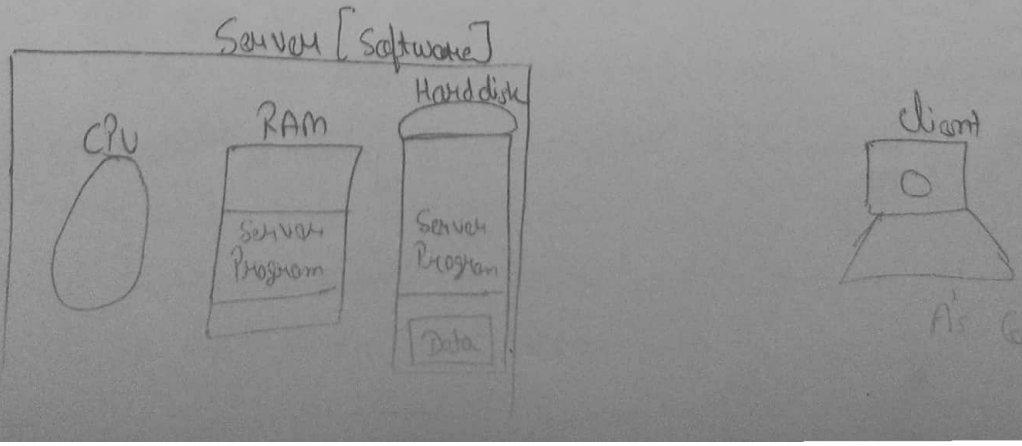
DATABASE MANAGEMENT SYSTEM

★ Over-View :-

- **Data** :- Any fact that can be stored or recorded, like :- audio file, videofile.

Now, where this data is stored, it is stored in Harddisk (i.e. Database, (i.e. Collection of data)).

- **Database Managament System** :- To understand this, first we have to learn about internet, See we have to two things, Server and client.



Now, Server :- Computer which have Server programs (Softwares) in it which Serves millions of clients.

Client :- Computer which have client programs installed in it.
Eg :- Computer, Laptop, etc.

• Client can be Chrome, firefox, etc. also, which requests for the web page or data to show.

Now, We will understand with an example, Suppose, there is client which is surfing our website and it requests for the web page. the Server, where we hosted our website, then from the Server which has Server programs in the RAM/ Harddisk will run and give appropriate Web page to the client. Now if client requests for an audio/video file then Data from the Database for that particular video will be given to the client, so he/she can see it, So these programs (or Softwares) which is used to Construct, manipulate, Delete the Data in Database is called Database Management System.

• DataBase Systems :- Program + Data
↓
DBMS Code

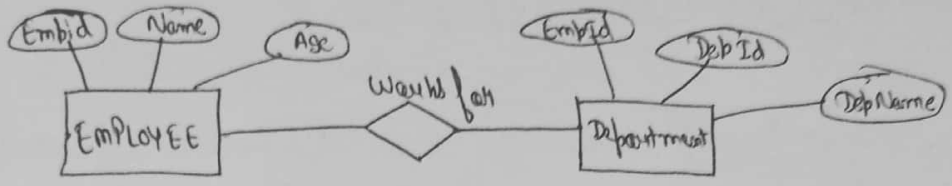
★ Data Model :- It is used to represent the structure of data in database

→ ER-Model → Diagramatical Representation

→ Relational Model → Represented using a Collection of tables

⇒ Important Terminologies :- Table (or Relation)
Row (or Record)
Column (or Field)

Ex :- For ER Model



This model shows how our data present in database or what we actually do with our data by help of Diagrams.

Ex :- For Relational Model

Employee

EmpId	Name	Age
100	Ram	25
101	Sam	35
102	⋮	⋮
⋮	⋮	⋮
185	Bab	45

Department

EmpId	DepId	DepName
100	Ram	Yes
101	Bab	Yes
⋮	⋮	⋮
106	Sahil	No

This model show the Data in form of tables with values and tables are also known as relation, thats why it is known as relational model.

Note :- (1) See these both [redacted] models are meaningful and we can convert one form to another easily.

(2) Both are having Advantages and disadvantages, like Relational model represents more data but ER model is easy to understand, like Employee has EmpId, Name and Age and works for Department having EmpId, DepId and Dep Name.

(3) Like, if i designing a database for School and i have to show it to principal, then i will just show ER model rather than Relational model, bcz it will have large no. of records and difficult to understand.

☆ E-R Model :-

It is basically used to represent the structure of our database by diagrams.

* Entity → objects
* Relationship → Association b/w Entities
* Attributes → Properties of Entities

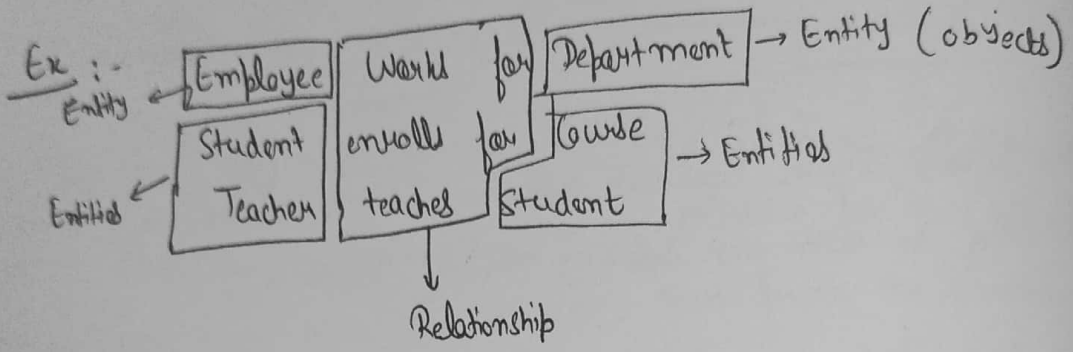
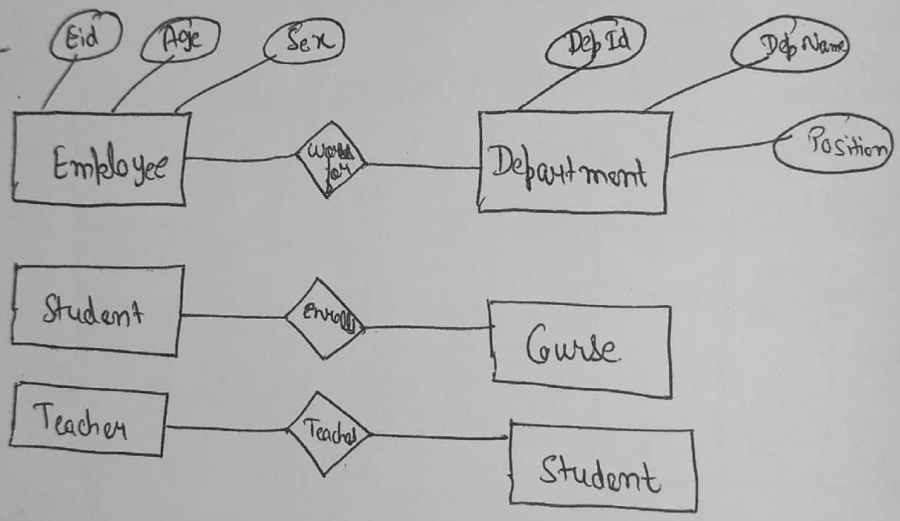
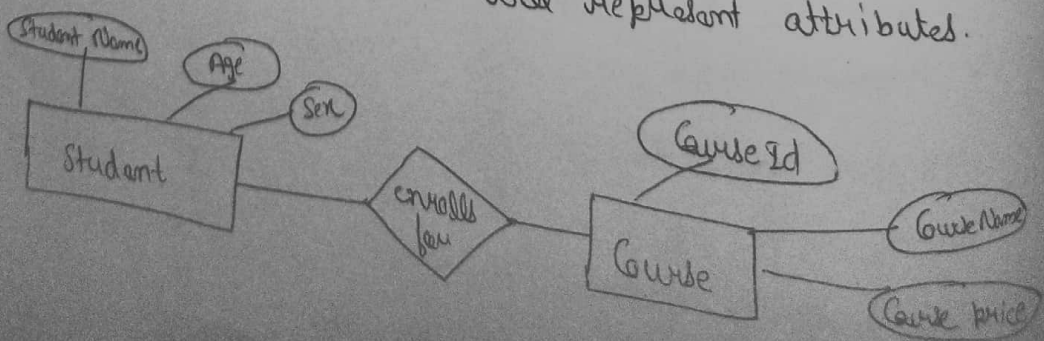


Diagram :-



- Now, Square boxes are used to represent the entity.
- Diamond is used to represent the relation b/w Entities.
- We can say, Table Names will represents the entities/objects.
- Now, Attributes can be represented by using circle/oval shape.
- We can say Columns in table will represent attributes.

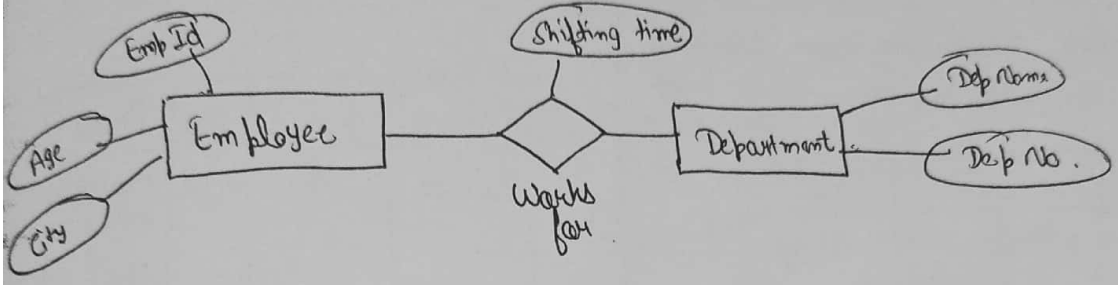


Attributes :- Properties of an Entity (or) Relationship.

Ex-1 :- Entity → Student
Attributes → Roll No., Name, Age, city.

Ex-2 :- Entity → Bookshop
Attributes → Bookshop_Name, address, city.

Ex-3 :- Employee ⟹ Entity
Attributes → Emp Id, Emp Name, Age



Like, Sam Shifting time for accounts is 9:00 AM to 10:00 PM
↓
attribute

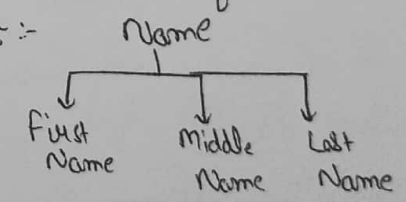
⇒ **Classification of Attributes** :-

① Simple Attributes vs. Composite Attributes

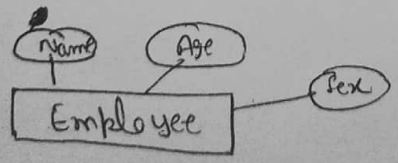
Simple Attributes ⇓ Cannot be divided further

Composite Attributes ⇓ Can be divided further

Ex :-



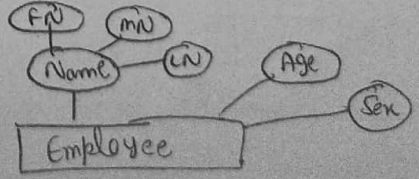
★ ER-model :-



★ Relational :-

Name	Age	Sex

★ E-R model :-



★ Relational :-

Name			Age	Sex
FN	MN	LN		

(2) Single Valued Attribute vs. Multi Valued Attribute

Only one value of attribute is possible for an entity.

Ex :- Age attribute for an Employee entity., Age can be once only, bcz a person can have only one Age no., we can make it multi valued also, depending upon database designer but it will be meaningless though.

Name	Sex	Age
Prince	male	18 yrs

More than one value of attribute is possible for an entity.

Ex :- Customer having more than one Contact no. / Customer having more than one Newspaper from News paper Agency.

Name	Sex	Age	Phone
Prince	male	18 yrs	8365150[illegible] 94[illegible]

Note :- (1) A Simple Attribute can be a Combined with or related with Single valued Attribute like, Age, A person cannot have more than one Age attribute and Age can't be divided further.

(2) A Simple Attribute can be related with Multi Valued Attribute like Contact No., A person can have multiple Contact no's and Contact no cannot divide further.

(3) A Composite Attribute can be related with Single valued Attribute like Name, A person cannot have more than one name (depending upon designer) and Name can be divided further into FN, MN, LN.

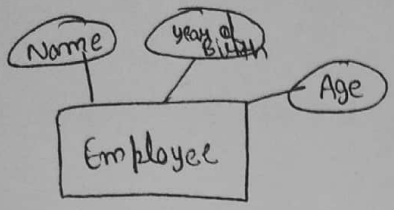
(4) A Composite Attribute can be related with Multi valued Attribute like Name, A person may have Alias name also (depending upon designer) and Name can be divided further into FN, MN, LN.

* These 4 attributes are inter-related with each other.

(3) Stored Attributes Vs. Derived Attributes

We can only store the values of this attribute

We can derive the values of these attribute from stored attributes.



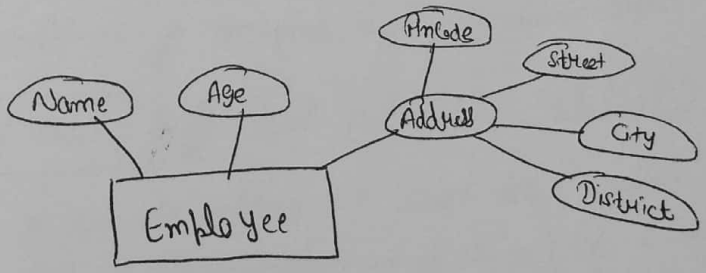
Name	YoB	Age
Bob	1992	2021-1992
Sam	2005	2021-2005

Age = (Current year - YoB)

Here, Year of Birth is Stored Attribute, bez we have to store it first to calculate Age and we can calculate Age but substract YoB from current year, Hence, Age is Derived Attribute.

(4) Complex Attributes :- An attribute which is both Composite and Multi-Valued.

Example :-

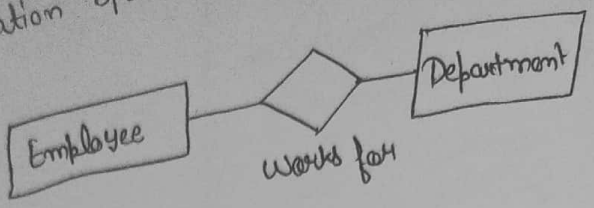


An Employee have Name, Age and Address, And Address can be further divided into pinCode, Street No., City, District, Hence it is Composite Attribute and an Employee can have more than one Home | Residance, So, Address can have multiple values, Hence it is multi-Valued Attribute, Hence known as a Complex Attribute.

Note :- All these things are depending upon the designer of our database.

☆ Relationship :- Association b/w Entities.

Ex :-



⇒ Degree of a Relationship :- Number of Entities participating in a relationship.

- For above Example, Degree = 2

⇒ Cardinality Ratio :- It is defined b/w Entities of Relationship or we can say, it is the ratio of maximum no. of relationships an entity can participate in.

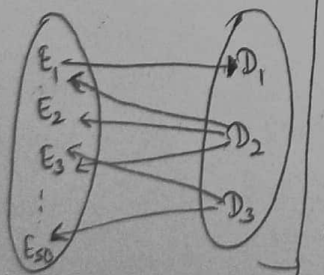
To understand this, we must know about Requirement Analysis,

Requirement Analysis means, the Conditions applied during Designing of Database. Take an Example of Employee and Department.

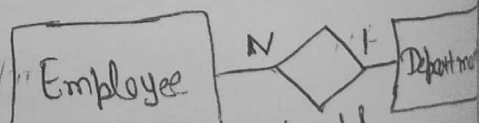
- An Employee can work for atmost one Department.
- A Department can have any no. of Employees.

Now, Calculate Cardinality Ratio of works for Relationship.

now, Suppose, we have 50 employees and 3 departments and we will both as a Set (as we do in mathematics) ; E = {E1, E2, ... E50} ; D = {D1, D2,



Acc to Condition, E_1 or E_2 or any can work for any one(1) Department and in similar way, Department can have any no. of E (N)



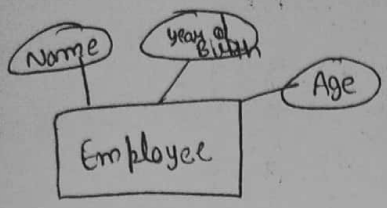
where, N = No. of Emp

- So Cardinality Ratio of Employee and Department ⇒ N : 1
- So Cardinality Ratio of Department and Employee ⇒ 1 : N

Stored Attributes Vs. Derived Attributes

We can only store the values of this attribute

We can derive the values of these attribute from stored attributes.



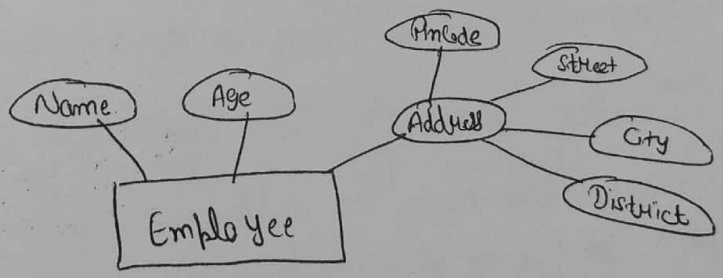
Name	YOB	Age
Bob	1992	2021-1992
Sam	2005	2021-2005

Age = (Current year − YOB)

Here, Year of Birth is Stored Attribute, bcz we have to store it first to calculate Age and we can calculate Age but substract YOB from current year, Hence, Age is Derived Attribute.

(4) Complex Attributes :- An attribute which is both Composite and Multi-Valued.

Example :-

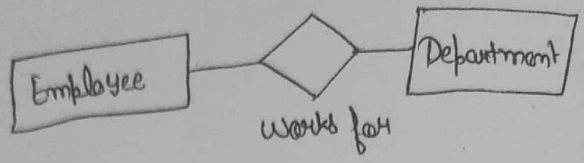


An Employee have Name, Age and Address, And Address can be further divided into pinCode, Street No., City, District, Hence it is Composite Attribute and an Employee can have more than one Home | Residance, So, Address can have multiple values, Hence it is multi-valued Attribute, Hence known as a Complex Attribute.

Note :- All these things are depending upon the designer of our database.

☆ Relationship :- Association b/w Entities.

Ex :-



⇒ Degree of a Relationship :- Number of Entities participating in a relationship.

- For above Example, Degree = 2

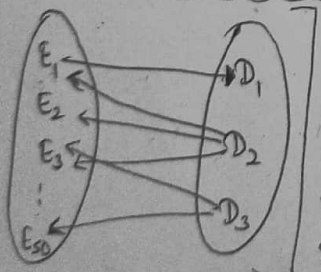
⇒ Cardinality Ratio :- It is defined b/w Entities of Relationship or we can say, it is the ratio of maximum no. of relationships an entity can participate in.

To understand this, we must know about Requirement Analysis, Requirement Analysis means, the Conditions applied during Designing of Database. Take an Example of Employee and Department.

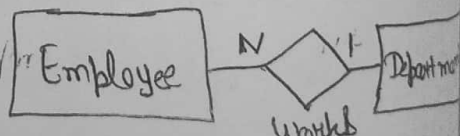
- An Employee can work for atmost one Department.
- A Department can have any no. of Employees.

Now, Calculate Cardinality Ratio of works for Relationship.

now, Suppose, we have 50 employees and 3 departments and we will both as a Set (as we do in mathematics) ; E = {E1, E2, -- E50} ; D = {D1, D2,



Acc to Condition, E_1 or E_2 or any, can work for any one(1) Department and in similar way, Department can have any no. of E (N)



where, N = No. of Emp

- So Cardinality Ratio of Employee and Department ⇒ N : 1
- So Cardinality Ratio of Department and Employee ⇒ 1 : N

Basic Terminologies in Relational Model :-

- **Relation :-** It is nothing but table.
- **Tuple :-** A Row in the relation (table). Also called as Record.
- **Attribute :-** A Column in the relation. Also called as field.
- **Domain of Attribute :-** Set of values an attribute can take.

Employee

EmpId	Emp Name	Age	Sex
100	Sam	30	m
101	Bob	21	m
102	Alice	25	f
103	John	22	m

★ Schema (or) Entity type

Employee (EmpId, EmpName, Age, Sex)

★ Entity (1000, Sam, 30, m)

Domain of Sex :- {m, f, Trans}

⇒ • like Domain of Age will be any type of integer, in case of float, it will show an error (i.e in SQL). For Name, it will be Varchar (i.e. A-Z or a-z).

Question :- Why do we say Table as Relation?

Answer :- Table is known as Relation bcz it is actually Derived from Discrete mathematics

$A = \{1, 2\}$

$B = \{a, b\}$

Now, Cartesian Product, All possible value of (A, B)

$\Rightarrow A \times B$

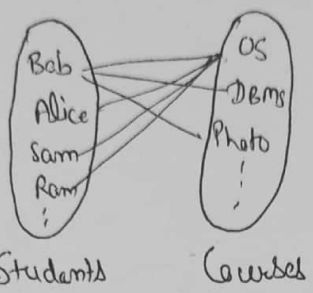
$\Rightarrow \{(1,a), (1,b), (2,a), (2,b)\}$

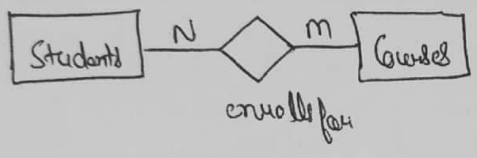
One More Example :- Requirement Analysis

A Student can enroll for any no. of courses and a Course can have any no. of students.

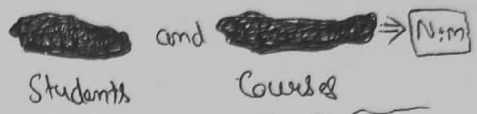
N ⟵ Student = { Bob, Alice, Sam, Ram, ---- }

M ⟵ Course = { OS, DBMS, Photoshop, --- }





So, Cardinality Ratio of

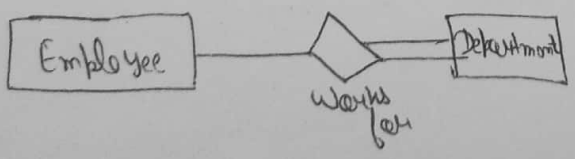


C.R of Courses & Students

M : N

★ Participation :- It is Existence Based on minimum no. of Relationships an entity can participate in. (Minimum Cardinality)

Total Participation vs. Partial Participation :- If Every Entity participates in the relationship atleast once, than we say that participation of that Entity in that relationship is total. Else the participation of that Entity in that relationship is partial.



Employee = { Sam, Bob, Alice }

Department = { Accounts, Finance }

Total Participation is denoted by "double lines" whereas Partial participation is denoted by "Single Line".

Example :- Student = { Ram, Sam, Bob, Alice }

Course = { OS, DBMS, Algo, Automata }

Ram ⇒ OS, DBMS

Sam ⇒ OS, Algo

Bob ⇒ OS, DBMS

Alice ⇒ OS

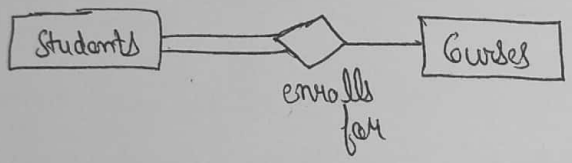
Student

SName	—	—
Ram		
Sam		
Bob		
Alice		

Courses

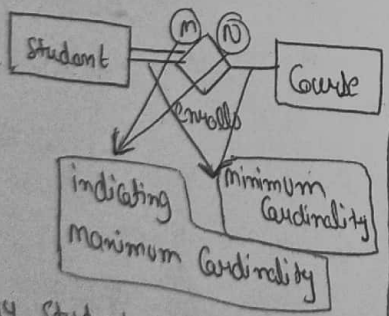
Name	—	—
OS		
DBMS		
Algo		
Automata		

Now, Every Student has atleast one Course, so the participation of an entity Student for enrolls for is Total, Whereas Automata Course doesn't have any Student, so due to Automata Course, the participation of entity Course is Partial.



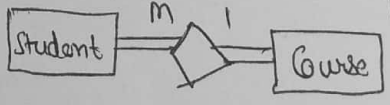
☆ Understanding Participation and Cardinality

1



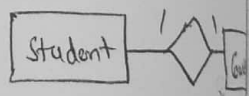
- Every Student has enrolled for atleast one Course. (=)
- it is not mandatory that a Course should have atleast one student (–).
- Maximum no. of Courses a Student can enrolled for (N).
- Maximum no. of Students a Course can have (m).

2



- Every Student has enrolled for one Course atleast (=).
- Every Course have atleast one Student (=).
- Maximum no. of Courses a Student can have is (1).
- Maximum no. of Students a Course can have (m).

3



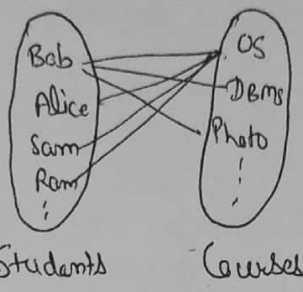
- It is not mandatory that a Student can for a Course (–).
- it is not mandatory that a Course can atleast one Student
- Maximum no. of a Student can have
- Maximum no. of

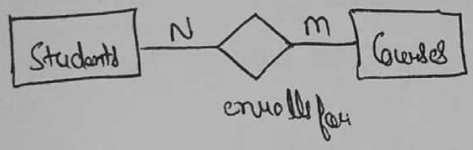
More Example :- Requirement Analysis

A Student can enroll for any no. of courses and a Course can have any no. of students.

N ⟵ Student = { Bob, Alice, Sam, Ram, ---- }

M ⟵ Course = { OS, DBMS, Photoshop, --- }





So, Cardinality Ratio of [redacted] and [redacted] ⇒ N:M

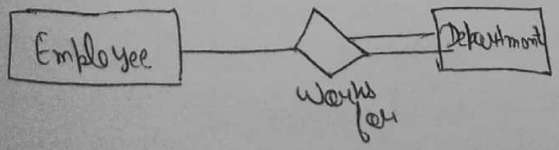
Students Courses

C.R of Courses & Students

M : N

★ Participation :- It is Existence Based on minimum no. of Relationships an entity can participate in. (minimum Cardinality)

Total Participation VS. Partial Participation :- If Every Entity participates in the relationship atleast once, than we say that participation of that Entity in that relationship is total. Else the participation of that Entity in that relationship is partial.



Employee = { Sam, Bob, Alice }

Department = { Accounts, Finance }

Total Participation is denoted by "double lines" whereas Partial participation is denoted by "single line".

Example :- Student = { Ram, Sam, Bob, Alice }

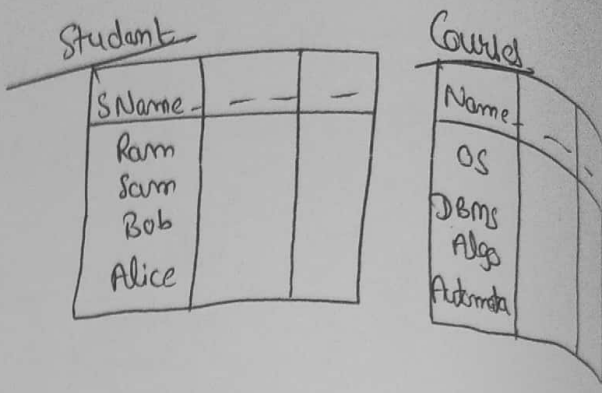
Course = { OS, DBMS, Algo, Automata }

Ram ⇒ OS, DBMS

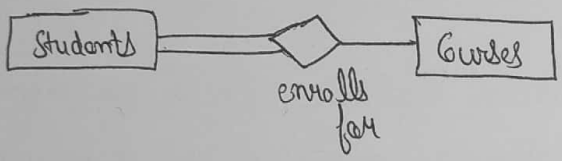
Sam ⇒ OS, Algo

Bob ⇒ OS, DBMS

Alice ⇒ OS

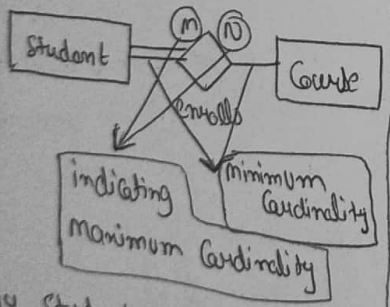


Now, Every Student has atleast one Course, so the participation of an entity Student for enrolls for is Total, whereas Automata Course doesn't have any Student, so due to Automata Course, the participation of entity Course is Partial.



☆ Understanding Participation and Cardinality

(1)

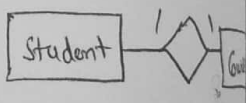


- Every Student has enrolled for atleast one Course. (=)
- It is not mandatory that a Course should have atleast one Student (–).
- Maximum no. of Courses a Student can enrolled for (N).
- Maximum no. of Students a Course can have (m).

(2)

- Every Student has enrolled for one Course atleast (=).
- Every Course have atleast one Student (=).
- Maximum no. of Courses a Student can have is (1).
- Maximum no. of Students a Course can have (m).

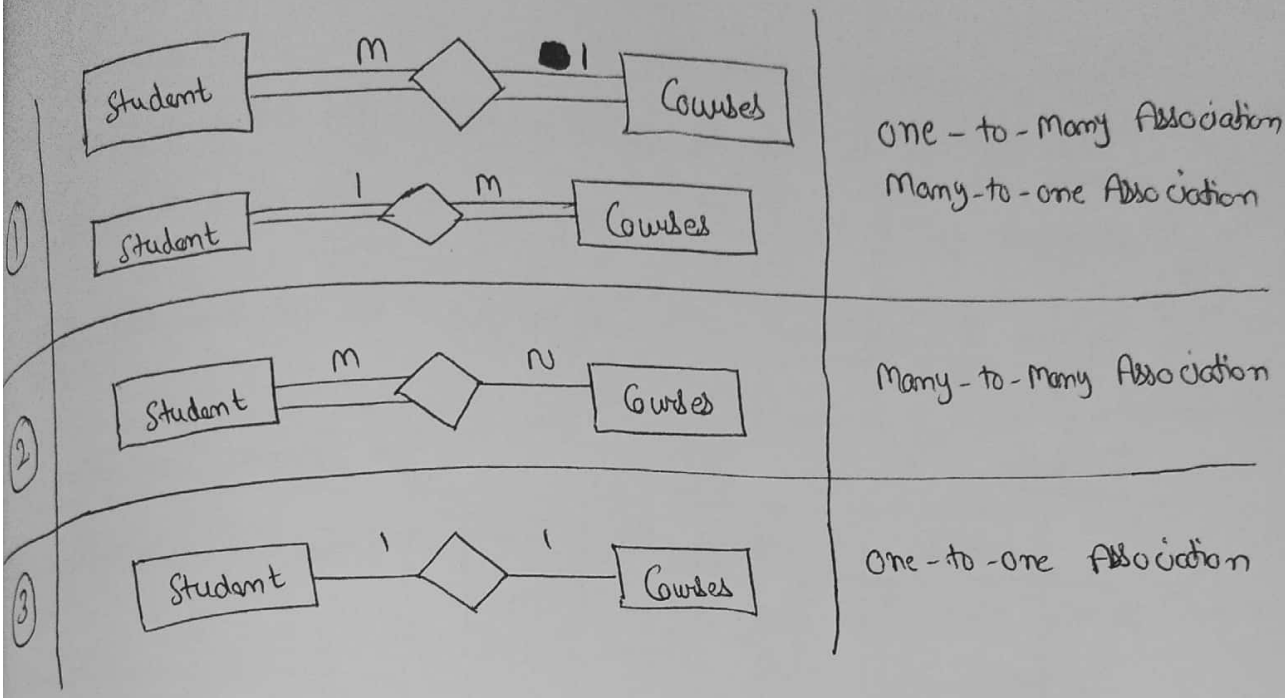
(3)



- It is not mandatory that a Student can enrolled for a Course (–).
- It is not mandatory that a Course can have atleast one Student
- Maximum no. of Courses a Student can have is
- Maximum no. of Students a Course can have (1)

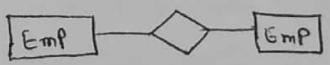
Types of Relationships :-

No. of Students ⇒ m
No. of Courses ⇒ n



Imp Note :- Association is predicted By maximun No. of Cardinality.

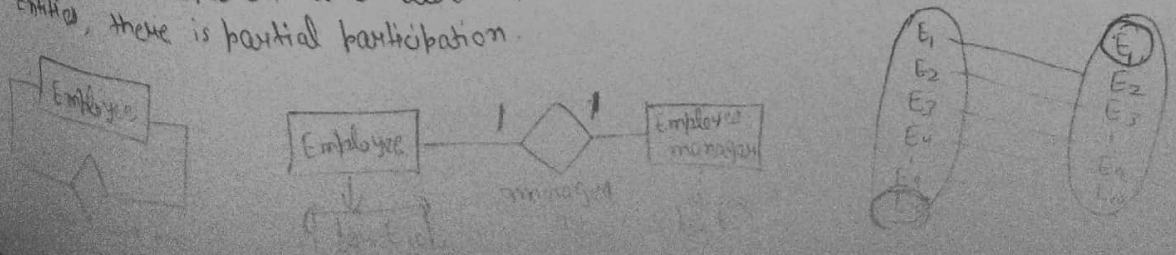
★ **Recursive Relationship** :- If a Relationship takes place b/w two entities and if both the entities are same, then we say such a relationship as Recursive Relationship.



Assume that a Company has 10 Employees named $E_1, E_2, \dots E_{10}$. An Employee E_{i+1} is the manager of Employee E_i where i can take values from 1 to 9.

It means E_1 managed by E_2 and so on $\left[E_1 \xrightarrow{\text{managed by (Relation)}} E_2 \rightarrow E_3 \rightarrow E_4 \dots E_9 \rightarrow E_{10}\right]$

⇒ It is Recursive because, Employee is being managed by another another Employee. But the Employee (E_{10}) is not managed by anyone, Hence, it is partial participation and also E_1 is not manging any Employee. So for both Entities, there is partial participation.



★ Basic Terminologies in Relational Model :-

- **Relation :-** It is nothing but table.
- **Tuple :-** A Row in the relation (table). Also called as Record.
- **Attribute :-** A Column in the relation. Also called as field.
- **Domain of Attribute :-** Set of values an attribute can take.

Employee

EmpId	Emp Name	Age	Sex
100	Sam	30	m
101	Bob	21	m
102	Alice	25	f
103	John	22	m

★ Schema (or) Entity

Employee (EmpId, Emp...

★ Entity (1000, Sam, 3...

Domain of Sex :- {m, f, Trans}

⇒ • The Domain of Age will be any type of integer, in case of float, it show an error (i.e. in SQL). For Name, it will be Varchar (i.e. A-Z...

Question :- Why do we say Table as Relation?

Answer :- Table is known as Relation bcz it is actually Derived fr... Discrete mathematics

$A = \{1, 2\}$

$B = \{a, b\}$

Now, Cartesian Product, All possible value of (A, B)

⇒ $A \times B$

⇒ $\{(1,a), (1,b), (2,a), (2,b)\}$

Now, if we take a Subset of A×B, then it will be known as Relation.

Let us take C be the Subset

$C = \{ (1,a) \}$

$C = \{ (1,a)\ (2,b) \}$

$C = \{ (2,a), (1,b) \}$

$C = \{ \}$ — Trivial Subsets

$C = \{ (1,a), (1,b), (2,a), (2,b) \}$

→ C is a Subset of A×B then we can say C must be a Relation.

$C \subset (A \times B)$

bcz, empty Subset will be a Subset of any Set and also, Same Set will also be a Subset of original Set.

⇒ Now, Come to Database.

Employee

Emp Id	Sex
1000	M
1010	F
1025	TG
1050	M

Employee = { (1000, m), (1010, f), (1025, TG), (1050, m) }

Emp Id = { 1000, 1001, 1002, --- 2000 }

Sex = { m, f, TG }

Now, Emp Id × Sex = { (1000, m), (1000, f), (1000, TG),
(1001, m), (1001, f), (1001, TG),
⋮
(2000, m), (2000, f), (2000, TG) }

- Now, we can see that, the Tuple in Employee table will be the Subset of Cartesian product of its Attributes or we can say the values of Employee will be Subset of (Emp Id × Sex), that's why Employee is known as Relation, bcz Subset of (Emp Id × Sex) will be Relation acc. to Topic "Set the Area in Algebra" from Discrete Mathematics.

- **Degree of Relation :-** No. of Attributes in a relation is called as degree of relation.

- **State of Relation :-**

Employee

EmpName	Age	Sex
Ram	18	M
~~Sam~~	~~20~~	~~M~~ → if Sam left the Company.

Actually State of Relation means the current Tuples / state of a table.

- **Intension :-** Schema (or) Entity type is also called as Intension.
  Employee (Emp Name, Age, Sex) → Intension

- **Extension :-** Instance of Schema (or) Entity type is Entity of table is also known as Extension.
  (Ram, 18, M) → Extension

⇒ Important Rules in Relational Model :-

☆ No Duplicates should exist in a relation.
We know, Relation is a set, A = {1, 2, 1} ✗ → it is not allowed in Sets

For Employee :- Employee = {(1000, -) (1001, -), (1000, -)} → not allowed

☆ Though Domain of an attribute doesn't Contain "NULL", we can still use it if necessary.
① Not Applicable
② Doesn't Exist.

Example :- (Case 1) :- if a person doesn't have phone no. but phone has domain 10 digits, though we can use "Null" here, as "Null" is not a part of Domain and it comes under Doesn't exist Case.

Employee

EId	FN	MN	LN	Phone
1000	Ram	Singh	S	012
~~1000~~	~~Ram~~	~~Singh~~	~~S~~	~~012~~ → not allowed
1002	A	B	C	NULL → not having no

(2) :- Suppose, a person doesn't have Middle Name and it cannot under Not allowed

Constraints :- A limitation or Restriction.

Domain Constraints :-

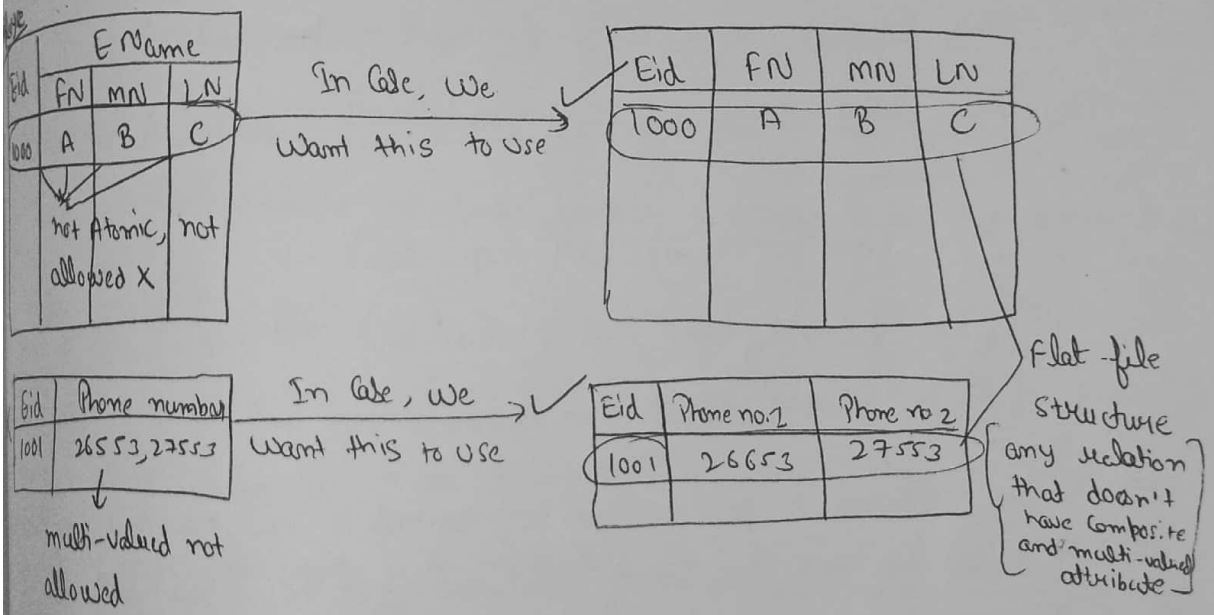
(i) An attribute can take values only from its Domain

ex :- like Sex → {M, F, TG}

if we take value otherthan from above set, for Sex, then it will throw an error, except "NULL".

(ii) Within each tuple, the value of each attribute must be an atomic value (or) Every relation should have a flat file structure.

So, we should not use Composite and Multivalued Attribute



key Constraints :-

(i) No 2 tuples should have the same value for all Attributes (or) A relation should never have dulplicate tuples.

Employee

Eid	EName	Age	Sex
1000	ABC	35	M
~~1000~~	~~ABC~~	~~35~~	M

- **Super key** :- Set of attributes which uniquely identify a tuple in a relation.

Employee

EmpId	Name	Age	Sex
1000	Bob	25	M
1001	Sam	25	M
1002	Bob	26	M
1003	Alice	27	F

Here, we can see that, we can differ employees on the basis of their EmpId, so EmpId will be Super key. (like primary key).

- Now whether (EmpId, Name) is Super key or not?

⇒ Yes, bcz EmpId differs every Employee here also.

Note :- "Any SuperSet of a Superkey is a Super key."

Like, EmpId is a Superkey, Now any SuperSet of this will be a Superkey.

Eg :- (EmpId, Name) → (1000 Bob)

(EmpId, Age) → (1000 25)

(EmpId, Age, Sex) → (1000 25 M)

⇒ To check whether a Set is Super key or not, just go and check all the tuples for that Set Set of attributes, if all the tuples will be different then Set will be Super key. (i.e. distinct / No Duplicate)

Table

A_1	A_2	A_3	A_4
1	2	5	5
2	2	6	5
3	3	5	6
4	4	8	6

(A_1) → ✓

(A_2) → ✗

(A_3) → ✗

(A_4) → ✗

(A_1, A_2) (A_1, A_3) (A_1, A_3, A_4) → ✓

(A_2, A_3) → ✓ {(2,5) (2,6) (3,5) (4,8) → all are distinct}

(A_3, A_4) → ✓ {(5,5) (6,5) (5,6) (8,6) → all are distinct}

key :- Minimal Set of attributes which uniquely identify a type in a relation. (or) Minimal Super key.

How to check :- if we delete one or more of the attributes from the set and then we are also able to find a Super key, then this set will not be a key.

Example :- (i) (A_1, A_2) → Super key

if we remove A_2

(A_1) → Super key

it means (A_1, A_2) is not a key.

(ii) (A_2, A_3) → Super key

if we remove A_3 | A_2

(A_3) | (A_2) → not a Super key

it means (A_2, A_3) is a key. → (or minimal Super key, i.e we can't remove attributes further and able to say it is a Super key).

Note :- (No. of Super keys) > (No. of keys)

⇒ if a Set is not a Super key, then it will surely not to be minimal Super key.

⇒ if a Single attribute is a Super key, then it must be a minimal Super key.

(A_1) → Super key as well as minimal Super key.

- **Candidate key** :- These are Similar to keys (i.e. Minimal Super key) but the main difference is that, these keys are the Candidates for primary key. (i.e. for Selection of primary key).
- **Primary key** :- Among all available Candidate keys, the key which we are using in order to implement the database is known as primary key. We can select only one key as primary key.

Problem-1 :- A relation R has 4 attributes namely A, B, C and D. If it has a Single key (A, C), how many Super keys are possible for R?

Answer :-

<u>A</u>	<u>C</u>	B	D

We know, Super key is nothing but Super Set of key, So Any Superkey will Contain (A, C)

So, AC is Super key

Also, ACB ✓

ACD ✓

ACBD ✓

So, Total 4 Super keys are possible for this Relation.

But this is not Correct way, So To find all possible Super-keys we will use Combinatrix.

We know, A & C will be always attribute for Super key

⇒ AC _ _ (B's place, D's place) [ACB ACD ACBD AC]

We Can fill these 2 places in ways :- We Use permutation

$2 \times 2 = 4$ Ways

Problem-2 :- A relation R has 4 attributes namely A, B, C and D. If it has a Single key A, how many Super-keys are possible for R?

Answer :-

A _ _ _ (B, C, D)

1 way, 2 way, 2 way, 2 way

So, All possible Super keys :- $1 \times 2 \times 2 \times 2$

⇒ 8 ways

Note :- Key Constraints :-

① No 2 tuples should have the same value for all attributes
(or)
there should not be any duplicate tuple in a relation.

② Every table should definitely have a primary key. Also a primary key should never accept NULL Values.

③ Entity Integrity Constraints :- A primary key cannot have NULL values.

Employee

Ename	Phone	Age	Sex
Sam	1234	40	M
Bob	2312	45	M
NULL	5678	35	M
NULL	9012	30	F

- We use primary key, to identify a single particular tuple as uniquely identified a tuple, thats why primary key can't be NULL, bcz if it is NULL, then we can't identify a tuple.

④ Foreign key :- A foreign key cannot take value apart from the values in Reffered attribute is known as Referential Integrity Constraints.

Employee

Eid	Ename	Dep Id
1	Bob	1000
2	Sam	1000
3	Alice	1001
4	Jacob	1000
5	Boby	1002

Reffering Attribute

Department

Reffered Attribute

Dep Id	Dep Name	Dept Head
1000	Accounts	
1001	Finance	
1002	Management	
1003	Marketing	
1004	Sales	

Domain of this attribute

Foreign key :- Any Attribute which is reffering to some other Attribute, we say reffering attribute as a foreign key and reffered attribute must be a primary key of its relation (or table).

* imp

Example :- Employee

Emp Id	Emp Name	Age	Manager
1000	Bob	35	
1001	Sam	25	
1002	Alice	19	
1003	Peter	20	
1004	Anderson	26	

- Manager is a foreign key in this relation, which is refferring to Emp Name.
- Here we can see that a foreign key can be a attribute which is not a primary key (like here, it is refferring to Emp Name but it is not a primary key) but practically it will not possible bcz suppose there are 2 employees with same Name, than there will be a confusion, who is the manager, so practically it is not possible, but there is not any hard & fast rule that
  - ★ Reffered attribute must be from different relation.
  - ★ Reffered attribute must be a primary key.

Foreign key

1. More than 1 foreign key is possible in a relation.
2. A foreign key can take NULL value.

Primary key

1. A relation can have only one primary key.
2. A primary key cannot have NULL values.

Employee

Eid	Ename	Dep Id	Manager Id
1	Bob	1000	2
2	Sam	1000	3
3	Alice	1001	4
4	Peter	1003	NULL

Department

Did	DName
1000	
1001	
1002	
1003	

Note :- How Updation, Insertion and Deletion can violet these Constraints.

* Domain Constraints :- Insert, Update
* Key Constraints :- Insert, Update
* Entity Integrity Constraints :- Insert, Update
* Referential Integrity Constraints :- Insert, Update, Delete.

Now, To Deal with these operations while violating these Constraints

For Insertion and Updation :- Simply Reject the Operation

For Deletion :- (1) Reject the Operation

(2) On Delete Cascade (i.e. while deleting the tuple of referred attribute, Delete the tuples of referring attribute corresponding to it, in case of referential Integrity Constraint).

(3) On Delete SET NULL.

Problem :- The following table has two attributes 'A' and 'C' where A is the primary key and C is the foreign key referencing A with ON-DELETE CASCADE. The set of all tuples that must be additionally deleted to preserve referential integrity when the tuple (2,4) is deleted is —

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

A	C
~~2~~	~~4~~
3	4
4	3
~~5~~	~~2~~
~~7~~	~~2~~
~~9~~	~~5~~
6	4

Now, when deleting (2,4), and 2 is the primary key of A, so, tuple having 2 as a foreign key can't exist, so delete (5,2) and (7,2) and then also tuple having 5 and 7 as foreign key can't exist, so delete (9,5)

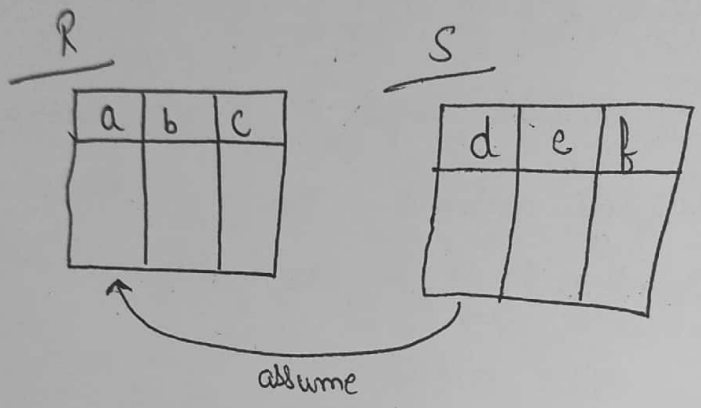
Problem :- Let R(a,b,c) and S(d,e,f) be two relations, 'd' is foreign key of S that references to the primary key of Consider the following 4 operations on R and S.

(a) Insert into R (b) Insert into S

(c) Delete from R (d) Delete from S

Which operation(s) among the above 4 can cause referential integrity Constraint violation?

⇒



(i) If we insert into R, then there will not be any Referential Integrity Constraint Violation.

(ii) If we insert into S, there may be Referential Integrity Violation, in case if 'd' take value which is not in 'a'.

(iii) If we Delete from R, there again may be Referential Integrity Violation, in case Suppose 'd' has value(s) and 'a' also, but if we delete it from R, then we have to also delete tuples corresponding to the value of (s) from S. to deal with it.

(iv) If we delete from S, there will not be any Referential Integrity Constraints.

Problem :- Let $R(Attr_1, Attr_2, Attr_3, \ldots, Attr_n)$

Let $Attr_1$ be the key

Find the no. of Superkeys possible for R?

$\Rightarrow$ if $Attr_1$ is a key, and we know Superkey is superset of key.

So, $<Attr_1>$ → Superkey

$<Attr_1, Attr_2>$ → Superkey

i.e. $Attr_1$ must be there

So, $<\underline{Attr_1}, \underline{Attr_2}, \underline{Attr_3}, \underline{Attr_4}, \ldots \underline{Attr_n}>$

1 way × (2 ways × 2 ways × 2 ways ---× 2 ways)

2^{n-1}

So, Total of 2^{n-1} Superkeys are possible for R.

Problem :- $R(Attr_1, Attr_2, \ldots Attr_n)$

If $(Attr_1, Attr_2)$ is the key (only key) of R. Find the no. of Superkeys possible for R?

$\Rightarrow <\underline{Attr_1}, \underline{Attr_2}, \underline{Attr_3}, \ldots \underline{Attr_n}>$

1 way × 1 way × 2 ways --- × 2 ways

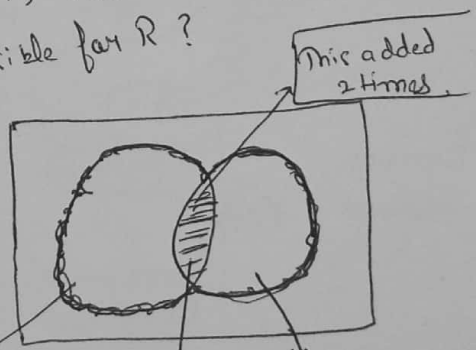
2^{n-2}

So, Total no. of Superkeys possible for R will be 2^{n-2}.

Problem :- $R(Attr_1, Attr_2, \ldots Attr_n)$

If $Attr_1$ and $Attr_2$ are 2 keys of R, find the no. of Superkeys possible for R?

$\Rightarrow$



No. of Superkeys possible assuming that $(attr_1)$ is key

2^{n-1}

No. of Superkeys possible assuming that $(attr_2)$ is key

2^{n-1}

No. of SK's possible assume both $(attr_1$ & $attr_2)$ are key

2^{n-2}

So, Required Ans $\Rightarrow [2^{n-1} + 2^{n-1}] - 2^{n-2}$

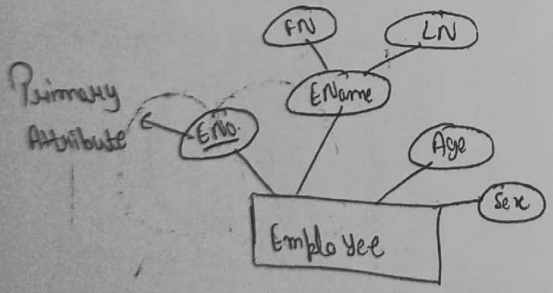
$\Rightarrow 2(2^{n-1}) - 2^{n-2}$

Problem :- Let R ($Attr_1$, $Attr_2$, ... $Attr_n$)
If $Attr_1$ and ($Attr_1$, $Attr_2$) are 2 keys of R, find the no. of Superkeys possible ?

⇒ This Question is not possible, bcz if $Attr_1$ is a key than ($Attr_1$, $Attr_2$) can't be a key, bcz we know a key is a Super minimal key, and ($Attr_1$, $Attr_2$) is a Super key but it is not a minimal Super key, as if we reduce one attribute, than remaining attribute is also a Super key ($Attr_1$, ~~$Attr_2$~~)

↓
Super key (bcz it is also a key)

★ Conversion of ER Model to Relational Model

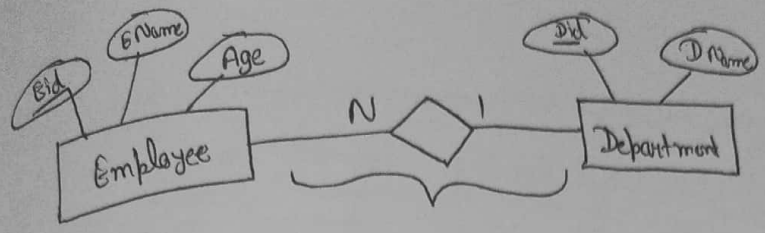


Employee
Primary key

ENo.	Age	Sex	FN	LN

1. For Every Entity in ER M... Construct a Relation (table).
2. A Simple Attribute in ER diagr... will be an Attribute (or) Colum... in the Relation.
3. If an attribute in ER diagram is prime attribute, than make it as a primary key in the Relation...
4. If an attribute is a Composite attribute in the ER diagram, than break those Composite attributes in the Relation...

Converting one-to-many relationship in ER diagram to relational model :-



- An Employee can have almost 1 Department.
- A Department can have almost N employee

Employee

Eid	Ename	Age
1	Ram	25
2	Sam	26
3	Bob	29

Department

Did	Dname
1000	Accounts
1001	Finance
1002	Marketing

"Now, To show Relationship b/w two relational tables, we will take a primary key from either table and make it as a foreign key for another table".

Now, Case-1 :- if Did is moved as a foreign key for Employee

Employee

Eid	Ename	Age	Did
1	Ram	25	1000
2	Sam	26	1002
3	Bob	29	1001

Did	Dname
1000	Accounts
1001	Finance
1002	Marketing

This is totally Valid

"An Employee can work only for one department."

Now, Case-2 :- If Eid is moved as a foreign key for Department

Eid	Ename	Age
1	Ram	25
2	Sam	26
3	Bob	29

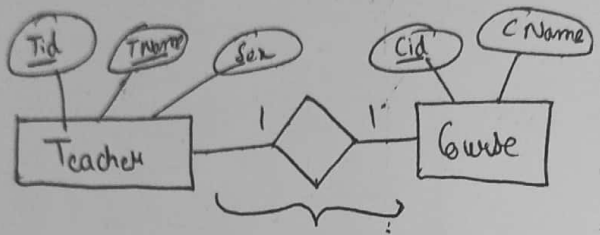
Did	Dname	Eid
1000	Accounts	1, 2
1001	Finance	2, 3
1002	Marketing	3, 1

→ Invaid :- bcz it is not a flat-file structure.

This totally Invalid, bcz in this we are violating rules for relational model

Note :- In case of one-to-many Relationship, Always it is recommended to move the primary key of (1's) side as foreign key to (N's) side.

Converting one-to-one Relationship in ER Diagram To Relational Model



Teacher

- A Teacher can teach 1 Course.
- A Course can taught atmost 1 teacher.

Tid	Tname	Sex	Cid
1	Bob	m	
2	Jack	m	
3	Sam	m	

No Necessity of multi valued attribute Here

Cid	Cname	Tid
1000	DBMS	
1001	OS	
1002	Algorithms	

No Nece... of multi... attribute...

- In this one-to-one Relationship, We can move any primary key from either table as a foreign key.
- We can also use third table to represent this relationship but this will not good in practise as it will take more space.

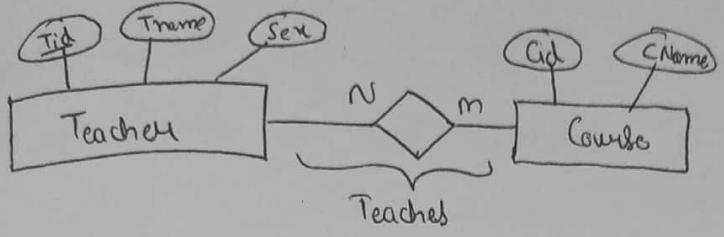
Third Table :-

Tid	Cid
Bob	
Jack	
Sam	

or

Tid	Cid
	DBMS
	OS
	Algorithms

Converting many-to-many Relationship in ER Diagram TO Relational Model



- A Teacher can teach atmost M courses.
- A Course can be tought by atmost N Teachers.

- In this many-to-many Relationship, we can't move any primary key as a foreign key bcz ⇒

Teacher

Tid	Tname	Sex	Cid
1	Bob	M	DBMS OS
2	Jack	M	
3	Sam	M	

→ Not Valid

Courses

Cid	Cname	Tid
1000	DBMS	Bob Jack
1001	OS	
1002	Algorithms	

→ Not Valid

- So In many-to-many Relationship, we have to make third Table

Tid	Cid
Bob	DBMS
Bob	OS
Jack	OS

foreign key

foreign key

Every tuple will be unique when (Tid, Cid) → primary key

- If Tid is a primary key, than it will again violet flat-file structure.
- If Cid is a primary key, then it will again violet flat-file structure.
- If there will not any primary key, but it is not possible acc to rule, a relation must have primary key.
- So the only solution is, making (Tid, Cid) will act as a primary key so. Every tuple will be diffarent.

Imp Question:- Given the Basic ER and Relational models, which of the following is incorrect?

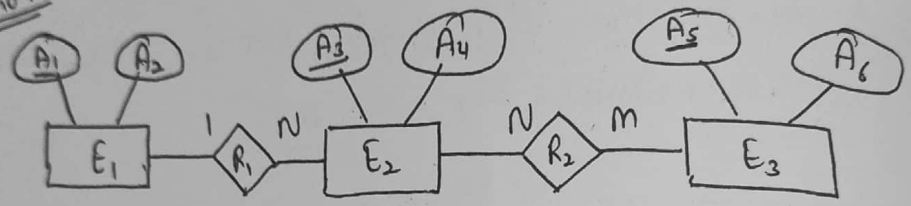
A) In ER model, An Attribute of an Entity can have more than one value.

B) In ER model, An Attribute of an Entity can be Composite.

C) In a row of a Relational Table, An attribute can have more than 1 value. ← this is wrong/Incorrect

D) In a row of Relational table, an attribute can have exactly one value (or) a NULL value.

Do in Revision



What is the minimum no. of tables needed to Represent ER diagram in Relational model?

⇒ E_1

A_1	A_2

E_2

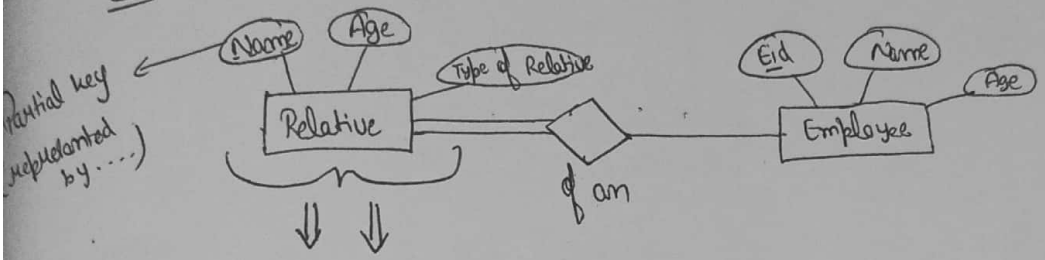
A_3	A_4

E_3

A_5	A_6

Converting Week Entities in ER Diagram to Relational Model :-

- Week Entity :- An Entity with no prime attribute.
- Strong Entity :- An Entity with prime attribute.



It is not applicable to relational model, bcz in this ER model we don't have any prime attribute (i.e. Week Entity) and there is a rule for relational model, that there must be a primary key in table

- Note :- Weak Entities are still allowed in ER Diagram with a Condition that, it should be in total participation with Strong Entity.

To Represent Relatives in relational model, we need to move the Eid of Employee to relatives table, bcz relatives are in total participation with Employee, Now for a primary key we can't make Eid/Name/Age as a primary key, else we want to make (Eid, Name) as a primary key which defines the tuple differently and in this primary key, there will be a partial key (i.e Name here) which means we don't have a same relative for same Eid.

Employee

Eid	Name	Age
1	Prince	20
2	Riya	25
3	Hardik	29

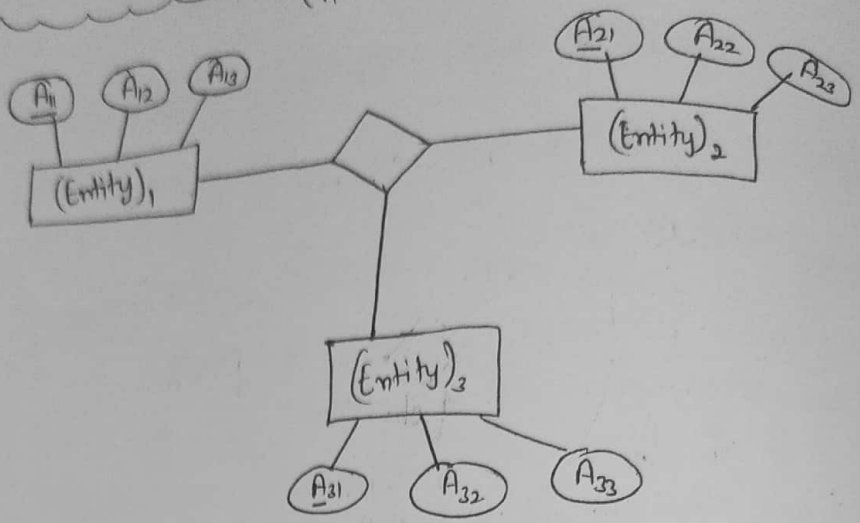
Relative

Eid	Name	Age	Type of Relative
1	Ram	55	Uncle
1	Sam	70	Father
2	Sam	85	brother

Note :- A Week Entity must have partial key, though partial key doesn't act as primary key, though partial key alone doesn't identify a tuple uniquely, but for given value of Strong Entities key, this partial key is able to identify a unique tuple

★ Converting n-ary Relationships in ER diagram to Relational ...

$(n > 2)$



Entity1

A_{11}	A_{12}	A_{13}

Entity2

A_{21}	A_{22}	A_{23}

Entity3

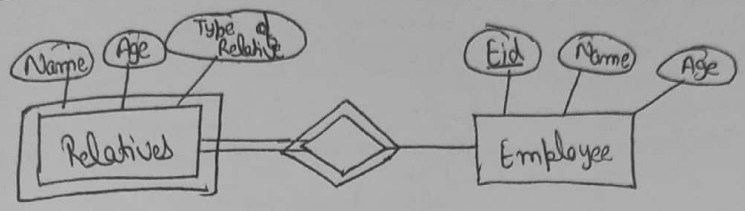
A_{31}	A_{32}	A_{33}

Now, In case of n-ary Relationships (i.e. $n>2$, not a Binary Relation) We always need to create a new table having attributes the primary keys of all Entities respectively and each acts as foreign key to their respective relation and the all set of attributes will acts a primary key for the table.

A_{11}	A_{21}	A_{31}

Identifying Relationship :- Relationship exists b/w a Strong Entity And Weak Entity is known as Identifying Relationship.

The actual ER Diagram of a model, we studied on previous page



Symbols Used in ER Diagrams

① Entity :-

② Relationship :-

③ Attribute :-

④ key Attribute | prime attribute :-

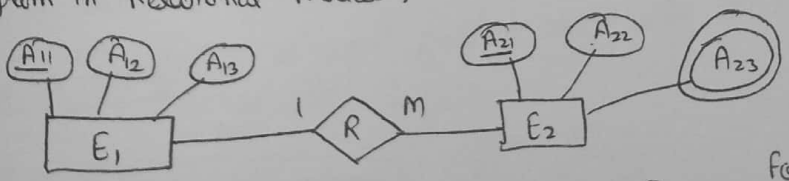
⑤ Multivalued Attribute :-

⑥ Weak Entity :-

⑦ Composite Attribute :-

⑧ Identifying Relationship :-

What is the minimum no. of tables needed to represent the above ER Diagram in Relational model?



For a multi valued Attribute, we have to create a new table

E1

A11	A12	A13

E2

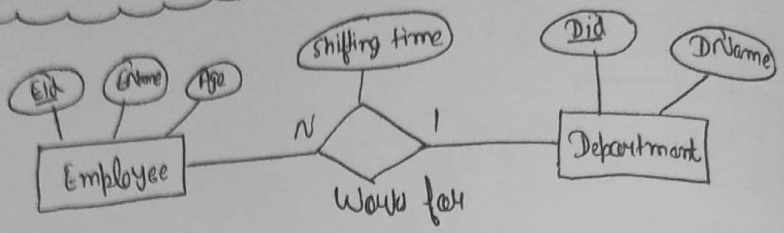
A21	A22	A11

A21	A23

Total Nos : 3 tables

★ Attributes To Relationship :- Property denoting the Relationship both Entities.

① One to many Relationship :-



Sam → Commerce = 9 Am to 5 Pm
Ram → Commerce = 10 Am to 3 Pm

Employee

Eid	Ename	Age	Did
1	Sam	45	1000
2	Ram	43	1000

Department

Did	DName
1000	Commerce
1001	management

Now, where to move the Relationship in Relation (i.e. Table).

- We can't move Shifting time to Departments Table bez Department Table doesn't have anything about Employee and Shifting time belongs to employee
- The other point we can say if we [redacted] move Shifting time to Department table, than this attribute will definitely contain more than one value bez in one tuple it will contain times of each one but multi valued attributes are not allowed, Hence it is not pos
- Hence We will also move Shifting time to Employee table.

Employee

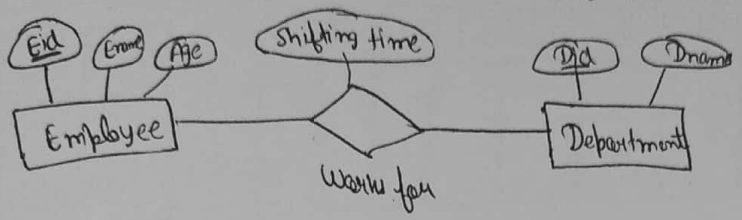
Eid	Ename	Age	Did	Shifting time
1	Sam	45	1000	9 to 5
2	Ram	43	1000	10 to 3

Department

Did	DName
1000	Commerce
1001	Management

One to One Relationship :-

We already know, we can move any primary key of either table to another table, Hence we should also move Relationship to either table but on that side where we move primary key.



Sam → Commerce = 9 am to 5 Pm

Ram → Commerce = 10 am to 3 Pm

Case 1 :-

Employee

Eid	EName	Age	Did	Shift time
1	Sam	45	1001	9 to 5
2	Ram	46	1001	10 to 3

Department

Did	Dname
1000	Science
1001	Commerce

Case 2 :-

Employee

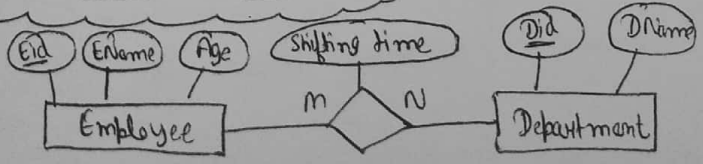
Eid	EName	Age
1	Sam	45
2	Ram	46

Department

Did	DName	Eid	Shift
1000	Science	1	9 to 5
1001	Commerce	2	10 to 3

"Both Cases are Valid"

Many to Many Relationship :-



Employee

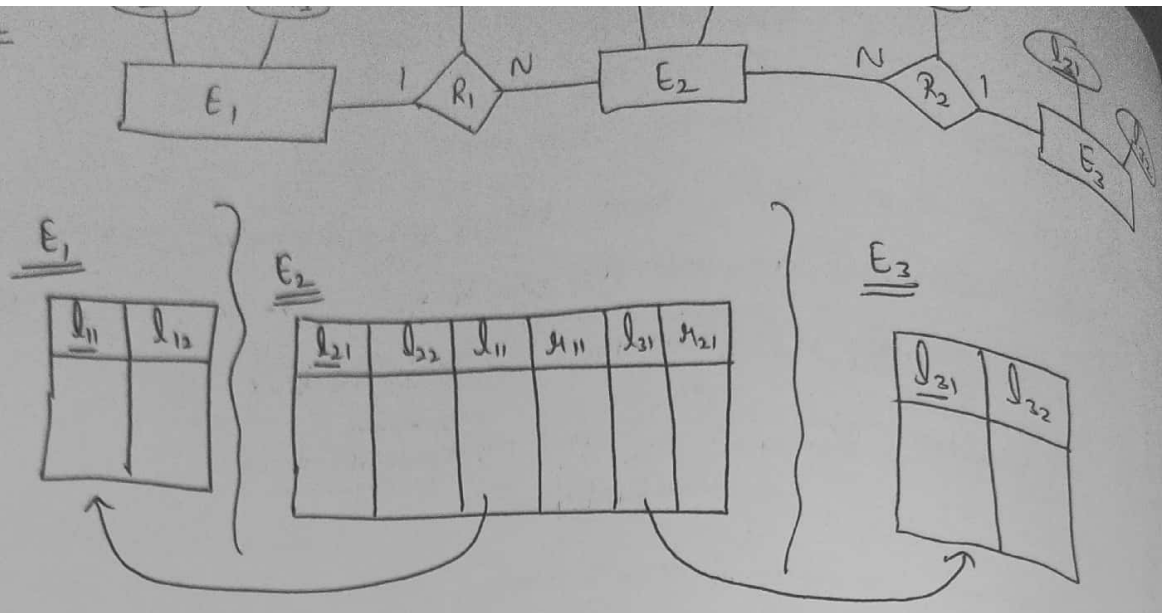
Eid	EName	Age
1	Sam	45
2	Ram	46

Department

Did	Dname
1000	Science
1001	Commerce

New table To Represent Relationship

Eid	Did	Shifting time
1	1000	9 to 5
2	1000	10 to 3



★ Functional Dependencies :- (FD's)

$A \rightarrow B$ ⇒ it means B is functionally dependent on A, where A and B are attributes of a relation

A	B
1	a
2	b
1	a

if for any value of A, if we can get the value of B, then we can say that B is functionally dependent on A.

$A=1 \rightarrow B=a$
$A=2 \rightarrow B=b$
} for value of A=1, the value of B=a

Also, $B=a \rightarrow A=1$
$B=b \rightarrow A=2$

$\therefore$ $A \rightarrow B$ and $B \rightarrow A$

A	B
1	a
1	b
2	b

Here, for given value of A=
We can not tell the value of
So clearly, $A \not\rightarrow B$
i.e. B is not functionally dep
on A.

Similarly, $B \not\rightarrow A$

$A \rightarrow BC$, $AC \rightarrow B$

A	B	C
1	2	3
1	2	4
2	3	5

A	B	C
1	2	3
2	3	1
1	2	3

$\Downarrow$

$A=1 \rightarrow BC=2,3$

$A=1 \rightarrow BC=2,4$

$A \not\rightarrow BC$

$A=1 \rightarrow BC=2,3$

$A=2 \rightarrow BC=3,1$

$A \rightarrow C$

$A,C=(1,3) \rightarrow B=2$

$(1,4) \rightarrow B=2$

$(2,5) \rightarrow B=3$

$\Rightarrow AC \rightarrow B$

$AC=(1,2) \rightarrow B=2$

$AC=(2,1) \rightarrow B=3$

$\Rightarrow AC \rightarrow B$

Imp

Note :- To check whether Right Side is functionally dependant or not in case, if the attributes on left side are unique, it means they don't have duplicates, then ~~ → ~~ always holds

☆ 3 - Types of Functional Dependencies (FD's)

1. Trivial Functional Dependency
2. Non - Trivial Functional Dependency
3. Semi - Trivial Functional Dependency

(i) **Trivial F.D** :- $X \to Y$ is a trivial F.D, iff $Y \subseteq X$

Where, X & Y can be Set of attributes or a single attribute as well.

Ex :- (i) $AB \to A$

$X = \{A, B\}$; $Y = \{A\}$

$Y \subseteq X$

This type of F.D is trivial, and it always holds True

(ii) $A \to A$

$X = \{A\}$; $Y = \{A\}$

$Y \subseteq X$

(iii) $ABC \to BC$

$X = \{A, B, C\}$; $Y = \{BC\}$

$Y \subseteq X$

A	B	C
1	2	
5	2	

$\Downarrow$

$AB \to A$

$(1,2) \to 1$

$(5,2) \to 5$

(ii) **Non-Trivial F.D** :- $X \to Y$ is a Non-trivial F.D iff $X \cap Y = \phi$

Where, X & Y can be Set of attributes or a Single attribute as well

$X \cap Y = \phi$, means Element in X & Y are different (no Common element)

Ex :- (i) $A \to B$

$X = \{A\}$; $Y = \{B\}$ $\Rightarrow$ $\{1, 2\} \to \{3, 4\}$

$X \cap Y = \phi$ $\quad X \cap Y = \phi$

This type of F.D is Non-trivial, it may or may not be held

(ii) $AB \to AC$ $\to$ this is not a Non-trivial F.D, bcz $\{A\}$ is present on both the sides

Semi-Trivial F.D :- A F.D which is neither Trivial nor Non-Trivial is known as Semi-Trivial F.D.

$X \to Y$ is a Semi-trivial F.D

iff $(X \cap Y \neq \phi$ and $Y \not\subseteq X)$

Where X, Y are Set of attributes or Single attribute.

Eg:- $AB \to AC$

$X = \{A, B\}$; $Y = \{A, C\}$

$X \cap Y = A \neq \phi$ [$\because$ Not non Trivial]

$Y \not\subseteq X$ [$\because$ Not Trivial]

Hence, it is Semi-Trivial F.D, and it may or may not be true (i.e. holds true).

$R(A, B, C)$

To Prove :- $A \longrightarrow BC$ $\equiv$ $A \to B$ and $A \to C$

$A \longrightarrow BC$: if it holds / if it doesn't holds

$\equiv$: Equivalent means

$A \to B$ and $A \to C$: then it also holds / then it also doesn't holds

To prove this equivalency, we have to prove two Cases

Case 1 :- "If $A \to BC$ holds, then $A \to B$ and $A \to C$ will hold"

$\Rightarrow$ Suppose $A \to BC$ holds and let us assume $A \to B$ doesn't holds or $A \to C$ doesn't holds, So Now, if $A \to B$ doesn't holds, then we have take duplicate values of A and if we take duplicate values of A but different values of B, then $A \to B$ doesn't hold, but along with this $A \to BC$ will also doesn't hold, but we Suppose $A \to BC$ holds, So our Contradiction is wrong, Hence $A \to BC$ will holds, if $A \to B$ and $A \to C$ holds

B	C
2	1
3	1

$A = 1$ — $BC = (2, 1)$

$BC = (3, 1)$

Case-2 :- If $A \to B$ and $A \to C$ holds, then $A \to BC$ will holds

⇒ Suppose $A \to B$ and $A \to C$ holds and let us assume $A \to BC$ doesn't holds, which means for unique value of A, there will be different values of (B, C) which means values for B, C will be different (i.e. either B or C) but if we do this then $A \to B$ or $A \to C$ will not holds, but we suppose $A \to B$ or $A \to C$ holds, so our supposition is wrong that $A \to BC$ holds, Hence $A \to B$ and $A \to C$ holds, then $A \to BC$ will hold

A	B	C
1	2	...
1	4	...

Hence :- $A \to BC \equiv A \to B$ and $A \to C$

★ Closure Set of Attributes :- Closure Set of an Attribute A, can be defined as set of all Attributes which can be uniquely identified by A and it is denoted by A^+.

Let R (ABCDE) is a relation having FD's $A \to B$, $B \to D$, $C \to DE$, $CD \to AB$

$A^+ = \{A, B, D\}$ ⇒ These are the set of attributes which uniquely identifiedby A.

Explanation :- ① $A \to A$ (Trivial FD's)
② $A \to B$ (Given)
③ Also, $B \to D$, Hence $A \to D$

Note :- Always check the Left hand side of all the functional dependencies to find left hand side which is Completely full of attributes which are present in the set

$B^+ = \{B, D\}$

$D^+ = \{D\}$

$C^+ = \{C, D, E, A, B\}$

$E^+ = \{E\}$

$(AB)^+ = \{A, B, D\}$

[∵ $AB \to AB$
then, $AB \to A$, $AB \to B$
Also, $B \to D$, Hence $AB \to D$]

$(ABD)^+ = \{A, B, D\}$

Note :- If the closure set of an attribute or set of attributes, contain all the attributes in the relation, then it will definitly be a super key but it need not to be a key whereas if the closure set of a single attribute contains all the attr. in the relation, then it will be a super key as well as a key (ie minimal key → bcz it cant be reduced further)

Suppose, $(AB)^+ = \{A, B, C, D, E\}$ & $A^+ = \{A, B, C, D, E\}$

then $(AB)^+$ is a super key but not a minimal key

bcz $(AB)^+ \dashrightarrow A^+$ is also a super key

$\therefore (AB)^+$ is not a key

Now, from previous Example, C^+ is a super key as well as a key (ie minimal key).

Closure Operations

V. Imp

:- What we will learn,

Given a relation with set of functional depen holding on that relation, I may need to find

1. The keys which are present in that relation (and also no. of
2. Identifying equivalence of 2 set of FD's
3. Finding minimal FD set.

In order to find out all the above 3, closure operations will helpful (which means closure operation makes finding out of t above 3 to be much more easier).

Imp

R(ABCDE)

How many key does this relation h

A	B	C	D	E

FD's

$A \to B$

$B \to D$

$C \to DE$

$CD \to AB$

Now, Suppose if we want to prove 'A' as a key, than its clos must contain all the attributes and also it has to be a super key, so first to solve these type of Questions, check right side of and in this Question we can see that 'C' is missing, so Any Super form should definitely have 'C' inside it. Like C, CAB, CB, CA --

↓

but these will be the keys bez if Remove any attribute than Can't find a super minimal key.

So, $C^+ = \{C, D, E, A, B\}$

Hence, C is a Super key

$\Rightarrow$ C is a key

Note :- First we come up with an idea that any super key you form should definitely have 'C' inside it. bcz if we don't have C in our super key, then we are not able to derive and if we are not able to derive C, then, we can't able to derive all the attributes in the relation and if we don't able to derive all attributes, then it will not be a super key. Hence, it will not be a key.

Ex: $R(W, X, Y, Z)$

FD's

$WX \rightarrow YZ$

$Z \rightarrow W$

Find the keys of R?

So, X should be part of any key of R.

$(X)^+ = \{X\}$

So, X is not the super key or minimal key

So, To solve these type of questions, we will be going step by step first we will find the keys of size 1. (i.e single attribute)

$(X)^+ = \{X\}$

X is not a key

Keys of size 2 :- (i) $(X\,W)^+ = \{X, W, Y, Z\}$ ✓ $\Rightarrow$ XW is a key of R

(ii) $(X\,Y)^+ = \{X, Y\} \Rightarrow X$ $\Rightarrow$ XY is not a key of R

(iii) $(X\,Z)^+ = \{X, Z, W, Y\}$ ✓ $\Rightarrow$ XZ is a key of R

So, once we have done with size 2, we will go for size 3, so Generally we will find or try to add a single attribute to the wrong closure above to get correct (✓) closures. If we remove an attribute, then definitely it will

keys of size 3 :- $(X\ Y\ W)^+ = \{X, Y, W, Z\} \Rightarrow$ it is a key

$(X\ Y\ Z)^+ \rightarrow$ we will don't see its closure bcz if we remove Y, then $(X * Z)^+$ will be a subset key that means not a key.

but it is not a wrong item, bcz $(X\ Y\ Z)^+ = \{X, Y, Z, W\}$ this is not a wrong item, but also not a key.

Note :- we will continue to further size, until we get zero wrong items.

⇒ Candidate keys for Sub-Relation :-

To Avoid redundancy, we generally split a larger table into smaller sub-tables (called as sub-relation).

Now once you have broken a large table into set of smaller sub tables, how will you find out keys of each relation?

R (M, N, O, P) FD's $\{MN \rightarrow OP,\ P \rightarrow M\}$

What are the keys of sub-relation R (N, O, P)

⇒ As usual check the right side of FD's

So, N must be present of any key of R.

$(N)^+ = \{N\}$

So, N is not a key

keys of size 2 :- $(NO)^+ = \{N, O\}$ X

$(NP)^+ = \{N, P, M, O\}$ ✓

⇒ NP is a key

key of size 3 :- $(NO\ P)^+$ ⇒ but it will not be key $(NO)^+$ is a key

So, $(NP)^+$ is the only key

NOTE

We will no
Consider 'M'
to be in cl
Set of any k
bcz we ha
to find the
for sub-rela
(N, O, P).

Equivalance of 2 Functional Dependancies :-

Two or more than two sets of functional dependancies are called equivalance if the right hand side of one set of functional dependancy can be determined using the second FD set, similarly the right-hand side of the second FD set can be determined using the first FD set

Check whether both of them are equivalent or not.

$F = \{ M \to N, N \to O, O \to P \}$

$G = \{ M \to NO, O \to P \}$

G Covers F :- for this we will take FD's from F and tell whether G covers or not

So, $(M)^+ = \{N, M, O$

$(N)^+ = \{N\}$

$N \to O$ doesn't covered by G and we can say that $F \not\subseteq G$

So, whether G covers F or F covers G is not true than F & G are not Equivalent

$F = \{ M \to N, MN \to O, P \to MO, P \to Q \}$ $\quad G = \{ M \to NO, P \to MN \}$

⇒ **F Covers G :-** $(M)^+ = \{M, N, O$

$(P)^+ = \{P, M, O, N$

So F Covers G

⇒ **G Covers F :-** $(M)^+ = \{M, N, O$

$(MN)^+ = \{M, N, O$

$(P)^+ = \{P, M, N, O$

$P \to Q$ doesn't covered by G

Note :- Application of closures :- (1) Finding no. of keys in a relation
(2) Equivalance of 2 FD's
(3) Minimization of FD's

★ Minimal Cover / Minimal FD's :-

If I can Minimize the FD set F to FD set G in a Such a way that F and G are equivalent, then G is called as the minimal FD set of F.

⇒ Procedure to find minimal FD set :-

- Step-1 :- Split the FD's in Such a way that RHS only Contains a Single attribute
- Step-2 :- Find the redundant FD's and delete them from the Set.
- Step-3 :- Find the redudant attributes on LHS and delete them (Applicable only if LHS Contains more than 1 attribute).

Minimize :- $\{M \to O, MO \to P, Q \to MP, Q \to R\}$

- Step-1 :- Simply split the FD's in Such a way that RHS only Contains a Single attribute (like we called it as decomposition):

$M \to O$
$MO \to P$
$Q \to M$
$Q \to P$
$Q \to R$

Step-2 :- Now find the redundant FD's and delete them from the set, To do this we have take all the FD's one by one and check every FD with all the remaining four by taking Closure of it

Now let us take ① $M \to O$

So, $(M)^+ = \{M,$

✗ We cant obtain 'O' from the remaining FD's, So it will not be redundant FD

② $MO \to P$

$(MO)^+ = \{M, O$

✗ We cant obtain 'P' from the remaining FD's, So it will not be redundant FD.

③ $Q \to M$

$(Q)^+ = \{Q, P, R$

✗ We cant obtain 'M' from the remaining FD's, So it will not be redundant FD.

④ $Q \to P$

$(Q)^+ = \{Q, M, R, O, P$

✓ We can obtain 'P' from the remaining FD's, So it will be redundant FD

⑤ $Q \to R$

$(Q)^+ = \{M, P, O,$

We cant obtain 'R' from the remaining FD's, So it will not be redundant FD.

So, Sol^n will be :

$M \to O$
$MO \to P$
$Q \to M$
$Q \to R$

Step-3 :-

$M \rightarrow O$

$MQ \rightarrow P$ → we will perform on this

$Q \rightarrow M$

$Q \rightarrow R$

So, take $(M)^+ = \{M, O\}$, so closure of M contains O, so we can delete 'Q' from L.H.S

So,

$M \rightarrow O$

$M \rightarrow P$

$Q \rightarrow M$

$Q \rightarrow R$

→ This is the minimized FD set

Note :- If $(M)^+$ doesn't contains O, then we will check for Q that if it contains M, then we will delete 'm'.

DeCompoSition :- The Process of breaking up or dividing a Single relation into two or more Sub relations is called the decomposition of a relation.

⇒ **Properties of DeComposition** :-

(i) **Lossless DeComposition** :- Loseless DeComposition ensured, No information is lost from the original relation during deComposition.
When the sub relations are joined back, the same relation is obtained that was deComposed.

(ii) **Dependancy Preservation** :- Dependency Preservation ensures None of the functional dependicies that hold on the original relation are lost.
The sub relations still hold or satisfy the functional dependicies of the original relation.

⇒ **Type of DeComposition** :-

(i) **Loseless Join DeComposition** :-

- Consider there is a relation R which is decomposed into sub relations $R_1, R_2, \ldots R_n$.
- This DeComposition is called loseless Join De Comporition when the join of the sub relations results in the same Relation R that was deCompo sed.
- For lossless join DeComposition, we always have - $R_1 \bowtie R_2 \bowtie R_3 \ldots \bowtie R_n = R$ where $\bowtie$ is a natural join operator

(ii) Lossy Join DeComposition :-

- Consider there is a Relation R which is decomposed into s relations $R_1, R_2, \ldots, R_n$.
- This DeComposition is called lossy join DeComposition when the sub relations does not result in the same relation R was decomposed.
- For lossy join DeComposition, we always have - $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n \supset R$ where $\bowtie$ is a natural join operator.

★ Normalisation :- In DBMS, Database Normalization is a of making the Database Consistant by-

- Reducing the redundancies
- Ensuring the integrity of Data through loseless DeComposit

⇒ Normal Forms :-

(i) First Normal Form (1 NF) :- A given relation is called i- First Normal Form (1 NF) if ea Cell of the table contains only an atomic value. ie. if attribute of every tuple is either Single valued or a nu value.

(ii) Second Normal Form (2 NF) :- A given relation is called only if Second Normal Form (2 NF) if an

- Relation already exists in 1NF.
- No Partial Dependancy exists in the relation.

$A \to B$ is called a partial Dependency if and if 'A' is a subset Some Candidate key and B is a non-prime attribute

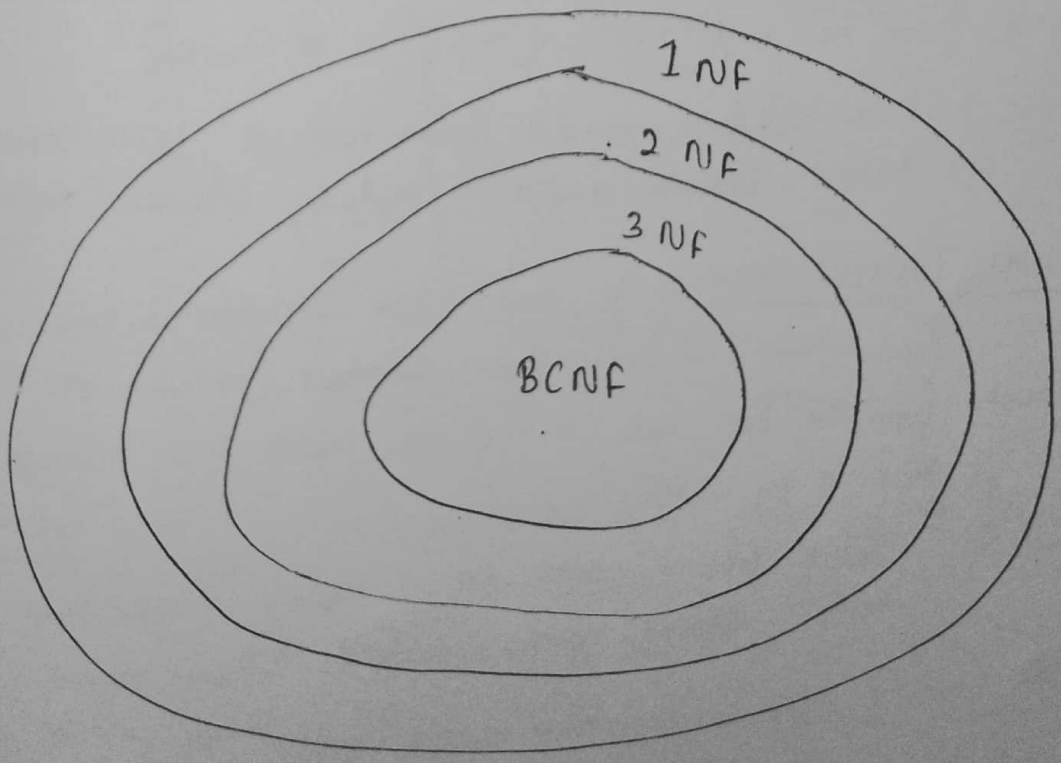
) Third Normal form :- (3NF) A given relation is called in Third Normal form (3NF) if and only if

- Relation already exists in 2NF.
- No transitive dependancy exists for non-prime attributes.

$A \rightarrow B$ is called a transitive dependancy if and only if 'A' is not a super key and B is a non-prime attribute.

(iv) Boyce - Codd Normal form :- A given relation is called in BCNF if and only if

- Relation already exists in 3NF.
- For each non-trivial functional dependancy 'A → B', A is a super key of the relation.



★ Joins :- A JOIN Clause is used to combine rows from two or more tables, based on a related column b/w

(i) INNER JOIN :- The INNER JOIN keyword selects records that have matching values in both tables. Syntax —

Select column_name(s) FROM table1 INNER JOIN table2
ON table1.column_name = table2.column_name;

Ex :- Select Orders.OrderID, Customers.CustomerName FROM Orders INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID

(ii) LEFT (OUTER) JOIN :- The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

Syntax :- Select column_name(s) FROM table_1 LEFT JOIN table2
ON table1.column_name = table2.column_name;

(iii) RIGHT (OUTER) JOIN :- The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the right side, if there is no match.

Syntax :- Select column_names FROM table1 RIGHT JOIN table2
table1.column_name = table2.column_name;

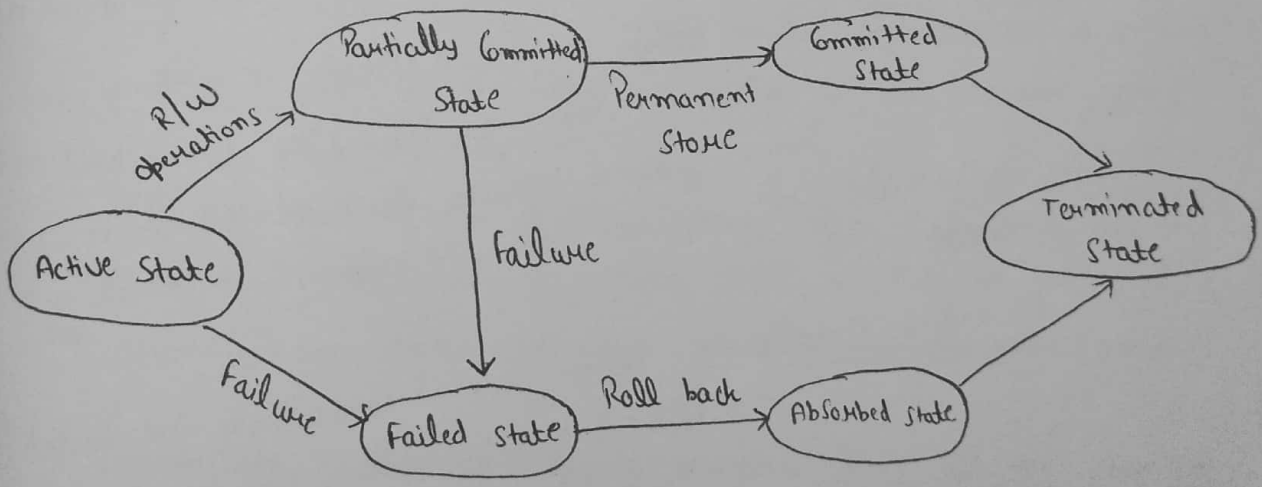
(iv) FULL (outer) JOIN :- The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

Syntax :- SELECT column_name(s) FROM table1 FULL OUTER
table2 ON table1.column_name = table2.column_name
WHERE condition;

Transaction :- Transaction is a single logical unit of work formed by set of operations. Operations in Transaction :-

- **Read operation** :- Read (A) instruction will read the value of 'A' from the Database and will store it in the buffer in main memory.
- **Write operation** :- Write (A) will write the updated value of 'A' from the buffer to the Database

⇒ **Transaction States** :-



1) **Active State** :-
- This is the first state in the life cycle of a Transaction.
- A Transaction is called in an active state as long as its instructions are getting executed
- All the changes made by the transaction now are stored in the buffer in main memory

2.) Partially Committed State :- After the last instruction o transaction has been executed into a partially Committed State.

- After entering this state, the transaction is Consider partially Committed.
- It is not Considered fully Committed because all the made by the transaction are still stored in the b main memory.

3.) Committed State :-
- After all the changes made by the action have been Successfully stored database, it enters into a Committed
- Now, the transaction is Considered to be fully Committed

4.) Failed State :- When a transaction is getting execute active state or partially Committed State Some failures occurs due to which it becom impossible to continue the execution, it enters into a State.

5.) Absorbed State :-
- After the transaction has failed and into a failed state, all the changes ma it have to be undone.
- To undo the changes made by the transaction, it bec necessary to roll back the transaction.
- After the transaction has rolled back completely, it ent absorbed State.

6.) Terminated State :-
- This is the last state in the life cycle Transaction.
- After entering the Committed State or absorbed State, the Transaction finally enters into a Terminated State where it

ACID Properties :- To ensure the Consistancy of the database, Certain properties are followed by all the transactions occuring in the System. The properties are called as ACID Properties of a transaction.

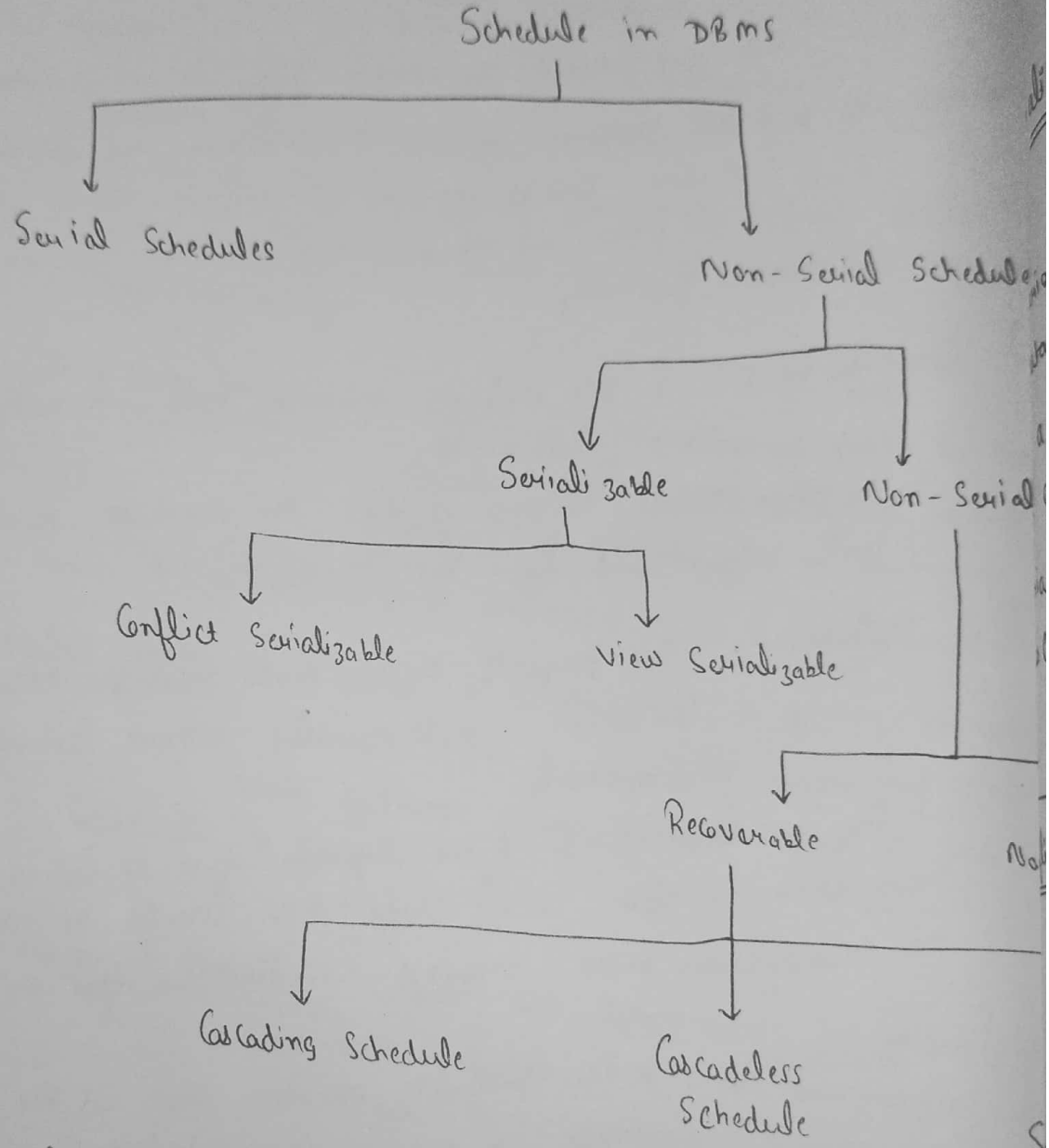
1) **Atomicity** :-
- This property ensures that either the transaction occurs Completely or it doesnot occur at all.
- In other words, it ensures that no transaction occurs partially.

2) **Consistancy** :-
- This property ensures that integrity Constraints are maintained.
- In other words, it ensures that the database remains Consistent before and after the transaction.

3) **Isolation** :-
- This property ensure that multiple transactions Can occur simultaneously without Causing any in Consistancy
- The resultant state of the System after executing all the transactions is same as the state that would be acheived if the transactions were executed Serially one after the other.

4) **Durability** :-
- This property ensures that all the changes made by a transaction after its Successful execution are written Succesfully to the disk.
- It also ensures that these changes exist permanently and are never lost even if there occurs a failure of any kind

★ **Schedules** :- The order in which the operations of mu transactions appear for execution is called '



① **Serial Schedules** :-
- All the transactions execute Serially after the other.
- When one transaction executes, no other tr is allowed to execute.
- Serial Schedules are always Consistant, Recoverable Cascadeless and Strict

Non- Serial schedules :- • Multiple transactions executed Concurrently.
• Operations of all the transactions are inter leaved or mixed with each other.
• They are not always :- Consistent, Recovarable, Cascadeless and strict.

Serializability :- Some non- Serial schedules may lead to inConsistency of the database. Serializability is a Concept that helps to identify which non- Serial Schedules are Correct and will maintain the Consistency of Database.

• If a given non- Serial schedule of 'n' transactions is eQuivalent to Some Serial schedule of 'n' transactions, then it is Called as Serializable Schedule and they are always :- Consistent, Recoverable, Cascadeless and strict.

Types of Serializability :-

• Conflict Serializability :- If a given non- Serial Schedule Can be Converted into Serial Schedule by Swapping its non- Conflicting operations, then it is called a Conflict Serializable Schedule.

• View Serializability :- If a given Schedule is found to be Viewed as eQuivalent to Some Serial Schedule, then it is Called a view Serializable Schedule.

⇒ **Non-Serializable Schedules** :- A non-serial schedule is not serializable is called non-serializable schedule. This sch is not guaranteed to produce the same effect as produ by some serial schedule on any consistent database. The may or may not be consistent, may or may not be recovera

⇒ **Irrecoverable Schedules** :- If in a schedule, a transactio a dirty read operation from an committed transaction and commits before the transaction which it has read the value then such a schedule is know Irrecoverable schedule.

⇒ **Recoverable Schedules** :- If in a schedule, a transaction per a dirty read operation from an un-itted transaction and its commit operation is delayed till th uncommitted transaction either commits or roll backs then Schedule is known as a Recoverable schedule.

Types of Recoverable Schedule :-

(i) **Cascading Schedule** :- If in a schedule, failure of one transac causes, several other dependent transac to roll back or abort, than such a schedule is called as Casc Schedule or Cascading rollback or Cascading abort.

(ii) **Cascadeless Schedule** :- If in a schedule, a transaction is no allowed to read a data item until t last transaction that has written it is committed or aborte then such a schedule is called as a Cascadeless schedule.

(iii) **Strict Schedule** :- If in a schedule, a transaction is neither allowed to read nor write a da item until the last transaction that has written it is com or aborted, then such a schedule is called as a strict sch

SQL

DDL :- It is a short name of Data Definition Language, which Deals with Database schemas and descriptions of how the Data should reside in the database.

- CREATE
- ALTER
- DROP
- TRUNCATE
- RENAME

DML :- It is a short name of Data manipulation language which deals with data manipulation and includes most common SQL statements such as

- SELECT, INSERT, UPDATE, DELETE, MERGE.

- It is used to store, modify, retrieve, delete and update data in database.

DCL :- It is short name of Data Control language which includes Commands such as GRANT and mostly concerned with rights, permissions and other controls of the Database System.

- GRANT :- allow users access privileges to the database.
- REVOKE :- Withdraw users access privileges given by using the GRANT Command.

TCL :- TCL is a short name of Transaction Control language which deals with a transaction with in a database

- COMMIT :- Commits a transaction.
- ROLL BACK :- roll back a transaction in case of any error occurs.
- SAVE POINT :- to roll back the transaction making points with in groups SQL

- SQL is a standard language for storing, manipulating and retrieving data in databases.

SQL QUERIES

① Create table trainees (roll no int primary key, tname varcha
per float, branch varchar (10), mobile varchar (12))

② || Full Insertion
insert into trainees values (101, 'Arun', 77.88, 'cse', '

③ || Partial Insertion
insert into trainees (roll no, tname, mobile) values (103, 'Raman

④ || Updation
Update trainees Set per = 81.21 where branch = 'cse'
Update trainees Set per = per + 5 where per < 94

⑤ || To delete all records
delete from trainees

⑥ || Delete a record
delete from trainees where roll no = 103

— — — — — — Retrieval | Searching | Fetching — — — — — — —

⑦ || Fetching all records
Select * from trainees
Select * from trainees where branch = 'cse'
Select * from trainees where tname = 'Arun' and branch = '
SELECT tname, roll no. from trainees where branch = 'cse

⑧ || For unique values in a column
Select distinct branch from trainees
Select max(per) branch from trainees
Select max(per) as 'Highest' from trainees where branch = 'c

|| Nested Queries

Select * from trainees where bar = (select max(bar) as 'Highest' from trainees where branch = 'cse')

Select avg(bar) as 'Avg', min(bar) as 'min', max(bar) as 'Maximum' from trainees

* *

|| For Searching in Ascending order

Select * from employee order by name

|| For Searching in Descending order

Select * from employee order by name desc

|| For Searching top 2 employee having maximum Salary

Select * from employee order by total desc limit 2

|| For Searching the employee having maximum Salary except who is having highest (limit 1,2 ---> 1 is used for leaving top)

Select * from Employee order by total desc limit 1,2

|| For Searching whose name starts with m

Select * from employees where ename like 'm%'

|| For Searching whose name contains m

Select * from employees where ename like '%m%'.

|| For Searching a name having 4 letters

Select * from employees where name like '____'

|| For Searching a name ends with m

Select * from employees where name like '%m'

(18) // For Searching a name having m on 3rd position

Select * from employees where name like '__m%'

(19) // For current date and time

create table library (book varchar (20), doi date , dom date
toi time)

insert into library values ('real java' , CURRENT_DATE ( ), Date_o
(current_date ( ), interval 5 day) , CURRENT_ TIME ( ))

(20) IN → allows you to specify multiple values in WHERE clause.

Select * from employees where department IN('Accounts', 'Physic

(21) BETWEEN → allows you to selects values within given range, valu
can be numbers, texts and dates.

Select * from Products where price BETWEEN 10 AND 20

(22) TRUNCATE Table → Used to delete data inside a table, but not tabl
itself.

TRUNCATE TABLE Table_name ;

(23) ALTER Table → Used to add, delete or modify columns in an existing table and also used to add and drop variou
Constraints on an existing table

ALTER TABLE Table_name ADD COLUMN Colum_nname data

(24) ALTER TABLE Table_name MODIFY COLUMN Column_name data