu_id	name	city	pin
2	Ankit	NULL	821115

match.
character NOT specified within the bracket

b) IS NOT NULL

:→ This is used to select only the records with no NULL values in the column

syntax:- SELECT column_name FROM table_name WHERE column_name IS NOT NULL;

Eg:- mysql > SELECT * FROM bholi
→ WHERE city IS NOT NULL;

table :- bholi

stu_id	name	city	pin
1	Saket	Prayagraj	211019
2	Sumit	Sasaram	821115

c) IS NULL operator

:- It is used to test for empty values.

Eg:→ SELECT CustomerName, ContactName, Address
From customers
WHERE Address IS NULL;

⇓

It lists all customers with a NULL value in the "Address" field.

AND operator

:→ AND operator displays a record if both the first condition AND the second condition are true.

syntax :-
```
SELECT * FROM table_name
WHERE column_name = 'value'
AND column = 'value';
```

↓

It displays those columns which satisfy both condition.

table :- bholi

eg :-

stu_id	name	city
1	Saket	Prayag
2	Shubham	Sasaram
3	Saket	Patna

mysql >
```
SELECT * FROM bholi
WHERE name = 'Saket'
AND stu_id = 3;
```

Ist condition

IInd condition

It shows only those records which satisfy both condition.

stu_id	name	city
3	Saket	Patna

Page No.
Date / /

OR operator

:→ OR operator displays a record if either the first condition OR the second condition is true.

syntax :-

```
SELECT * FROM bholi
WHERE column_name = 'Value'
OR column = 'Value';
```

Ist

IInd condition

⇓

It displays if either Ist OR IInd condition is true. (any condition is true)

Combinations of AND & OR.

syntax :-

```
SELECT * FROM table_name
WHERE column_name = 'value'
AND(column_name = 'value' OR
    column_name1 = 'value');
```

eg:- SELECT * FROM bholi
WHERE name = 'Anu' ~~OR~~ → name must be Anu
AND (stud_id = 5 OR address = 'Kolkata');

⇓

- name must be Anu but
stu_id may be 5
address may be Kolkata.] atleast any of them is true.

table :- bhali

stu_id	name	city	pin
2	Anu	Delhi	111025
3	Sonu	Mumbai	546879
4	Anu	Kolkata	568974

eg :-

```
mysql> SELECT * FROM bhali
       WHERE name = 'Anu'
       AND (city='Delhi' OR stu_id=4);
```

stu_id	name	city	pin
2	Anu	Delhi	111025
4	Anu	Kolkata	568974

Here, name must be Anu but city and stu_id are connected with OR operator
so, city may be Delhi or not
stu_id may be 4 or not

NOT LIKE

syntax :-

```
SELECT * FROM table_name
WHERE column_name NOT LIKE 'pattern';
```

eg :-

```
SELECT * FROM bhali
WHERE name NOT LIKE '%nu';
```

⇓

It displays all the name not ending with nu.

~~ORDER BY desc~~

NOT operator

:- NOT operator displays a record if the condition(s) is NOT TRUE.

Syntax :-

```
SELECT column 1, column 2, ....
FROM table_name
WHERE ~~condition~~ NOT condition.
```

Eg:-

```
SELECT * From Customers
Where NOT country = "Germany"
```

ORDER BY

- ASC → It sorts in ascending order.
- DESC → It sorts in descending order.

Syntax :-

```
SELECT * FROM table-name
ORDER BY column_name ASC or DESC
```

Page No.
Date / /

• ORDER BY descending order

• Using keyword :- DESC
↓
It sorts in descending order

syntax :-
```
SELECT * FROM table_name
ORDER BY column_name DESC;
```

Eg :-
```
SELECT * FROM bholi
ORDER BY stu_id DESC;
```

table :- bholi

stu_id	name	city
3	S	Ptna
4	K	Dlhi
2	A	Kol.

After DESC →

table :- bholi

stu_id	name	city
4	K	Delhi
3	S	Patna
2	A	Kolkata

• ORDER BY Ascending order (By default)

• Using keyword :- ASC
↓
It sorts in Ascending Order

syntax :-
```
SELECT * FROM table_name
ORDER BY stu_id ASC;
```

Eg :-
```
SELECT * FROM bholi
ORDER BY stu_id ; (ASC is optional)
```

Eg :-
```
SELECT * from bholi
ORDER BY stu_id DESC, name ASC;
```

Constraint → NOT NULL
→ Unique Key
→ Primary Key

1. NOT NULL

:→ NOT NULL constraint enforces a field to always contain a value.
This means that you cannot insert a new record, or update a record without adding a value to this field.

Eg:→
```
CREATE TABLE student
(
  Name varchar(30),
  Roll integer(5),
  Mobile_no integer(10) NOT NULL
);
```

These may be empty.

Constraints

It signifies that mobile_no. can't be empty.

```
mysql> DESC student;
```

Field	Type	NULL	Key	Default	Extra
Name	Varchar(30)	YES			
Roll	int(5)	YES			
Mobile_no	int(10)	NO			

- There must be any value.
- It is compulsory to fill data or value in mobile_no.

2. Unique key (UNIQUE KEY)

:- It is a type of constraint, it uniquely identifies each record in a database table.

- There can be many ~~unique~~ UNIQUE constraint per table.
- A UNIQUE key column can contain NULL values.

Eg :→ CREATE TABLE student
(
Name varchar(30), UNIQUE KEY,
Roll integer (5),
Mobile_no integer (10) UNIQUE KEY.
);

⇓

- It uniquely identify.
- May contain null value
- More than one UNIQUE KEY can be possible

mysql> DESC student;

Field	Type	NULL	Key	Default	Extra
Name	varchar(30)	YES	UNI	NULL	
Roll	int(5)	YES		NULL	
Mobile_no	int(10)	YES	UNI		

mysql> SELECT * FROM student
→ Values ('A', 101, 9876542123),
('A', 102, 9654387921);

⇓

It shows duplicate entry 'A' for Name field

3. PRIMARY KEY

:→ It also uniquely identifies each record in a database value.

- PRIMARY KEY must contain a ~~prime~~ UNIQUE values.
- A primary key column cannot contain NULL
- Each table can have ONE Primary key

syntax:-

```
CREATE TABLE student
(
Name varchar(30),
Roll integer (5) NOT NULL,
Mobile_no integer (10),
PRIMARY KEY (Roll)
);
```

2 way to write PK in table.

Or

```
CREATE TABLE student
( Name varchar(30),
Roll integer (5) NOT NULL PRIMARY KEY,
Mobile_no integer (10)
);
```

mysql > DESC student;

Field	Type	NULL	Key	Default
Name	Varchar(30)	YES		
Roll	int(5)	NO	PRIMARY	
Mobile_no	int(10)	YES		

~~* Auto Increasement~~

~~Auto Increase~~

* AUTO_INCREMENT

a.) General

:→ AUTO INCREMENT is used to generate a unique number, when a new record is inserted into a table.

syntax :-

```
( CREATE TABLE table_name

    column_name int NOT NULL AUTO_INCREMENT,
    column_name1 varchar (50) NOT NULL,
    column_name2 varchar (50),
    PRIMARY KEY (column_name)
    );
```

It is the column_name corresponding to auto_increasement

Eg :-

```
→ ( CREATE TABLE t_tab
→ * Roll int NOT NULL AUTO_INCREMENT,
→ Name Varchar (30),
→ City Varchar (50),
→ PRIMARY KEY (Roll) *
→ );
```

tablename

It is that column_name which are related to Auto INCREMENT

Field	Type	NULL	Key	Default	Extra
Roll	int	No	PRI	NULL	auto_increment
Name	varchar(30)	YES		NULL	
City	varchar(50)	YES		NULL	

eg:- mysql → INSERT INTO t_tab(Roll, Name, City)
→ VALUES (1, 'SAKET', 'PRAYAGRAJ'),
(NULL, 'Aman', 'Sasaram')
(NULL, 'Bholi', 'Patna');

In AUTO_INCREMENT, If we pass NULL then it generates next values instead of NULL)

mysql > SELECT * FROM t_tab;

Roll	Name	City
1	Saket	PRAYAGRAJ
2	Aman	Sasaram
3	Bholi	Patna

We pass NULL value but it generates 2 and 3 in the place of NULL

Note :- If I pass NULL value for AUTO_INCREMENT column then it ignores NULL and generates unique number for them.

Case:-I If both values are NULL

mysql> INSERT INTO table_name
VALUES
(NULL, 'SHUBHAM', 'Sasaram'),
(NULL, 'SAKET', 'Prayag');

Here, It ignores NULL value and generates unique number in an order for them

Roll	Name	City
1	SHUBHAM	Sasaram
2	SAKET	Prayag

Case :- II If Ist value is NULL and IInd value is 1.

```
mysql> INSERT INTO table_name
       VALUES
       (NULL, 'Shubham', 'Sasaram'),
       ( 1, 'Saket', 'Patna');
```

Here, It ignores NULL value and generate 1 but the second which I pass here is already 1. So, (1, 'Saket', 'Patna') is automatically erased from database.

Roll	Name	City
1	Shubham	Sasaram

Case :- III If Ist value is 1 and IInd value is NULL

```
mysql> INSERT INTO table_name
       VALUES
       ( 1, 'Shubham', 'Patna'),
       (NULL, 'Saket', 'Delhi');
```

Here, The first auto_increasent column value is 1 and the second value is NULL then auto_increasement ignores NULL value and generate 2.

Roll	Name	City
1	Shubham	Patna
2	Saket	Delhi

5> AUTO_INCREMENT with particular number

:→ By default, AUTO_INCREMENT starts from 1.
But, if we want to start number from the given particular number then we use syntax :-

ALTER TABLE table_name AUTO_INCREMENT = 10;

Keyword

syntax :-

```
CREATE TABLE table_name
(
id int NOT NULL AUTO_INCREMENT PRIMARY KEY,
name varchar (50)
);
```

mysql > DESC table_name;

Field	Type	NULL	KEY	DEFAULT	EXTRA
id	int	NO	PRI	NULL	auto_increment
name	varchar(50)	YES		NULL	

```
mysql > ALTER TABLE table_name AUTO INCREMENT=10;
mysql > INSERT INTO table_name (name)
        VALUES ('RAM'),
               ('ALi');
```

id	name
10	RAM
11	Ali

c) AUTO_INCREMENT WITHOUT specifying column name

syntax :-

```
INSERT INTO table_name
VALUES (NULL, 'Rohan');
```

```
mysql> SELECT * FROM table_name.
```

id	name
10	Ram
11	Ali
12	Rohan

Aliases (AS)

:→ Aliases are used to temporary rename column name or a table name.

For Table

syntax :- SELECT column_name
FROM table_name AS alias_name;

Ex :→ SELECT name FROM student AS vidyarthi;

previous table name — new table name

Here, this statement change the name of table from student to vidyarthi

Page No.
Date / /

For column

Syntax :- SELECT column_name
AS alias_name FROM table_name;

Eg :- SELECT name AS student_name FROM student;

Here, It changes the ~~name~~ column name of name from student table to student_name.

Note :-

student_name
Or
"student name"
Or
[student name]

all are same

Arithmetic Operators (*, /, +, -).

:- We can operate all arithmetic operators in mysql.

syntax :→

⇒ SELECT column_name, column_name Operator value FROM table_name;

Eg :- SELECT name, cost, cost+100 FROM bhali

column_name → operator value → table_name

Or

SELECT name, cost, id, cost+100 AS "New Cost" FROM items_tab.

name	cost	~~id~~	Newcost

Note :- For operator value

⇒ We can add value of two or more columns in any columns.

Newcost = cost + 100
or Newcost = cost + id.

Page No.
Date / /

SELECT DISTINCT

→ It is not used in primary & Unique key

:→ SELECT DISTINCT statement is used to display only distinct (different) values.

→ It doesnot show duplicate value

syntax :- SELECT DISTINCT column_name FROM table_name.

Eg :→ SELECT DISTINCT name FROM bhali;

↓ column_name ↓ table_name

table :- emp

id	name	dept	salary	city
1	Ram	IT	30000	Delhi
2	Amit	HR	35000	Mumbai
3	Sunil	Trainer	40000	Kolkata
4	Ram	Executive	10000	Bhopal
5	Dev	IT	25000	Patna

mysql > SELECT DISTINCT name FROM emp

⇒ Ram
Amit
Sunil
Dev

Here; In emp table, Ram exist two times but it display only once.
It ignores duplicate entry.

mysql > SELECT DISTINCT dept FROM emp.

column_name

table_name

⇒ IT
HR
Trainer
Executive

Here; In dept column, IT exists two times but it ignores duplicate entry of IT and show only once.

ALTER TABLE

Data Definition Language (DDL)

:- CREATE DATABASE	CONSTRAINT (Keys)
ALTER DATABASE	CREATE INDEX
DROP DATABASE	DROP INDEX
CREATE TABLE	CREATE VIEW
ALTER TABLE	ALTER VIEW
DROP TABLE	DROP VIEW

ALTER TABLE

:→ This command is used to Add/change/Modify/DROP existing structure of Table.

1. ADD Column.
2. Enable / Disable constraints.
3. Change Column
4. Modify Column
5. Drop Column

1. ADD Column → By default, it adds new column in last.

(a) Add one column

:→ When a new column is to be added to the existing table structure without constraints.

syntax :-

```
ALTER TABLE table_name
ADD COLUMN column_name
        datatype (size);
```

Name	Roll

Eg :-

```
ALTER TABLE bholi
ADD stu_id INT(5);
```

COLUMN

It adds new column in existing table

Name	Roll	stu_id

(b) Add more than one column

syntax :-

```
ALTER TABLE table_name
ADD COLUMN column_name datatype(5),
ADD COLUMN column_name datatype(size);
```

It adds two columns in existing table

Eg :-

```
ALTER TABLE bholi
ADD COLUMN city varchar(20),
ADD COLUMN email varchar(30);
```

Name	Roll	Stu_id	city	email

(c) ADD COLUMN by Position

- LAST (By default)
- FIRST
- AFTER

• FIRST

syntax :-

```
ALTER TABLE table_name
ADD COLUMN column_name datatype(size) FIRST;
```

It adds new column at the FIRST Field.

S.No	Id	Name	city	pin

• AFTER

:→

syntax :- ALTER TABLE table_name
ADD COLUMN column_name datatype(size)
AFTER column_name;

It adds new column after the any other column.

Eg :- ALTER TABLE table_name
ADD COLUMN state varchar(30)
AFTER city ;

Here, It adds new column named as state after the column city.

S.No	Id	Name	City	State	pin

(d) Add column with constraints

:- When a new column is to be added to the table with constraints.

Syntax :- ALTER TABLE table_name
ADD COLUMN column_name datatype(size)
CONSTRAINT _name;

Eg:- ALTER TABLE items → table
ADD COLUMN item_no int(5) NOTNULL;

item_no → new column which we want to add

NOTNULL → constraints

2. Keyword → CHANGE COLUMN

Page No. KRISHNA
Date / /

A. change column name and its data-type without constrain

a.) one column and its datatype

:→ This is used to change name and its data-types of an existing column.

syntax :-

```
ALTER TABLE table_name
CHANGE COLUMN oldcolumn_name
newcolumn_name new_data_type (size);
```

b.) more than one column name and its datatype

syntax :-

```
ALTER TABLE table_name
CHANGE COLUMN old_name newcolumn_name
new_data_type (size),
CHANGE COLUMN old_column_name new_column
new_data_type (size);
```

⇓

It creates two new column with their datatypes in place of old_column.

Page No.
Date / /

B. change column name and its datatype with Constrain

Syntax :- ALTER TABLE table_name
CHANGE COLUMN old_column new_column datatype (size) constraint_name,
ADD constraint_name (column_name);

eg :- ALTER TABLE my_tab
CHANGE COLUMN roll id int(5) NOT NULL,
ADD PRIMARY KEY (new column_name);

Here, my_tab is table_name.
roll is old_column_name.
id is new_column_name
NOT NULL and PRIMARY KEY ARE constraints.

C. MODIFY COLUMN

a) modify column datatype and its size

:- This is used to modify size of the datatype or the datatype itself of an existing column. (but column_name must be empty)

syntax:-

```
ALTER TABLE table_name
MODIFY COLUMN column_name new datatype(size);
```

Eg:-

```
ALTER TABLE my_tab
MODIFY name char(20);
```

table_name

existing column_name

new datatype

size

D. DROP COLUMN

:- It is used to delete any column in a table.

a) Delete column in a table

syntax :-

```
ALTER TABLE table_name
DROP COLUMN column_name ;
```

Eg:-

```
ALTER TABLE my_tab
DROP COLUMN roll;
```

table_name

column_name

b) Removing constraint from a column

syntax :-

```
ALTER TABLE table_name
DROP constraint_name column_name;
```

Eg :-

```
ALTER TABLE my_tab
DROP UNIQUE KEY (roll);
```

UNIQUE KEY → new constraint_name

(roll) → column_name

E. DROP TABLE

syntax :-

```
DROP TABLE table_name;
```

(It deletes table with their data)

F. TRUNCATE TABLE

:- It is used, when we only want to delete the data inside the table, and not the table itself.

syntax :-

```
TRUNCATE TABLE table_name;
```

DESC table_name :→ It shows table is not deleted ;

SELECT * FROM table_name → It deletes data from tab

Drop table (DROP TABLE)	Truncate table (TRUNCATE TABLE)
• It deletes a table with data.	• It deletes only data.
	• Table exists after Truncate.

RENAME TABLE

:- It is used to rename one or more table.

table :- bholi

Id	name	City
1	Akash	Prayag
2	Sumit	Sasharam

Syntax :-

RENAME TABLE old_tablename to new_tablename

Eg :-

RENAME TABLE bholi to saket.

bholi → old table_name

saket → new table_name

ALTER DATABASE

:→ It is used to change the overall characteristics of the database.

• These characteristics are stored in db.opt file in the database directory.

syntax :- ALTER DATABASE database_name;

DROP DATABASE

:- It is used to delete a database.

syntax :- DROP DATABASE database_name;

Eg :- DROP DATABASE my_db;

database_name.

Here, my_db database is deleted from database list.

SHOW COLUMNS

:- It is just like DESC.

- It shows all the columns of table, their data types and specific details.

syntax :- SHOW COLUMNS FROM table_name

SHOW CREATE DATABASE

:→ It is used to show the command which you have written while creating your Database.

Syntax :- SHOW CREATE DATABASE database_name;

Eg :-

SHOW CREATE DATABASE my_db;

Output :-

Database	CREATE Database
my_db	CREATE DATABASE my_db.

SHOW CREATE TABLE

:→ It shows command which you have written while creating your table

syntax :- [SHOW CREATE TABLE table_name;

Ex :- SHOW CREATE TABLE emp;
↳ table_name

UPDATE Record

:→ It is used to update existing records in a table.

syntax :-

UPDATE table_name
SET column1 = value1, column2 = value2,
WHERE any_column = any value.
↳ must be unique

table :- emp

emp_id	name	city	salary
101	Saket	Delhi	80,000
102	Shubham	Chennai	60,000
103	Amit	Mumbai	55,000

Eg:- UPDATE emp.
SET name = 'saket', salary = 86000
WHERE emp_id = 101;

emp_id	name	city	salary
101	Saket	Delhi	86000
102	Shubham	Chennai	60000

any column which must be unique and reference to saket

Here, First of all, it searches emp_id=101 then it updates their name and salary. If we donot use WHERE clause then all salary records are replaced by 86000 and name by saket.

Note:- WHERE is necessary otherwise all records will be replaced by given name and salary.

UPDATE with CASE

Syntax:-
```
UPDATE table_name
SET new_column (which we want to update)
CASE
    WHEN column_name1=some_value1
    THEN new_value1
    WHEN column_name2=some_value2
    ELSE new_value3.
END;
```

Page No.
Date / /

Eg:- UPDATE student → table_name
SET Result =
CASE → column_name (which we want to update)
WHEN mark >= 300 THEN 'First'
WHEN mark < 300 AND mark >= 250 THEN 'Second'
WHEN mark < 250 AND mark >= 150 THEN 'Third'
ELSE 'Fail'
END;

DELETE record

:→ It is used to delete records in a table.

1. DELETE one record

syntax :-

```
DELETE FROM table_name
WHERE column_name = value AND
column_name = value .
```

must be unique

Eg:- DELECT FROM emp
WHERE emp_name = 'Rohan' AND
id = 5 ;

Here, atleast one column must be unique.

2. DELETE all records

syntax :-

```
DELETE FROM table_name;
```

Or

```
DELETE * FROM table_name;
```

⇓

It deletes ~~permanently~~ records permanently from table.

Copy old table to New table

1. Within same database

Syntax :-

```
CREATE TABLE newtable_name LIKE old-table;
&
INSERT new_table SELECT * FROM
                old_table;
```

Eg :-
```
CREATE TABLE teacher LIKE student;
INSERT teacher SELECT * FROM student;
```

⇓ Here, teacher is new_table_name
student is old_table_name.

So, All record of ~~student~~ table from student is copied to new_table teacher.

2. In different database

syntax :-

CREATE TABLE new_table LIKE old database.old table name;

INSERT new_table SELECT * FROM old database.old table name;

Eg:- CREATE TABLE teacher LIKE my_db.student;

new_tablename old_database old table name

INSERT teacher SELECT * FROM my_db.student

FUNCTIONS

- MIN(column_name) → Smallest value of selected column.
- MAX(column_name) → Largest value of selected column.
- sum(column_name) → The total sum of a numerical value of column
- AVG(column_name) → The average value of a numerical column.
- SQRT(column_name) → The square root of a numerical value of this column
- ROUND(column_name, decimal) → Fn is used to round a numerical value.

MIN(column_name)

:- Syntax :-

SELECT MIN(column_name) FROM table_name;

table :- emp

Name	Registration	city	Salary
Saket	11903813	Prayag	90000
Amit	11904652	Sasaram	46000
Sulekh	11903020	Patna	65100

It finds minimum value from salary column.

eg :-

mysql → SELECT MIN(salary) FROM emp;

MIN(salary)
46000

Note :- If we want to rename MIN(salary) then we have to use Alias (AS).

Syntax :- SELECT MIN(salary) AS newsalary FROM emp;

MAX(column_name)

:- Syntax :- SELECT MAX(column_name) FROM table_name;

Eg :- SELECT MAX(salary) FROM emp;

MAX(salary)
90000

It finds ~~minimum~~ maximum value from salary column.

SUM (column_name)

syntax :- SELECT SUM(column_name) FROM table_name;

Eg :- SELECT SUM(salary) FROM emp;

It adds all the numerical value from salary column

SUM(salary)
201100

AVG (column_name)

It displays average value of salary column

syntax :- SELECT AVG(column_name) FROM table_name;

Eg :- SELECT AVG (salary) FROM emp;

AVG(salary)
67033.33

Or SELECT AVG(salary) AS av_salary FROM emp;

It renames AVG(salary) as av_salary.

av_salary
67033.33

SQRT(column_name) → SELECT column, SQRT(column) FROM table-name;

syntax :- SELECT column1, SQRT(column1) FROM tablename;

Eg :- SELECT *, SQRT(salary) AS sroot FROM emp;

Name	Registration	Salary	sroot
Saket	11903813	90000	300.00
Amit	11904652	46000	214.47
Sulekh	11903020	65100	255.15

ROUND(column_name, decimal)

:- syntax :-

```
SELECT *, ROUND(column, decimal)
FROM table_name;
```

table :- products

Id	price	P-name
1	6999.55	Mobile
2	16000.56	Computer
3	10000.23	Laptop

Eg:- SELECT *, ROUND(price, 1) AS approx FROM products.

Id	price	P-name	approx
1	6999.55	Mobile	6999.6
2	16000.56	computer	16000.6
3	10000.23	Laptop	10000.2

COUNT Function

:- • The COUNT(column_name) function returns the no. of values (NULL values will not be counted) of the specified column:

syntax:- [SELECT COUNT(column_name)
FROM table_name;

• COUNT(*) :- It returns all the number of records in a table.

syntax :- [SELECT COUNT(*)
FROM table_name;

• The COUNT(DISTINCT column_name)
:→ It returns the no. of distinct value of the specified column. (how many distinct values are present in a specific value)

syntax :- [SELECT COUNT(DISTINCT column_name)
FROM table_name;

table :- orders

Eg:-

OrderId	CustId	SellerId
1	101	201
2	102	202
3	103	203
4	101	205
5	101	204
6	102	301
7	103	4301

Eg:- SELECT COUNT(CustId)
FROM orders;

⇒ It tells how many customer ID are present in orders table.

COUNT(CustId)
7

Eg:- SELECT COUNT(DISTINCT CustId)
FROM orders;

COUNT(DISTINCT CustId)
3

⇒ It tells how many Distinct CustId are present in orders table

Here; In CustId, there are 3 distinct value 101, 102, 103.
Multiple value is not counted.

- To calculate how many (no.) of orders by each/specific CustId

Syntax :-

SELECT COUNT(CustId) FROM ~~ORDER~~ Orders;
WHERE CustId = 101

COUNT(CustId)
3

UPPER Function

:- It converts the value of a field to Uppercase.

- It can written as UCASE (column_name)

syntax :-

```
SELECT UPPER(column_name).
FROM table_name.
```

Or

```
SELECT UCASE (column_name)
FROM table_name.
```

Eg :-

```
SELECT UPPER(emp_name) AS
Name, city
FROM emp.
```

column name which we want to convert in uppercase.

table_name

column name which we want to convert in ~~fea~~ uppercase

LOWER Function

:- It converts the value of a field to lowercase.

- It can written as LCASE (column_name)

syntax :-

```
SELECT LOWER (column_name)
FROM table_name;
```

Or,

```
SELECT LCASE (column_name)
FROM table_name;
```

Id	city
101	Sasaram
102	Delhi
103	Kolkata

MID Function

:- It is used to extract characters from a text field.

syntax :- SELECT MID(column_name, start, length) FROM table_name;

Or, SELECT MID(city, 1, 3) FROM emp;

→ end with

→ starting from

↓ column-name

⇓ It displays only 3 letters of city_name from city column

Id	city
101	sas
102	Del
103	Kol

Or, SELECT SUBSTRING(city, 1, 3) AS shortcity FROM emp;

LENGTH Function

→ It also calculate space charac etc

:→ It displays the length of the value in a text field.

syntax :- SELECT column_name1, LENGTH(column1) FROM emp;

Eg :- SELECT city, LENGTH(city) FROM emp;

Id	City	LENGTH(city)
101	Sasaram	7
102	Delhi	5
103	Kolkata	7

⇓ It displays total length of value present in specific column

CONCAT Function

:→ It joints two columns.

syntax :-
```
SELECT column1,
CONCAT (column2 , column3)
FROM ~~emp~~ table_name;
```

table :- emp

Id	emp_name	city	salary
108	Dev	Mumbai	30000
109	Jai	Patna	15000
110	Vinay	Chennai	20000
111	Saket	Delhi	25000
112	Bholi	Ranchi	35000

Eg :-
```
SELECT emp_name,
CONCAT (city, salary) FROM emp;
```

~~table :- emp~~

Id	emp_name	city	salary
108			
109			

table :- emp

emp_name	CONCAT(city, salary)
Dev	Mumbai 30000
Jai	Patna 15000
Vinay	Chennai 20000
Saket	Delhi 25000
Bholi	Ranchi 35000

Page No.
Date / /

Eg :-
```
SELECT emp_name,
CONCAT (city, ' ', Salary)
FROM emp;
```

table :- emp

emp_name	CONCAT(city, ' ', Salary)
Dev	Mumbai, 30000
Jai	Patna, 15000
Vinay	Chennai, 20000
Saket	Delhi, 25000
Bholi	Ranchi, 35000

Eg:-
```
SELECT emp_name,
CONCAT (salary, SELECT MID(city, 1, 3)))
As address FROM emp;
```

It extracts only 3 letters starting from 1 to 3 from city column

table :- emp

emp_name	address
Dev	30000 Mum
Jai	15000 Pat
Vinay	20000 Che
Saket	25000 Del
Bholi	35000 Ran.

REVERSE Function

:- It ~~order~~ reverses the order of letter in string of any column.

syntax :- SELECT REVERSE (column-name) FROM column-name;

table :- emp

Id	emp_name	city
108	Dev	Delhi
109	Jai	Mumbai
110	Vinay	Patna
111	Saket	Chennai
112	Bholi	Ranchi

Eg:- SELECT REVESE(city) FROM emp;

REVERSE(city)
ihleD
iabmuM
antaP
iannehC
ihcnaR

NOW Function

→ it displays time and date of each record from table.

:- It returns the current system data and time

syntax :- SELECT column-name,
column_name1,
NOW () AS DateTime
FROM table_name;

Eg:- SELECT emp_name,
city, NOW () AS DateTime
FROM emp;

If there are two or more than two same data having different values then it displays values on the basis of function used with group by.

GROUP BY → Always uses with function

:→ syntax :-

```
SELECT column_name;
MIN(column_name 1)        # Need a function
FROM table_name
GROUP BY column_name;
```

→ same column_name

table :- emp

Id	Name	Dept	salary
185	Dev	IT	45000
189	Raj	HR	32000
201	Dev	Accountant	30000
203	Ram	IT	35480
206	Sunil	HR	62000

Eg :-

```
SELECT Name,        → column_name
       MIN(salary)
FROM emp            → table_name
GROUP BY Name;
```

Function ← MIN(salary)

Name → column_name

Id	Name	Dept.	salary
201	Dev	Accountant	30000
189	Raj	HR	32000
~~203~~	~~Ram~~	~~IT~~	~~35480~~
206	Sunil	HR	62000
203	Ram	IT	35480

⇒ It groups each dept together

Here; In emp table, There are 2 employees whose name is Dev.
But by using, GROUP BY with MIN fn then it displays Dev having minimum salary.

If we use MAX(column_name)

Eg:- SELECT Name,
MAX(salary)
FROM emp
GROUP BY Name;

both will be same. column_name.

Id	Name	Dept	Salary
201	Dev	Accountant	30000
189	Raj	HR	32000
203	Ram	IT	35480
206	Sunil	HR	62000
285	Dev	IT	45000

It displays Dev having maximum salary.

Id	Name	Dept	Salary
189	Raj	HR	32000
206	Sunil	HR	62000
203	Ram	IT	35480
285	Dev	IT	45000

⇓

~~groups each department together~~

It groups employee from same department together. diffn

But, if the ~~same~~ department has same employee name and we want to GROUP BY Name then it displays Name of employee on the basis of function

HAVING Clause

:- It always uses in function.

syntax :-

```
SELECT column_name 1,
Function_name (column_name2)
FROM table_name
GROUP BY column_name1
HAVING Function_name (column_name2) > 
```

same details

Eg :-

```
SELECT Name,
MIN(salary)
FROM emp
GROUP BY Name
HAVING MIN(salary) > 40000;
```

It displays each Name which having each MIN(salary) greater than 40000.

Table – Orders

OrderId	CustId	SellerId
1	101	201
2	102	202
3	103	203
4	101	205
5	101	204
6	102	301
7	103	301

Eg :-

```
SELECT CustId,
count(*)
FROM orders
GROUP BY CustId
```

~~having count(*)>2;~~

It groups all the distinct value of CustId

⇒

CustId	COUNT(*)
101	3
102	2
103	3

Eg :-

```
SELECT CustId,
COUNT(*)
FROM orders
GROUP BY CustId
HAVING COUNT(*)>2;
```

It groups all the distinct value of CustId

It displays which satisfy f^n.

HAVING comes with GROUP BY

function

⇒

CustId	COUNT(*)
101	3
103	3

* Primary key → Uniquely +

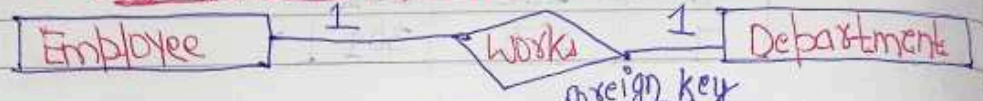
Page No. KRISHNA
Date / /

* Relationship

:→ It is the association between two or more entities.

One to one

- ER model relationship to relationship



~~table employee~~ f.key foreign key

Eid	name	age
E1	A	22
E2	B	25
E3	C	24
E4	A	22

Primary key

Eid	Did
E1	D1
E2	D2
E3	D4
E4	D3

Each Id of employee related uniquely to Did.

Did	Dname	Location
D1	IT	Patna
D2	CSE	Delhi
D3	Civil	Kolkata
D4	ECE	Mumbai

primary key

⇓

Primary key may be either Eid or Did.

For pratical

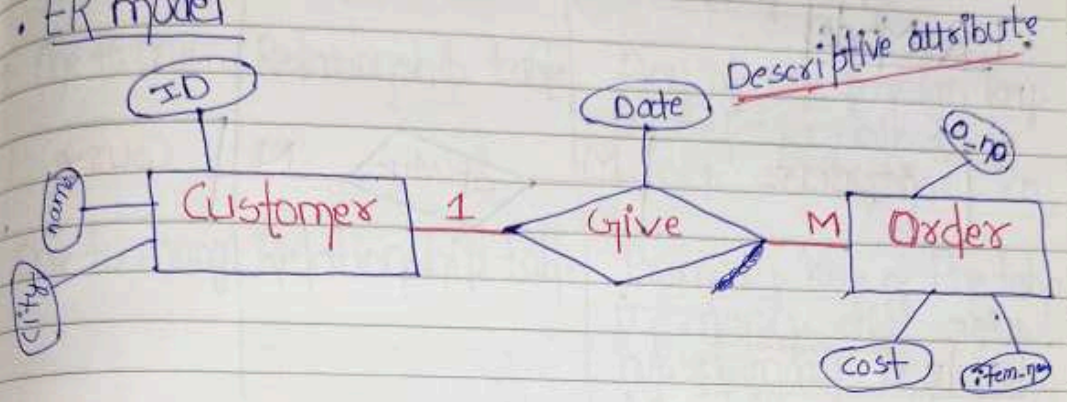
step:-1 Open any database.

step:-2 create 2 tables, one for employee and other for Department.

step:-3 Again, create a table for works and create foreign key with reference of Eid and Did.

One to Many

- ER model



- Relationship table

Note: Relationship table में primary key वही होगा जो many वाले side में primary key होगा।

customer

ID (Primary key)	name	city
C1	A	Jalandhar
C2	B	Patna
C3	C	Delhi
C4	D	Pune

Give (FK, FK)

ID	O_no	Date
C1	O1	26-2
C1	O2	27-2
C2	O3	25-2
C2	O4	21-2

primary key order

O_no	item_no	Cost
O1	Bucket	1,000
O2	shoes	2000
O3	shirt	1500
O4	Jeans	2000

Note :- primary key of relationship table will associate with primary key of Many only

~~But, here,~~ One to Many

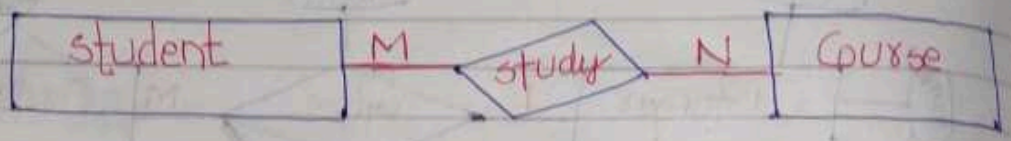
:- One customer can give more than one order so, C1 can give order having O_no O1 and O2.

Similarly, C2 can give order having O_no O3 and O4.

But, One ~~order~~ O_no associate with only One customer.

Many to Many (M-N)

- ER Model



Table

student

Rollno	Name	age
1	A	16
2	B	17
3	A	16
4	C	17
5	D	15

study (FK, FK)

Rollno	C-id
1	C1
2	C2
1	C2
2	C1
3	C3

course

C-id	name	Credit
C1	Maths	4
C2	Phy	4
C3	Chem	4
C4	Hindi	4

Referencing or relationship table

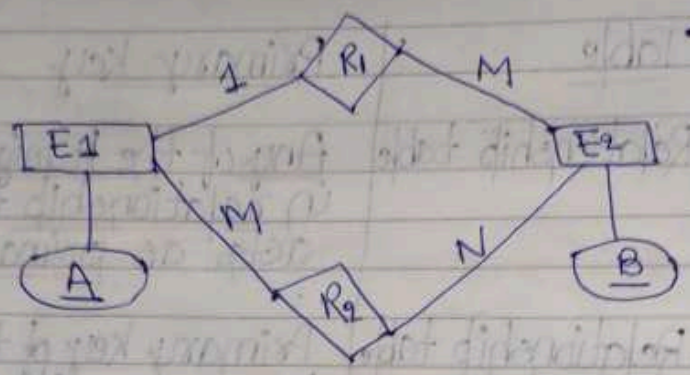
- Primary key in referencing or relationship table :- Primary key will be :-
  composite key of Rollno & C-id.

Here, One course study by many students and each student can study many courses.

Note :-

Cardinality	Table	Primary key
1. One to one	Relationship table	Any of the foreign key in relationship table acts as primary key.
2. One to many	Relationship table	Primary key of the table in many side acts as primary key of relationship or referencing table
3. Many to many	Relationship table	→ composite key of both foreign key of Relationship table acts as primary key.
4. Many to one	Relationship table	Primary key of the table in many side acts as primary key of relationship table.

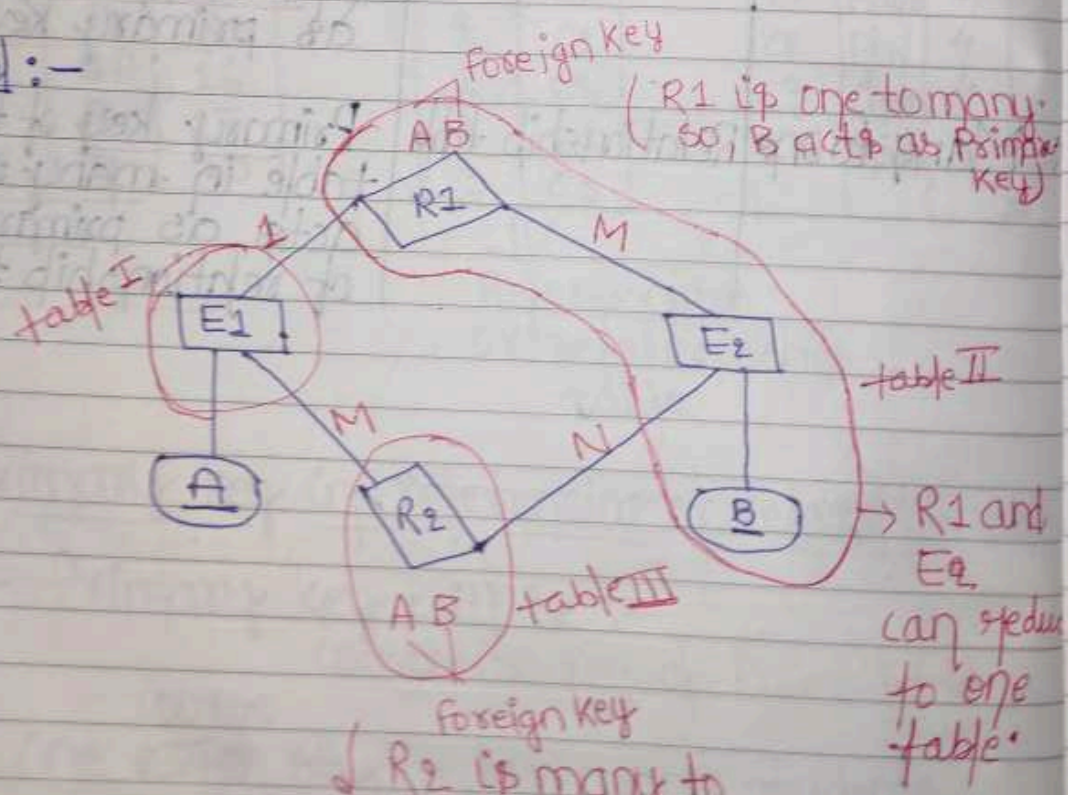
GATE



What is the minimum no. of tables required to represent this ER Model into Relational model?

a.) 2 b.) 3 c.) 4 d.) 5

Sol :-



so, minimum no. of tables required is 3.

table I = E1
table II = R1 E2.
table III = R2.

Ans

Ques Find Nth highest salary using SQL.

syntax :-

```
select column_name1, column_name2
From table_name AS alias_name1
where N-1 = (SELECT COUNT(DISTINCT column2)
             FROM table_name
             AS alias_name2
             where
             alias_name2.column2 >
             alias_name1.column1)
```

Nested SQL

Nth (eg:- For 4th, N=4)

emp e1

ID	Salary
1	10000
2	20000
3	20000
4	30000
5	40000
6	50000

emp e2

ID	salary
1	10000
2	20000
3	20000
4	30000
5	40000
6	50000

FOR 4th highest salary i.e., N=4

Eg:-

```
SELECT ID, salary
FROM emp e1
WHERE 3 = ( SELECT COUNT(DISTINCT salary)
            FROM emp e2
            WHERE e2.salary > e1.salary)
```

⇒ (2, 20000).

~~Unique key~~ FOREIGN KEY

→ A Foreign key in one table points to a primary key in another table (Parent table)

- The primary key used by a foreign key is also known as parent key. The table where the primary key is from is known as parent table.

- The foreign key can be used to make sure that the row in one table have corresponding row in another table.

- Foreign key value can be null, even though primary key value can't.

primary key

EmpId	Name	Address
1	Rani	Delhi
2	Jai	Mumbai
3	Veeru	Kolkata

Foreign key

Dept_Id	DName	Emp_ID
1	IT	1
2	HR	1
3	Admin	2

CREATE FOREIGN KEY

```
CREATE TABLE department
(
D_id int NOT NULL AUTO_INCREMENT PRIMARY KEY,
D_Name varchar (40),
E_id int,
CONSTRAINT employee_Eid_fk
FOREIGN KEY (E_id) REFERENCES employee(E_id)
);
```

new_tablename or child table

column name of parent table

It is optional

old_tablename

~~child table~~ Parent table

Here

- The CONSTRAINT clause allows to define constraint name for the foreign key constraint. It is optional
- The REFERENCES clause specifies the parent table and its columns to which the columns in the child table refer.

Page No. Date / /

- CREATE PARENT TABLE & CHILD TABLE

Parent table

```
CREATE TABLE employee
(
  eid INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  ename VARCHAR (40),
  address VARCHAR (40).
);
```

parent table_name

Child table

```
CREATE TABLE department
(
  did INT NOT NULL AUTO_INCREMENT
      PRIMARY KEY,
  dname VARCHAR (40),
  empid INT NOT NULL,
  CONSTRAINT employee_eid_fk
  FOREIGN KEY (emp_id) REFERENCES
  employee (eid).
```

child table_name.

parent table

~~column~~ primary key of parent table.

>>> DESC department;

Field	Type	NULL	Key	Default	Extra
did	int(11)	No	PRI	NULL	auto_increment
dname	varchar(40)	YES		NULL	
empid	int(11)	No		NULL	

DROP FOREIGN KEY

syntax :- ALTER TABLE table_name
DROP FOREIGN KEY parent_table_column_fk
constraint name.

Eg :- ALTER TABLE department
DROP FOREIGN KEY employee_eid_fk;
Constraint name

Add Foreign key Constraint in a table

:- CREATE TABLE department
(
did int not Null auto_increment primary key,
dname varchar(40),
empid int not null
);

create table

syntax :-
ALTER TABLE department
ADD CONSTRAINT employee_eid_fk
FOREIGN KEY (empid) REFERENCES
employee(eid);

table-name

~~mysql> SELECT * from INFORMATION_SCHEME~~

mysql> SELECT * from INFORMATION_SCHEME.TABLE_CONSTRAI
WHERE TABLE_NAME = 'department';

CONSTRAINT Name	TABLE_SCHEME	TABLE_NAME	CONSTRAINT Type	
PRIMARY	geek	department	PRIMARY	KEY
PRIMARY	geekyshows	department	PRIMARY	KEY
employee_eid_fk	geekyshows	department	FOREIGN	KEY.

Page No. Krishna
Date / /

Note :- We can't delete Parent table.
If we want to delete records then we can delete records of child table first then we can delete from parent table.

mysql > USE my_db ;
mysql > SHOW TABLES

output :-

Tables in my_db
department
employee

child table
Parent table

mysql > SELECT * FROM employee ;

Output :-

employee

eid	ename	address
1	Rani	Delhi
2	Sumit	Kolkata
3	Rahul	Mumbai
4	Raj	Delhi
5	Jay	Hyd
6	Veeru	Patna

Primary key

Parent table, we cannot delete any records from it

mysql > SELECT * FROM department ;

Output :-

department

did	dname	empid
1	IT	1
2	HR	1
3	Admin	1
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5
10	Admin	6

primary key

Foreign key

child table

mysql > DELETE FROM employee
WHERE eid = 1;

error, cannot delete or update a parent row.

Note :- Here; eid references to empid.
So, we have to delete records from child table first then we can delete records from the parent table.

• Delete emp = 1 from department table.
So, we use did = 1, 2 and 3.

mysql > ~~DELECT~~ DELETE FROM department
where did = 1;
mysql > DELETE FROM department
WHERE did = 2;
mysql > DELETE FROM department
WHERE did = 3;

these syntax delete emp=1

mysql > SELECT * FROM department;

department → child table

did	dname	empid
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5
10	Admin	6

primary key

empid → Foreign key

⇒ empid = 1, all records related to empid=1 are deleted from child table or Referencing table

Now, we can delete the same records from parent table

Date / /

```
mysql> SELECT * FROM employee
       WHERE eid=1;
```

Query OK, row affected.

employee → parent table

eid	ename	address
2	Sumit	Kol
3	Rahul	Mum
4	Raj	Delhi
5	Jay	Hyd
6	Veeru	Patna

Here, eid=1 are deleted now from emplo table.

Note :- If we want to delete records from parent table then we have to delete same records from child table first

On DELETE clause

- → ON DELETE CASCADE
- → ON DELETE SET NULL
- → ON DELETE NO ACTION
- → ON DELETE RESTRICT

• ON DELETE CASCADE

:→ As we know that, if we want to delete records from Parent table then we have to delete records from child table first.

But, with the help of **ON DELETE CASCADE** we can delete the records of Parent table directly and also delete the same record of child table automatically.

Syntax :- DELETE FROM parent_table_name
WHERE column_name ='value';

Eg:- DELETE FROM employee
WHERE eid = 1;

Note:- We have to add ON DELETE CASCADE during the creation of tables.

compulsory

On DELETE clause

- ON DELETE CASCADE
- ON DELETE SET NULL
- ON DELETE NO ACTION
- ON DELETE RESTRICT

• ON DELETE CASCADE

:→ As we know that, if we want to delete records from Parent table then we have to delete records from child table first.

But, with the help of ON DELETE CASCADE we can delete the records of Parent table directly and also delete the same record of child table automatically.

Syntax :- DELETE FROM parent_table_name
WHERE column_name = 'Value';

Eg:- DELETE FROM employee
WHERE eid = 1;

Note:- We have to add ON DELETE CASCADE during the creation of tables.

On DELETE clause

- → ON DELETE CASCADE
- → ON DELETE SET NULL
- → ON DELETE NO ACTION
- → ON DELETE RESTRICT

ON DELETE CASCADE

:→ As we know that, if we want to delete records from parent table then we have to delete records from child table first.

Situation, the help of ON DELETE CASCADE we can delete the records of parent table directly and also delete the records of child table automatically.

Syntax :- DELETE FROM parent_table_name WHERE column_name = 'value';

Eg:- DELETE FROM employee WHERE id=1;

Note:- We have to add ON DELETE CASCADE during the creation of table.

On DELETE clause

- → ON DELETE CASCADE
- → ON DELETE SET NULL
- → ON DELETE NO ACTION
- → ON DELETE RESTRICT

• ON DELETE CASCADE

:→ As we know that, If we want to delete records from parent table then we have to delete records from child table first.

But, with the help of ON DELETE CASCADE we can delete the records of parent table directly and also delete the record of child table automatically.

Syntax :- DELETE FROM parent_table_name
WHERE column_name = 'value';

Eg:- DELETE FROM employee
WHERE eid = 1;

Note :- We have to add ON DELETE CASCADE during the creation of table.

compulsory

On DELETE clause

- ON DELETE CASCADE
- ON DELETE SET NULL
- ON DELETE NO ACTION
- ON DELETE RESTRICT

• ON DELETE CASCADE

:→ As we know that, if we want to delete records from Parent table then we have to delete records from child table first.

But, with the help of **ON DELETE CASCADE** we can delete the records of Parent table directly and also delete the same record of child table automatically.

Syntax :- DELETE FROM parent_table_name
WHERE column_name = 'value';

Eg :- DELETE FROM employee
WHERE eid = 1;

Note :- We have to add ON DELETE CASCADE during the creation of tables.

compulsory

On DELETE clause

- → ON DELETE CASCADE
- → ON DELETE SET NULL
- → ON DELETE NO ACTION
- → ON DELETE RESTRICT

• ON DELETE CASCADE

:→ As we know that, if we want to delete records from Parent table then we have to delete records from child table first.

But, with the help of ON DELETE CASCADE we can delete the records of Parent table directly and also delete the same record of child table automatically.

Syntax :- DELETE FROM parent_table_name
WHERE column_name = 'value';

Eg :- DELETE FROM employee
WHERE eid = 1;

Note :- We have to add ON DELETE CASCADE during the creation of tables.

compulsory

On DELETE clause

- → ON DELETE CASCADE
- → ON DELETE SET NULL
- → ON DELETE NO ACTION
- → ON DELETE RESTRICT

• ON DELETE CASCADE

:→ As we know that, if we want to delete records from Parent table then we have to delete records from child table first.

But, with the help of ON DELETE CASCADE we can delete the records of Parent table directly and also delete the same record of child table automatically.

Syntax :- DELETE FROM parent_table_name
WHERE column_name = 'value';

Eg :- DELETE FROM employee
WHERE eid = 1;

Note :- We have to add ON DELETE CASCADE during the creation of tables.

compulsory

On DELETE clause

- → ON DELETE CASCADE
- → ON DELETE SET NULL
- → ON DELETE NO ACTION
- → ON DELETE RESTRICT

• ON DELETE CASCADE

:→ As we know that, if we want to delete records from Parent table then we have to delete records from child table first.

But, with the help of ON DELETE CASCADE we can delete the records of Parent table directly and also delete the same record of child table automatically.

Syntax :- DELETE FROM parent_table_name
WHERE column_name = 'Value';

Eg :- DELETE FROM employee
WHERE eid = 1;

Note :- We have to add ON DELETE CASCADE during the creation of tables.

compulsory

On DELETE clause

- → ON DELETE CASCADE
- → ON DELETE SET NULL
- → ON DELETE NO ACTION
- → ON DELETE RESTRICT

• ON DELETE CASCADE

:→ As we know that, if we want to delete records from Parent table then we have to delete records from child table first.

But, with the help of ON DELETE CASCADE we can delete the records of Parent table directly and also delete the same record of child table automatically.

Syntax :- DELETE FROM parent_table_name
WHERE column_name = 'value';

Eg :- DELETE FROM employee
WHERE eid = 1;

Note :- We have to add ON DELETE CASCADE during the creation of tables.

compulsory

On DELETE clause

- → ON DELETE CASCADE
- → ON DELETE SET NULL
- → ON DELETE NO ACTION
- → ON DELETE RESTRICT

• ON DELETE CASCADE

:→ As we know that, if we want to delete records from Parent table then we have to delete records from child table first.

But, with the help of ON DELETE CASCADE we can delete the records of Parent table directly and also delete the same record of child table automatically.

Syntax :- DELETE FROM parent_table_name
WHERE column_name = 'value';

Eg:- DELETE FROM employee
WHERE eid = 1;

Note :- We have to add ON DELETE CASCADE during the creation of tables.

compulsory

On DELETE clause

- → ON DELETE CASCADE
- → ON DELETE SET NULL
- → ON DELETE NO ACTION
- → ON DELETE RESTRICT

• ON DELETE CASCADE

:→ As we know that, if we want to delete records from Parent table then we have to delete records from child table first.

But, with the help of ON DELETE CASCADE we can delete the records of Parent table directly and also delete the same record of child table automatically.

Syntax :- DELETE FROM parent_table_name
WHERE column_name = 'value';

Eg :- DELETE FROM employee
WHERE eid = 1;

Note :- We have to add ON DELETE CASCADE during the creation of tables.

compulsory

Eg:- mysql> SHOW TABLES

Tables
employee

mysql> SELECT * FROM employee;

eid	ename	address
1	Rani	Delhi
2	Sumit	Kol
3	Sonam	Hyd
4	Priti	Mum
5	Kunal	Kol

```
mysql> CREATE TABLE department
(
did int not null auto_increment
                primary key,
dname varchar(40),
empid int not null,
CONSTRAINT employee_eid_fk
FOREIGN KEY (empid) REFERENCES
                employee (eid)
* ON DELETE CASCADE
);
```

```
mysql> INSERT INTO department (did, dname, empid)
-> VALUE (1,'IT',1),
(2,'HR',2),
(3,'Accountant',3),
(4,'IT',4),
(5,'HR',5);
```

mysql> SELECT * FROM department;

did	dname	empid
1	IT	1
2	HR	2
3	Accountant	3
4	IT	4
5	HR	5

mysql> DELETE FROM employee → parent table
WHERE eid=1;

It deletes records of parent table as well as child table.

eid	ename	address
2	Sumit	Kol
3	Sonam	Hyd
4	Priti	Mum
5	Kunal	Kol

mysql> DELETE FROM department; → child table
~~WHERE~~

did	dname	empid
2	HR	2
3	Accountant	3
4	IT	4
5	HR	5

Here, record correspondence of empid=1 deletes from child table also.

Page No.
Date / /

ON DELETE SET NULL

:→ With the help of ON DELETE CASCADE, we can delete the records of parent table and child table automatically.

- ON DELETE CASCADE, it deletes all the (whole row) data related to Foreign key.

But, if we want to delete the particular details of a record from parent table only then the ~~whole row are~~ same details are removed from child table only.

- It doesnot delete whole row.
- It sets NULL VALUE at the place from where data is deleted from child table.

Eg:- mysql> SELECT * FROM employee; → parent table

employee

eid	ename	address
1	Rani	Delhi
2	Sumit	Kol
3	Sonam	Hyd
4	Priti	Mum
5	Kunal	Kol

mysql> CREATE TABLE department → child table.
(
did int not null auto_increment primary key,
dname varchar (40),
empid int,
constraint employee_eid_fk
foreign key (empid) references employee(eid)
* ON DELETE SET NULL
);

CASCADE,
parent table
es all the
particular
arent table
name details
e only.
nce from
able.
parent table
employee;

ild table.

primary key,

mployee(eid)

```
mysql> INSERT INTO department (dname, empid)
       VALUES ('IT', 1),
              ('HR', 1),
              ('Admin', 1),
              ('IT', 2),
              ('Exe', 2),
              ('IT', 3),
              ('Exe', 3),
              ('IT', 4),
              ('HR', 5);
```

did auto_increment

department

auto_increment

did	dname	empid
1	IT	1
2	HR	1
3	Admin	1
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

~~mysql > DELETE * FROM~~

```
mysql > DELETE FROM employee
        Where eid=1;
```

eid	ename	address
2	Sumit	Kol
3	Sonam	Hyd
4	Priti	Mum
5	Kunal	Kol

mysql> SELECT * FROM department;

did	dname	empid
1	IT	NULL
2	HR	NULL
3	Admin	NULL
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

Records do Not deleted from child table, only add NULL value.

When we delete records corresponding to empid=1 then it ~~records~~ deletes data from parent table and add NULL in child table

ON DELETE NO ACTION / ON DELETE NO ACTION ON parent

:-

mysql> SELECT * FROM employee;

eid	ename	address
1	Rani	Del
2	Sumit	Kol
3	Sonam	Hyd
4	Priti	Mum
5	Kunal	Kol

Create child table with ON DELETE NO ACTION

```
mysql> CREATE TABLE department
(
    did int not null auto_increment primary key,
    dname varchar (40),
    empid int,
    constraint employee_eid_fk
    foreign key(empid) references employee(eid)
*   ON DELETE RESTRICT
);
```

```
mysql> INSERT INTO department (dname, empid)
VALUES ('IT', 1),
       ('HR', 1),
       ('Admin', 1),
       ('IT', 2),
       ('Exe', 2),
       ('IT', 3),
       ('Exe', 3),
       ('IT', 4),
       ('HR', 5);
```

```
mysql > SELECT * FROM department;
```

did	dname	empid
1	IT	1
2	HR	1
3	Admin	1
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

parent table

mysql> DELETE FROM employee
Where eid = 1;

Error : cannot delete or update a parent row

so; At first, we have to delete child table first and then delete the respective parent table

mysql> DELETE FROM department
where did=1;

mysql> DELETE FROM department
Where did=2;

mysql> DELETE FROM department
Where did=3;

deleted child table correspondence empid=1

child table

mysql> SELECT * FROM department;

did	dname	empid
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

Now, delete parent table

Page No. KRISHNA
Date / /

```
mysql> DELETE FROM employee
       where eid=1;
```

did	dname	address
2	Sumit	Kol
3	Sonam	Hyd
4	Priti	Mum
5	Kunal	Kol

ON UPDATE Clause

- ON UPDATE CASCADE
- ON UPDATE SETNULL

UPDATE IN PARENT TABLE FIRST

- ON UPDATE NO ACTION
- ON UPDATE RESTRICT

UPDATE IN CHILD TABLE FIRST

ON UPDATE CASCADE

:→ It is used to update the records of parent as well as child table.

- If we update any records ~~from~~ in parent table then the same records get updated from child table automatically.

Syntax :- ON UPDATE CASCADE (add at the end of creation of table).

mysql> SELECT * FROM employee; → Parent table

eid	ename	Address
1	Rani	Del
2	Sumit	Kol
3	Sonam	Mum
4	Priti	Hyd
5	Kunal	Kol

mysql> CREATE TABLE department → child table
(did int not null auto_increment primary key,
dname varchar(40),
empid int,
CONSTRAINT employee_eid_fk,
FOREIGN KEY (empid) REFERENCES employee(eid)
* ON UPDATE CASCADE
);

mysql > INSERT INTO department (dname, empid)
VALUES ('IT', 1),

('HR',1),
('Admin', 1),
⋮
('HR', 5);

mysql > SELECT * FROM department; → child table

did	dname	empid
1	IT	1
2	HR	1
3	Admin	1
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

mysql > UPDATE employee → parent table
SET eid = 10
WHERE eid = 1;

{ It updates the value of eid as 10

eid	ename	Address
2	Sumit	Kol
3	Sonam	Mum
4	Priti	Hyd
5	Kunal	Kol
10	Rani	Del

mysql > ~~SELECT * FROM department;~~

did	dname	empid
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5
10	IT	
11	IT	
12	IT	

mysql> SELECT * FROM department;

did	dname	empid
1	IT	10
2	HR	10
3	Admin	10
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

values updated

ON UPDATE SET NULL

:- It ~~Rules~~ updates the records in parent table and puts ~~SET~~ NULL value in Updated records in child table.

mysql> SELECT * FROM employee; → parent table

eid	ename	Address
1	Rani	Del
2	Sumit	Kol
3	Sonam	Mum
4	Priti	Hyd
5	Kunal	Kol

nt;

values updated

parent
ue in
.

ent table

→ child table

```
mysql> CREATE TABLE department
       (
        did int not null auto_increment primary key,
        dname varchar(40),
        empid int,
        CONSTRAINT employee_eid_fk,
        FOREIGN KEY (empid) REFERENCES
                            employee eid
       * ON UPDATE SET NULL
        );
```

```
mysql> INSERT INTO department(dname, empid)
       VALUES ('IT', 1),
       ('HR', 1),
       ('Admin', 1),
       ('IT', 2),
       ('Exe', 2),
       ('IT', 3),
       ('Exe', 3),
       ('IT', 4),
       ('HR', 5);
```

```
mysql> ~~SELECT~~ SELECT * FROM department;
```

did	dname	empid
1	IT	1
2	HR	1
3	Admin	1
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

Page No.
Date / /

```
mysql > UPDATE employee
        SET eid = 10
        WHERE eid = 1;
```

eid	ename	Address
2	Sumit	Kol
3	Sonam	Mum
4	Priti	Hyd
5	Kunal	Kol
10	Rani	Del

```
mysql > SELECT * FROM department;
```

did	dname	empid
1	IT	NULL
2	HR	NULL
3	Admin	NULL
4	IT	2
5	Exe	2
6	IT	3
7	Exe	3
8	IT	4
9	HR	5

• It is used when two or more columns have same records.

Composite Key

• The combination of two or more columns make primary key.

:→ It is a type of primary key which is created by the composition of two or more columns.

syntax :- PRIMARY KEY (column_1, column_2);

Eg:- CREATE TABLE course → table_name

composite key [coursecode varchar (40),
Date date,
Cname varchar (40),
Seat int,
Remain int,
Room int,
Rcapacity int,
* PRIMARY KEY (coursecode, Date)

mysql > DESC course;

Field	Type	NULL	Key	Default.
Coursecode	varchar (40)	NO	PRI	NULL
Date	date	NO	PRI	NULL
Cname	varchar (40)	YES		NULL
Seat	int (11)	YES		NULL
Remain	int (11)	YES		NULL
Room	int (11)	YES		NULL
Rcapacity	int (11)	YES		NULL

If two tables have same column-name then we use shorthand notation to identify from which table, the same columns belong to.

Shorthand notations

Student

Rollno	Name	Branch
1	Jay	Computer science
2	Suhani	Electronic and Comm
3	Kriti	Electronic and Comm.

Exam

Rollno	s_code	Marks	P_code.
1	CS11	50	CS
1	CS 12	60	CS
1	CS 13	55	CS
1	CS 14	70	CS
2	EC 101	66	EC
2	EC 102	70	EC
2	EC 103	50	EC
3	EC 101	45	EC
3	EC 102	50	EC.

Here, both tables student and Exam have same column Rollno.

~~SELECT Rollno, Name~~

If there is same column then RDMS unable to detect from which table, the same column belongs to.
So, we use shorthand notation for fully qualified notation of any column.

syntax :- table_name . column_name

Eg :- student . Rollno # RDMS easily detect this Rollno belongs to student table

Eg :- Exam . Rollno # This Rollno belongs to Exam table.

JOINS

Types :- CROSS JOIN, Natural JOIN | Equi JOIN, Self JOIN | Outer JOIN (Left, Right, Full)

CROSS JOIN

:- The cross join returns every row from one table crossed with every row from the second.

Employee

Empid	Name	City
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum

Department

did	DepName
101	IT
102	HR
103	Admin

syntax :- SELECT * FROM Employee CROSSJOIN Department;

Or,

SELECT * FROM Employee, Department;

Or,

SELECT Name, Depname FROM Employee CROSSJOIN Department;

Eg:- SELECT name, Depname from department, employee

Name	Depname
Rahul	IT
Rahul	HR
Rahul	Admin
Krish	IT
Krish	HR
Krish	Admin
Jay	IT
Jay	HR
Jay	Admin

Eg:- SELECT * FROM Department, Employee;

Employee

Empid	Name	City
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum

Department

did	DepName
101	IT
102	HR
103	Admin

Eg:- SELECT * FROM Employee, Department;

start from department

Empid	Name	City
1	Rahul	Delhi
2	Krish	Kol
3	Jay	Mum

did	DepName
101	IT
102	HR
103	Admin

mysql> SELECT * FROM Department, Employee;

did	DepName	Empid	Name	City
101	IT	1	Rahul	Delhi
102	HR	1	Rahul	Delhi
103	Admin	1	Rahul	Delhi
101	IT	2	Krish	Kol
102	HR	2	Krish	Kol
103	Admin	2	Krish	Kol
101	IT	3	Jay	Mum
102	HR	3	Jay	Mum
103	Admin	3	Jay	Mum

Self JOIN

:→ A table is joined itself.

Eg:- Find student id who is enrolled in at least two courses.

study

s_id	c_id	Since
S1	C1	2016
S2	C2	2017
S1	C2	2017

JOIN
= Cross product
+ some condition

Page No. Date / /

Apply, concept of self Join

T1

s_id	c_id	since
S1	C1	2016
S2	C2	2017
S1	C2	2017

T2

s_id	c_id	since
S1	C1	2016
S2	C2	2017
S1	C2	2017

Self Join :-

T1		T2	
S1	C1	S1	C1
S1	C1	S2	C2
S1	C1	S1	C2
S2	C2	S1	C1
S2	C2	S2	C2
S2	C2	S1	C2
S1	C2	S1	C1
S1	C2	S2	C2
S1	C2	S1	C2

Cross product

```
mysql> SELECT T1.S_id FROM
       STUDY AS T1,
       STUDY AS T2
       WHERE
       T1.S_id = T2.S_id
       AND
       T1.C_id <> T2.C_id
```

CROSS PRODUCT

condition

Not equal

Page No.
Date / /

Acc. to condition

T1		T2		
~~S1~~	~~C1~~	~~S1~~	~~C1~~	
~~S1~~	~~C1~~	~~S2~~	~~C2~~	
S1	C1	S1	C2	satisfy condition
~~S2~~	~~C2~~	~~S1~~	~~C1~~	
~~S2~~	~~C2~~	~~S2~~	~~C2~~	
~~S2~~	~~C2~~	~~S1~~	~~C2~~	
S1	C2	S1	C1	satisfy condition.
~~S1~~	~~C2~~	~~S2~~	~~C2~~	
~~S1~~	~~C2~~	~~S1~~	~~C2~~	

so, $\boxed{\text{Output :- S1}}$

Equi JOIN

Q→ Find the Emp_name who wanted in a department having location same as Address ?

primary Key

Emp

E_no	E_name	Address
1	Ram	Delhi
2	Varun	chd
3	Ravi	chd
4	Amit	Delhi

Dept

foreign Key

DepNo	Location	E_no
D1	Delhi	1
D2	Pune	2
D3	Patna	4

sol:- JOIN = CROSS PRODUCT + condition.

SELECT E_name
FROM Emp, Dept

Emp.E_no = Dept.E_no
AND
Emp.Address = Dept.Location

mysql > SELECT E_name FROM Emp, Dept] CROSS PRODUCT

condition [WHERE
Emp.E_no = Dept.E_no
AND
Emp.Address = Dept.Location.]

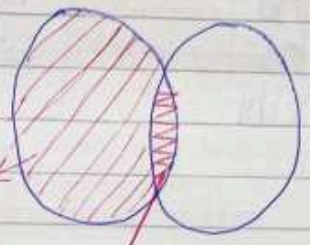
Left outer join

:→ It gives the matching rows and the rows which are in ~~table~~ left table but not in right table.

emp

emp_no	e_name	Dept_no
E1	Varun	D1
E2	Amrit	D2
E3	Ravi	D1
E4	Nitin	—

left table



matching rows.

Dept_no	d_name	Loc
D1	IT	Delhi
D2	HR	Hyd
D3	Finance	Pune

```
mysql> SELECT emp_no, e_name, d_name, LOC
       FROM emp → left table-name
       left outer join dept
       On (emp.dept_no = dept.dept_no);
```

condition

soln

matching rows

emp_no	e_name	d_name	LOC
E1	Varun	IT	Delhi
E2	Amrit	HR	Hyd
E3	Ravi	IT	Delhi
E4	Nitin	—	—

Natural JOIN (CROSS PRODUCT + condition)

emp

e-no	e-name	Address
1	Ram	Delhi
2	Varun	Chd
3	Ravi	chd
4	Amrit	Delhi

Dept

dept-no	Name	e-no
D1	HR	1
D2	IT	2
D3	MRKT	3

Ques Find the employee names who are working in a department.

Sol:-

```
SELECT e-name emp, Dept   → Cross product
WHERE
emp.e-no = Dept.e-no.     ] condition
```

Or,

```
SELECT e_name
FROM emp
Natural Join Dept.
```

① Adhaar card, Pancard, VoterId, mobile

Candidate key → Collection of all such keys which can distinguish b/w two entity.

Primary Key extracts from candidate key.

Ques Query the Name of students in table STUDENTS who scored higher than 75 marks. Order your output by the last character of each name. If two or more students have same names ending in the last 3 characters (Robby, Bobby) secondary sort them in ~~descending order~~ Ascending ID.

sol:→

column	Type
ID	integer
Name	string
Marks	Integer

QUERY

```
SELECT Name
FROM STUDENTS
Where Marks > 75
ORDER BY SUBSTR (name, -3, 3), ID;
```

Ques Query the list of name from Employee table in alphabetical order.

sol :-

```
SELECT name
FROM Employee
ORDER BY name ASC;
```

Ques Query the name of employee from Employee table having salary greater than 2000 and have been employee for less than 10 months. Sort your result by ascending employee_id.

→
```
SELECT name
FROM Employee
WHERE salary > 2000 AND months < 10
ORDER BY employee_id.
```

Ques Write a query to find the maximum total earnings to all employees as well as the total number of employees who have max^m total earnings. Then print these values as 2 space-separated integers.

(months * salary)

→
```
select * From
( select months * salary, count(*)
From employee
group by months * salary
order by 1 desc)
where rownum = 1;
```

Ques Query the list of CITY names starting with vowels (a,e,i,o,u) from STATION. Your result cannot contain duplicates.

→

```
SELECT DISTINCT CITY
FROM STATION
WHERE CITY LIKE 'A%' OR
CITY LIKE 'E%' OR CITY LIKE 'I%'
OR CITY LIKE 'O%' OR
   CITY LIKE 'U%';
```

Ques Query the list of CITY names starting with vowels and ending with vowels.

→

```
SELECT distinct city from station
where (city LIKE 'A%' or city LIKE 'E%'
OR city LIKE 'I%' OR city LIKE 'O%'
OR city LIKE 'U%') and
(city LIKE '%a' OR city LIKE '%e'
OR city LIKE '%i' OR city LIKE
'%o' OR CITY LIKE '%u');
```

Ques Query the list of city names From station that donot start with vowels.

→ SELECT DISTINCT CITY
FROM STATION
WHERE city NOT LIKE 'A%' and
city NOT LIKE 'E%' and
city NOT LIKE 'I%' and
city NOT LIKE 'O%' and city
NOT LIKE 'U%';

Q→ Query the list of city names from station that donot start with vowel and donot end with vowel.

→ SELECT ~~X FROM~~ DISTINCT CITY
FROM STATION WHERE
(city NOT LIKE 'A%' and city not like 'E%'
and city not like 'O%' and city
not like 'I%' and city not like
'U%') OR
(city NOT like '%A' and city not like '%E'
and city not like '%I' and city
not like '%O' and city not like
'%U');

Database Constraints

:→ A constraint is a rule that the database manager enforces.

Categories of Integrity

i) Domain integrity rules

:- It is concerned with maintaining the correctness of attribute values within relations.

- It ensures the data values inside a database follow defined rules for values, range and format.

ii) Entity integrity

:- It ensures that each row in a table is a uniquely identifiable entity.
We can apply entity integrity to a table by specifying PRIMARY KEY Constraints.

- It can enforced through indexes, UNIQUE constraints and PRIMARY KEY constraints
- The primary key for a row is unique, it does not match the primary key of any other row in the table.
- The primary key is not NULL.
- The uniqueness property ensures that the primary key of each row uniquely identify, there are no duplicate

iii) Referential Integrity

:→ It concerned with maintaining the correctness of relationship between relations.

- It is the state of the database in which all values of all foreign keys are valid.
- We can apply referential integrity using a Foreign KEY constraints.

Note :-

A table in which a referential ~~tabl~~ constraint and a foreign key are defined is called referencing table while a table is referenced from a referencing table with a foreign key is called a referenced table.

Rules :-

1. We can't delete a record from a primary table if matching records exist in a related table.
2. We can't change a primary key value in the primary table, if that record is related records.