Key terminologies:

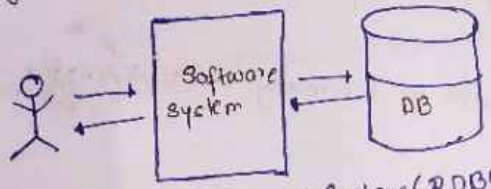
Data: The known facts that can be recorded and have implicit meaning
eg - known facts about a student like student ID, student name, branch percentage.

Record: A collection of Data items and

[S1 x CSE x 9.5]
[S2 x IT x 8.9] ⟵ Records

DataBase: An organised collection of Data or collection of related Data is called Database.

DataBase Management System (DBMS) Software system to facilitate end user to manage underlying DataBase



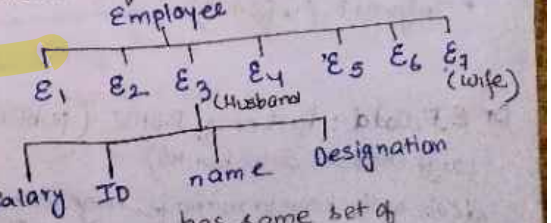
Operation
Creation of Tables
Insertion of Tables
Deletion of Tables
Updation System

Relational DataBase Management System (RDBMS): In a RDBMS data is stored in form of Tables and Software used to facilitate and manage it is called RDBMS

DataBase Organisation: Different ways of arrangement of DataBase records.

HDBMS: Hierarchial DBMS → Data is stored in tree like structure (logical container)
NDBMS: Network DBMS → Data is stored in form of graph
RDBMS: Relational DBMS → Data is stored in Tabels
OODBMS: Object Oriented DBMS → Data stored in form of Object
ORDBMS: Object Relational DBMS → Data stored in form of Object/Table

HDBMS:
Data is arranged in hierarchal manner.
eg: Some employees in an organisation
Based on One to many relationship.
A node has single Parent (inverted tree)

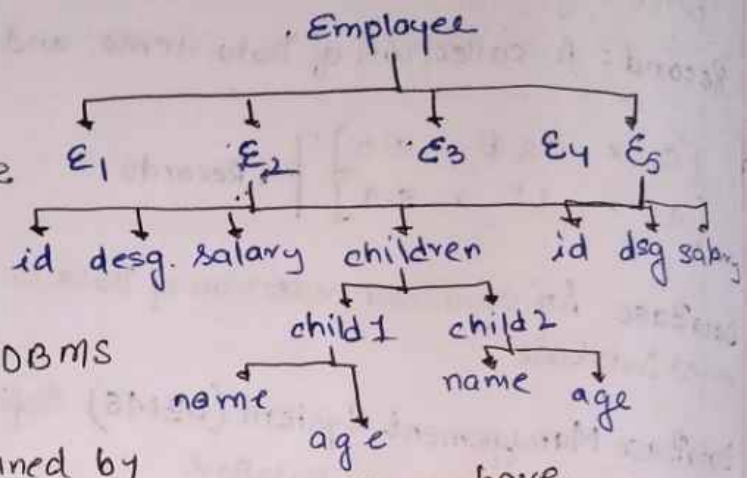


Each employee has some set of attributes.

- may have redudant data. eg: children feild fore both husband and wife are same.
- Data is arranged in a tree like structure

Disadvantage: • Retriving information is very time consuming as we move from Root to children

- A child cannot have default parent (Data redundancy) as it uses one to many relationship.



NDBMS:

NDBMS is based on graph Data structure
Uses many to many relationships
Concept of pointer is used
Data arrangement is slightly similar to HDBMS but is free of Data redundancy.
Redesigning HDBMS to get NDBMS, obtained by redesigning HDBMS by many to many relationship so a child can have multiple parent.

Advantage: Very fast in accessing data as internally uses concept of pointer for locating data

RDBMS

Software system that facilitate management of Data in relational Database.
Data is stored in form of tables Ex.
eg: - SQL server (ms)
- Oracle (Oracle)
- DB/2 (IBM)
- MySQL (oracle)
- Sybase (sybase)
- Informix (informix)

Dr E.F. Codd: Father of RDMS (published article 'A relational mode for data for large shared data banks)
deals with how to manage huge volume of data in more efficient manner based on mathematicall concepts • Relational Algebra
• Relational Calculus

In the articles Codd proposes 12 rules. If a DBMS satisfies atleast 6 out of 12 rules then that DBMS is said to be a RDBMS.

Relational DataBase Management System.

Relation: Any subset of a cartesian product. Let two sets S_1 and S_2

$S_1 = \{1, 2\}$ Cartesian Product $= S_1 \times S_2 = \{(1,a), (1,b), (2,a), (2,b)\}$

$S_2 = \{A, B\}$

Set of ordered collection of elements of both side and any subset of it is Relation

Let $R_1 = \{(1,a), (2,b)\}$ $R_1 \subseteq S_1 \times S_2$

$R_2 = \{(2,a)\}$ $R_2 \subseteq S_1 \times S_2$

$R_3 = \{(1,a)\ (1,b)\ (2,a)\ (2,b)\}$ $R_3 \subseteq S_1 \times S_2$ (Equal Subset)

No of Possibilities $= 2^{m \times n}$ m = Cardinality of S_1, n = Cardinality of S_2

for $S_1 \times S_2 = 2^4 = 16$

How many n-ary relations can be formed over n sets having p elements each.

No. of sets = n

$S_1 = \{p \text{ elemt}\}$ $S_2 = \{p \text{ elemts}\}$ ..... $S_n = \{p \text{ elements}\}$

No. of total relation = 2^{p^n}

or

R	A	B
R_1	1	a
R_2	1	b
R_3		c

Attributes: (A, B) Each colums

Records / Tuples: Each row in the relational tables

Mathematically R(A, B)

Degree: No. of attributes eg: 2 for R

Cardinality: No. of tuples

Domain: Set of values.

Schema: Structure of the table.

R (A = integer, B = characters)

Properties of a Relation

(1) A relation should not contain duplicate tuples

R	A	B
a	1	a ✓
b	2	b ✓
	1	a ✗ Not allowed

(2) Attributes can appear in any order

R	A	B
	1	a
	2	b

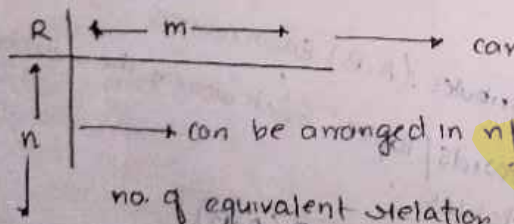
R	B	A
	b	2
	a	1

(3) Tuples can also appear in any order

R	A	B
	1	a
	2	b

R	A	B
	2	b
	1	a

Question If any relation has degree M and cardinality N then how many equivalent relations can be formed.



no. of equivalent relation = $m! \times n!$

Relational Data Base Design

eg: Contruction of House

(1) Requirement
(2) Prepare the plan
(3) Prepare the prototype
(4) Physical Construction of House

Data Base Design

(1) Requirement Specification
(2) Conceptual Design (Entity-Relationship Model)
(3) Logical design (Functional dependencies, SQL, Relational Algebra)
(4) Physical Design (B-Tree, B+ tree, indices) (File structure, storage etc)

In case of multi user Environment Concurency Control and Transactional relationship is important

Entity Relationship Model (ER-Model)

Useful Tool to design relational Data base
Utilizes E-R diagram to do so

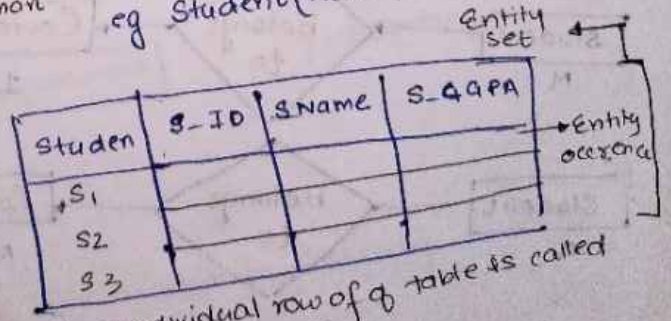
ER diagram: Pictorial representation of domain knowledge in terms of Entity Relationships and their attributes.

ER Diagram
- Entity
- Relationship
- Attributes

Entity: Any thing or object (real world thing or object) having independent existence
It may be physical or conceptual
eg: students of CSE branch

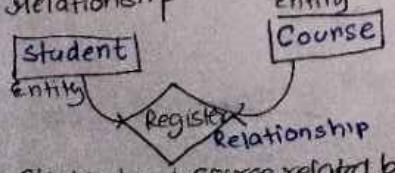
Entity set: Collection of Entities
It is represented in form of tables
eg Student (Name, CGPA, S_ID)

Entity
- Physical → Tangible eg student human
- Conceptual → Intangible eg sports



Each individual row of a table is called entity occurences.

Relationships: Association among entities is called relationship



Student and course related by register.

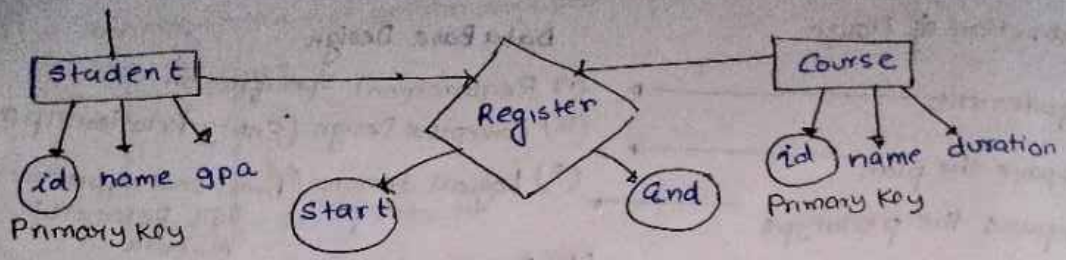


Table for above relationship

Register	S_ID	C_ID	StartDate	EndDate

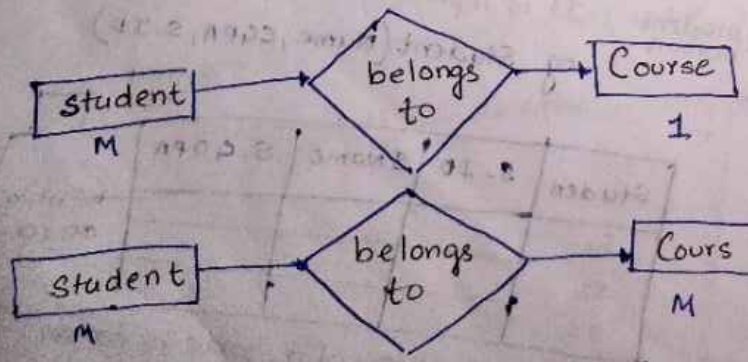
→ Each data is Relationship occurence

The relationship table is called Master table which store Metadata

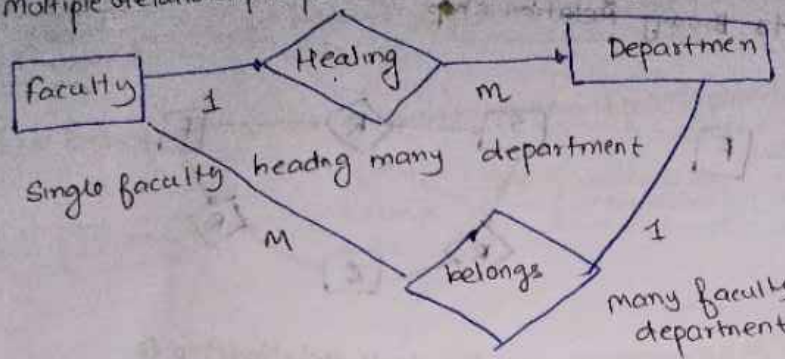
Metadata: Data About Data is meta Data

Types of Relationships

1. One to One (1-to-1)
2. Many to One or One to Many (M-to-1 or 1-to-M)
3. Many to Many



Multiple relationship is possible b/w two entities

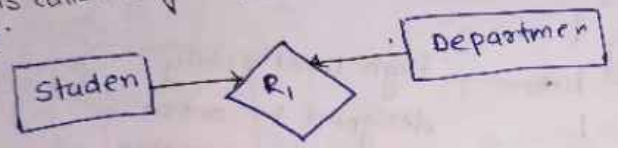


many faculty belongs to single department

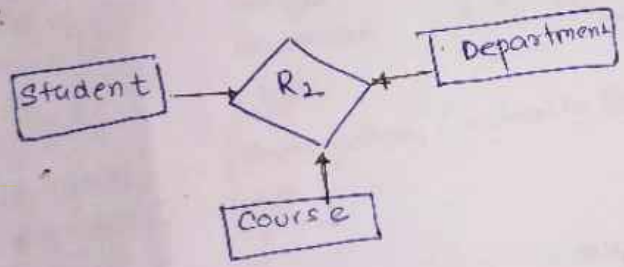
Degree of A Relationship: No of participating entities in a relationships. is called degree of a relationship

Degree of $R_1 = 2$
Binary Relationship.

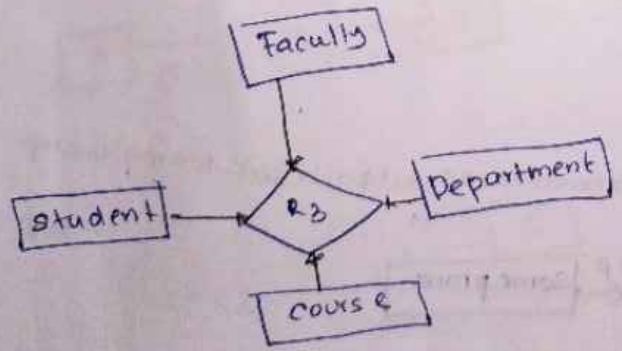
eg:



eg:



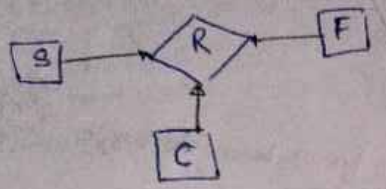
Ternary Relationship
No of entity participating : 3

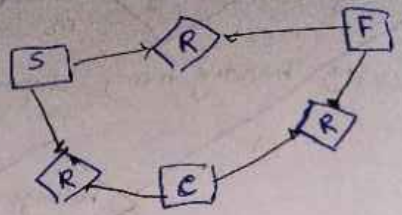


Quaternary relationship.
No. of entities participating : 4

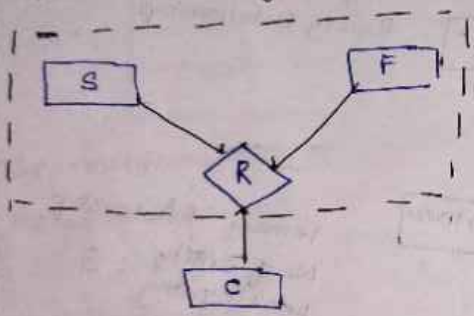
No of Participating Entits : M ⟶ M-ary relationship

Converting Ternary to Binary Relation ship





Conversion using Aggregation: Abstraction by which relationship is combined as a high level entity



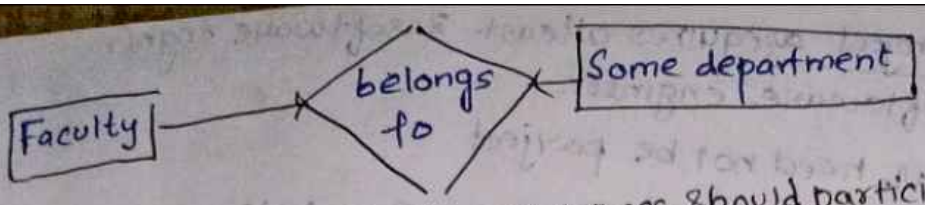
High level entity designed to act as a single relationship.

Constraints of ER Diagram

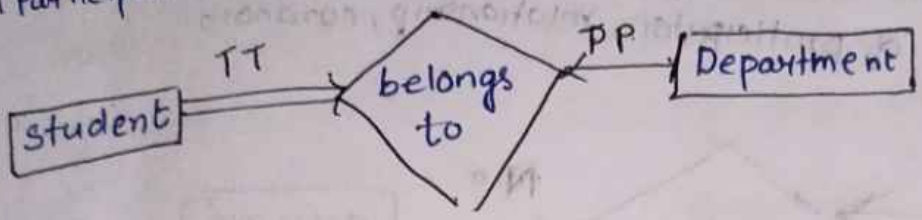
1. Participation Constrain → Partial / Total
2. Cardinality Ration
3. Keys

Partial Participation All entity occurences may not participate to a relationship

Some students register some process



Total Participation: All entity occurences should participate to a relationship.



All students belongs to some department

Cardinality Ratio/Cardinality: It describes maximum no. of instances that an entity occurence participating to a relationship

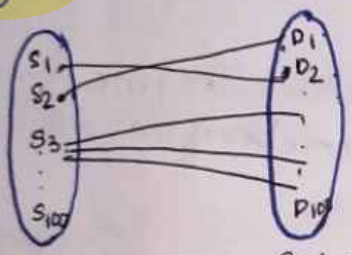
Cardinality Ratio of $S_3 = 3$ (Max 3 times occurence

Cardinality Ratio for S

Cardinality Ratio for D = 1 (Max ocurence is 1)

0 → Parital Participation

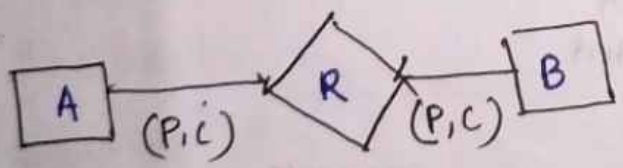
1 → Total Participation

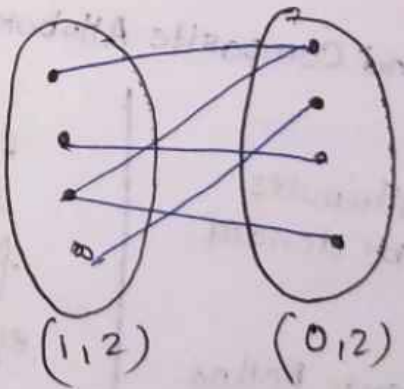


Set of students (0,3)

Set of Departments (1,1)

Multiplicity → (Participation, Cardinality Ratio) it represents the way entities are related.

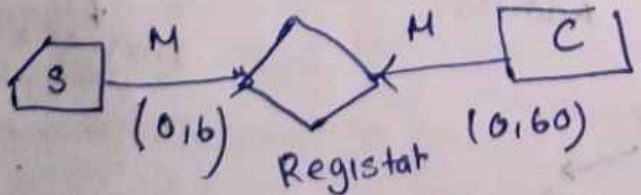




(1,2) (0,2)

Some student register some course

Que

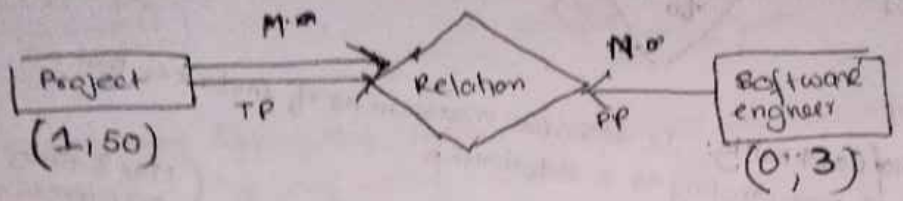


A student can max register to 6 courses

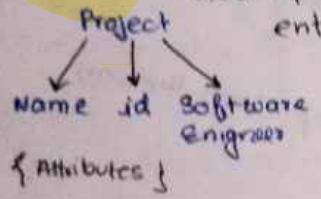
All students need not register courses but they can go maximum upto 6 course

All the courses need not be register by the student if at all they register max 60 can take up one course.

Que Each and every project requires atleast 2 software engineer and max of 50 software engineer.
All software engineer need not be project.
If the participate, they can go maximum of 3 projects.
Find the type of of participaton, relationship, cardnality



Attributes Any entity is represented by set of attributes. They are descriptive properties possessed by each member of an entity set.

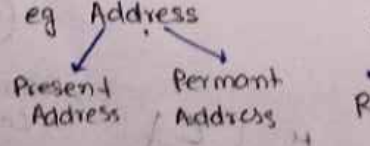


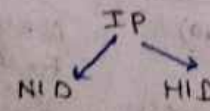
Attributes
- Simple and composite
- Singe valid and Multivalid
- Stored and Derived

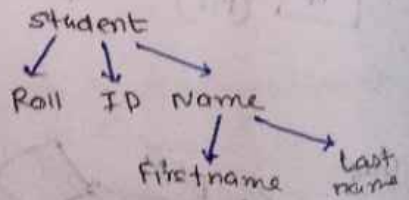
Simple and Composite Attributes

Simple
Simple Attributes cannot be devided furthur
eg: Student Rollno.

Composite
Attributes that can be devided furthur
eg Address







Single Valued and Multivalued Attribute

Single valued

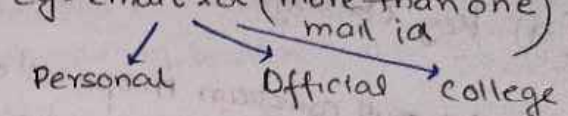
Always hold a single value

eg: DOB, password,

Multi Valued

Multivalued attributes holds multiple values at a time

eg: email id (more than one)

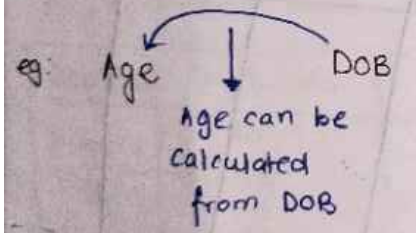


: Roll no for dual degree course

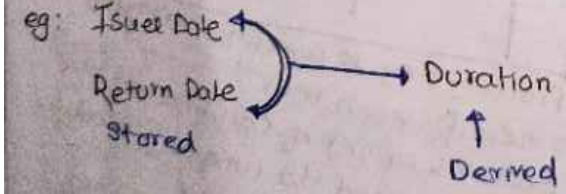
Stored and Derived Attributes

Holds the value and supplies the value to attributes

The attributes that are derived from stored Attributes.



Derived attributes gets their value from stored attributes



Store attribute, holds the value and supplies the value to derived ~~attribute~~ attributes

Data Integrity Constrans: Restrictions on a table so as to maintain proper data in the table.

eg: (a) Not Null
(b) Unique
(c) Check
(d) Key Constraints

Not Null: Not null constrain imposed on a table coloum then it can't allow null values but it may allow duplicate value

Unique: It will allow unique values but not duplicate values.

check: Checks the conditions for a value (Ball =₹3000 and Bal<=500)

Key Constrain: Single or a set of attribute used to uniquely identify each row or more than one row is called Key

Key Constrains
- Primary Key
- Super Key
- Candidate Key
- Alternate Key
- Composite Key
- Foreign Key

Primary Key:

S_ROLL	S_Name	S_Branch	S_CGPA
1			
5			
2			
6			
8			
:			
:			

Primary Key is used to uniquely Identify each row
(Colums or set of Columns in a table that is used to uniquely identify touples/rows in table
eg s_id in the case of student.

Candidate Keys
* When we use two keys combinely to uniquely identify the rows we call it ~~pr~~ candidate Key.
eg: S.name and S.branch.

Out of candidate Keys we seclect one primary Key

PK ⊂ CK

R (ABCDE)

AB → C

AC

Super Key: Primary key with some redundant attributes is called super keys. A super key is Candidate Key with additional Attribute

C_K → (st_id) → Primary_Key

→ st_name + std_branch (Alternate key)

Minimal form of super key is candidate key

Possible Super Keys → St_id + st_cgpa
→ st_name + std_branch + std_cgpa
→ St_id + std_name + std_branch.

Alternate Keys: Remaing Candidate Keys except Primary Key are called Alternate Key

C_Keys → P_K (only one key primary key).
→ Alt_Key (Rest Keys are alternate keys)

Foreign Key Foreign keys are columns of a table that points to primary key of another table. They act as cross reference between table

Dept_No	Dept Name	Loc
10		
20		
30		
40		
50		

Primary Key → 20

Master Table

Points to the primary key of Dept table

Emp

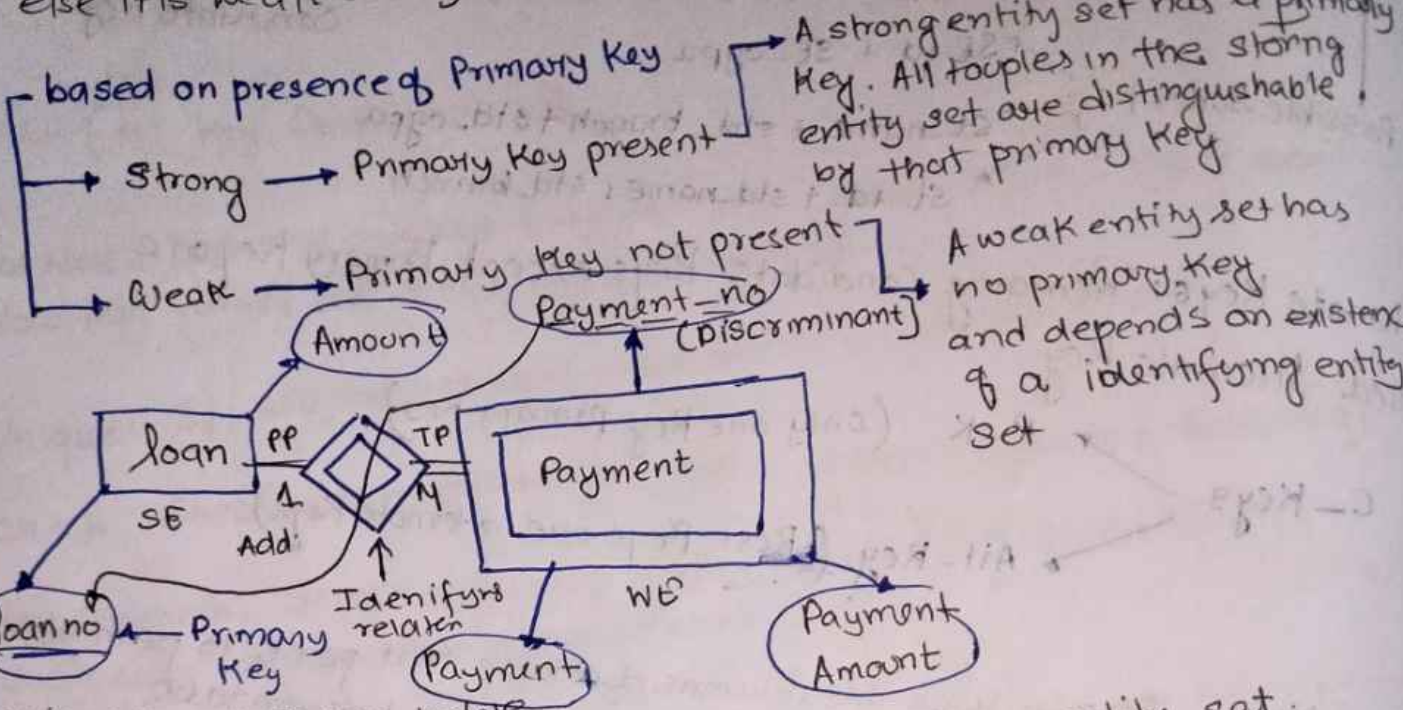
Emp_no	Emp_name	emp_salary	Emp_Job	Dept-no
Primary Key				has on 5 unique values as candidate of Dept

Dept-no ← Foreign Key

Child / Slave / Detailed Table

Classification of Entities

If table / Entity set consits a primary key it is a stoong entity
else it is weak entity



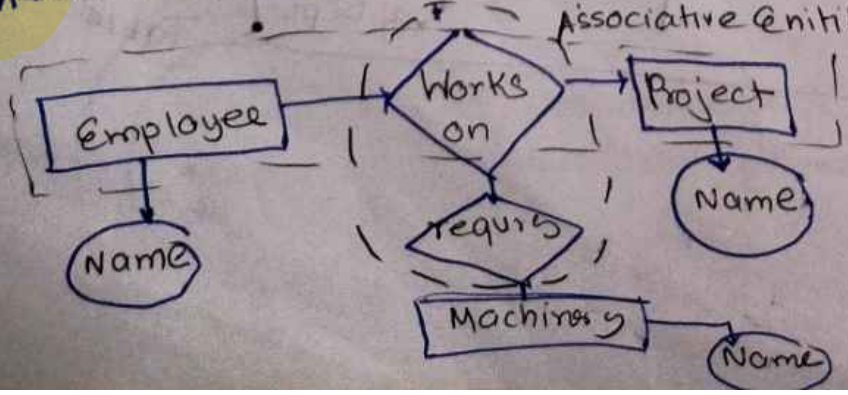
PK: loan no + Payment_no

In this case weak entity set depends on strong entity set.
identity and attributes
Add it with the primary key of storng Entity.

The attribute of the weak entity that has been add to primary key of strong entity is called **Discriminant Attribute**

Relationship b/w weak and storng entity is establish using **Identifying Relationship**.

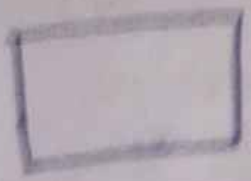
Associative Entity: looks like a entity but actually is a relationship.



E-R diagram

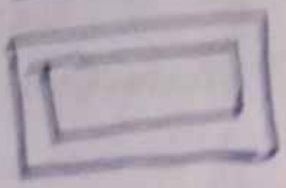
Pictorial representation of domain knowledge in terms of entities, relationships and attributes

Notations



Strong Entity (SE)

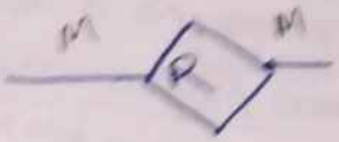
One to One Relation



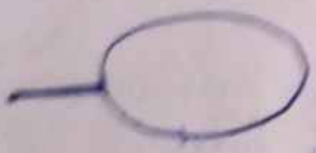
Weak Entity (WE)

One to many

Associative Relationship (AE)



Many to many

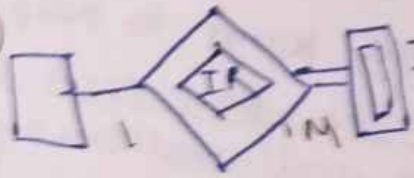


Attribute

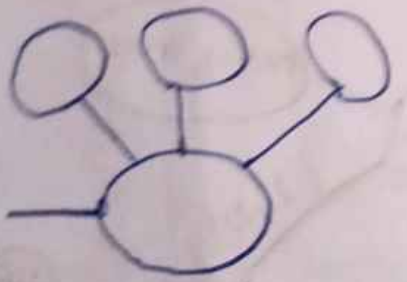
PP = Partial

TP = Total

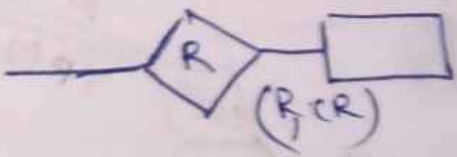
Primary key (PK)



Identy relation

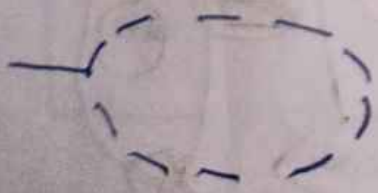


Composite Attributes (CA)



Multiple

multivalid Attributes (MVA)



Derived Attribute (DA)

Discriminant

variables

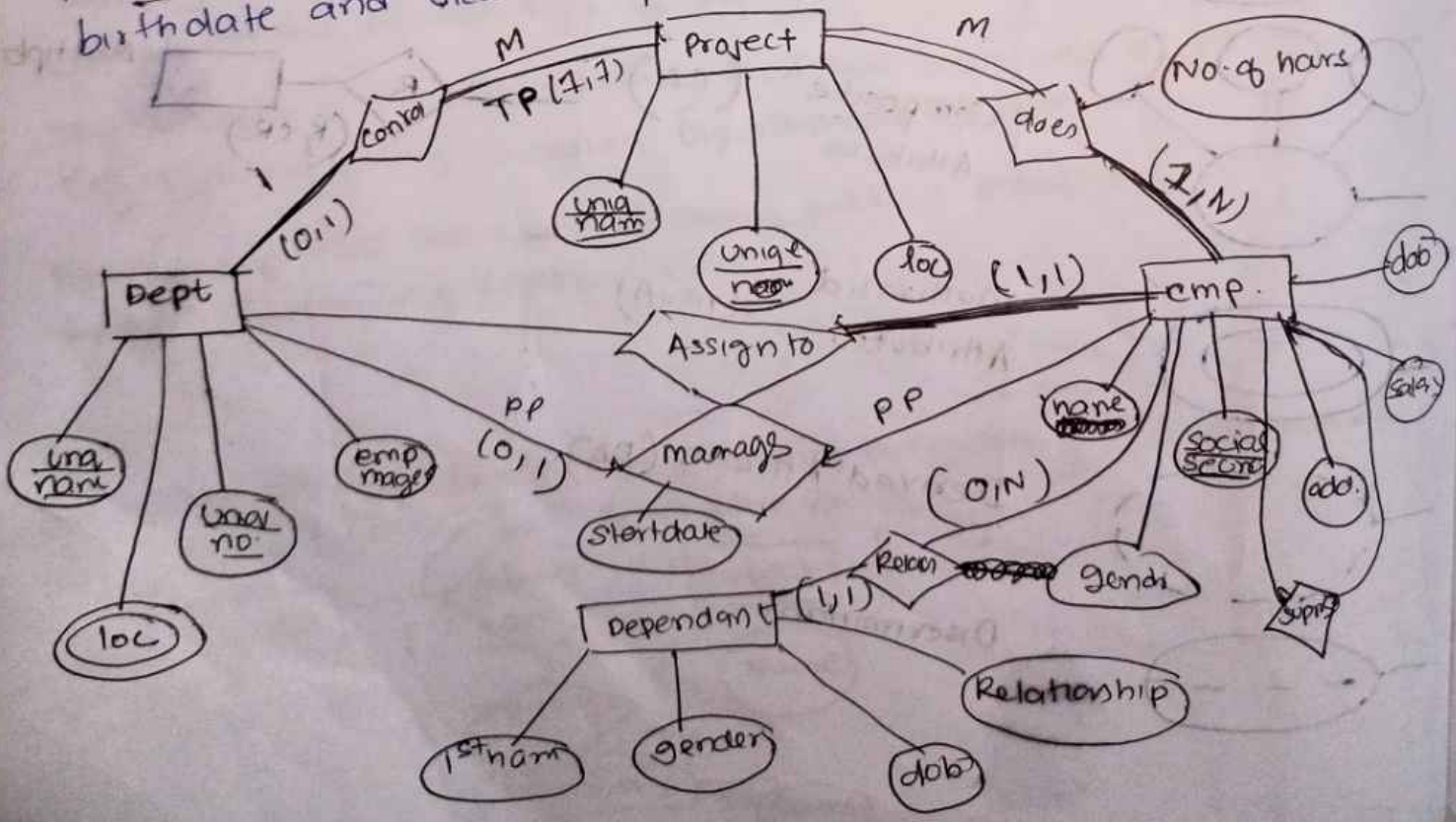
Construct A ER diagram

The orgpany is organised into departments. Each department has a unique name, unique no. and particular employee who manges into depart. We keep track of start date when that employe began manging the department. Dept. may have several location.

Dept controls a no. of projects. Each of which has a unique name, unique no and a single location.

We store each employees name, social security no, address, salary, gend and dob. An employee is assign to eactly one dept but may work on several projects which are not necessarily controlled by the same dept. We keep track no of hours per week that employee works on each project. We also keep track of direct superviser of each employee

We want to keep track of each employee for insurance purposes. We keep each each dependence 1st name, gender, birthdate and realtionship to employee



Conversion of ER-Diagram Relational Tables.

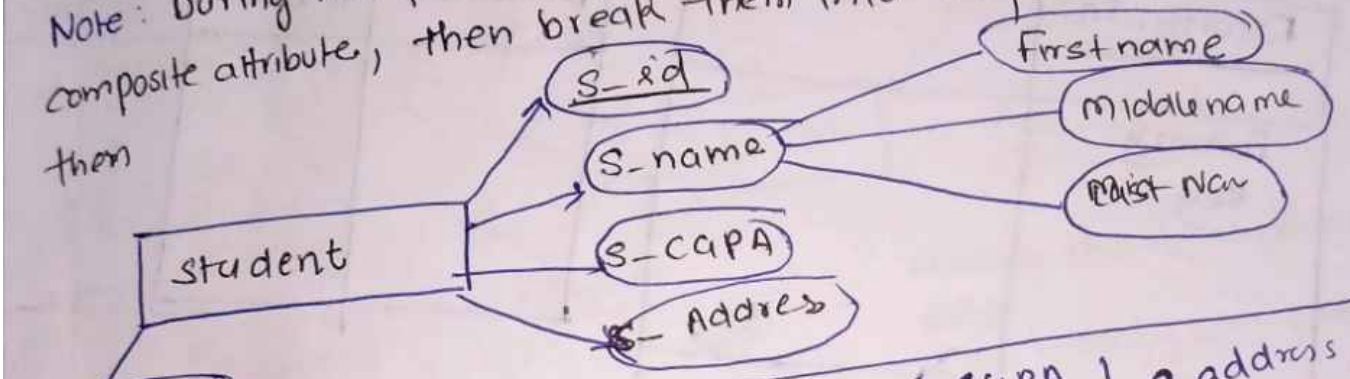
① Conversion of strong Entity to table (strong Entity → table)

(a) For strong entity set, create a separate table with same name

(b) include all attributes of storng entity set in the table

(c) Select a primary key for the table

Note: During this process ignore multivalid attribute. If we have composite attribute, then break them into simple attbute the include then



student

S_id	S_name			S_CGPA	S_address
Primary Key	First nam	Midd nam	last name		

S_emails: multivalid attribute

emails	S_ID
@	Forein Key
@	
@	

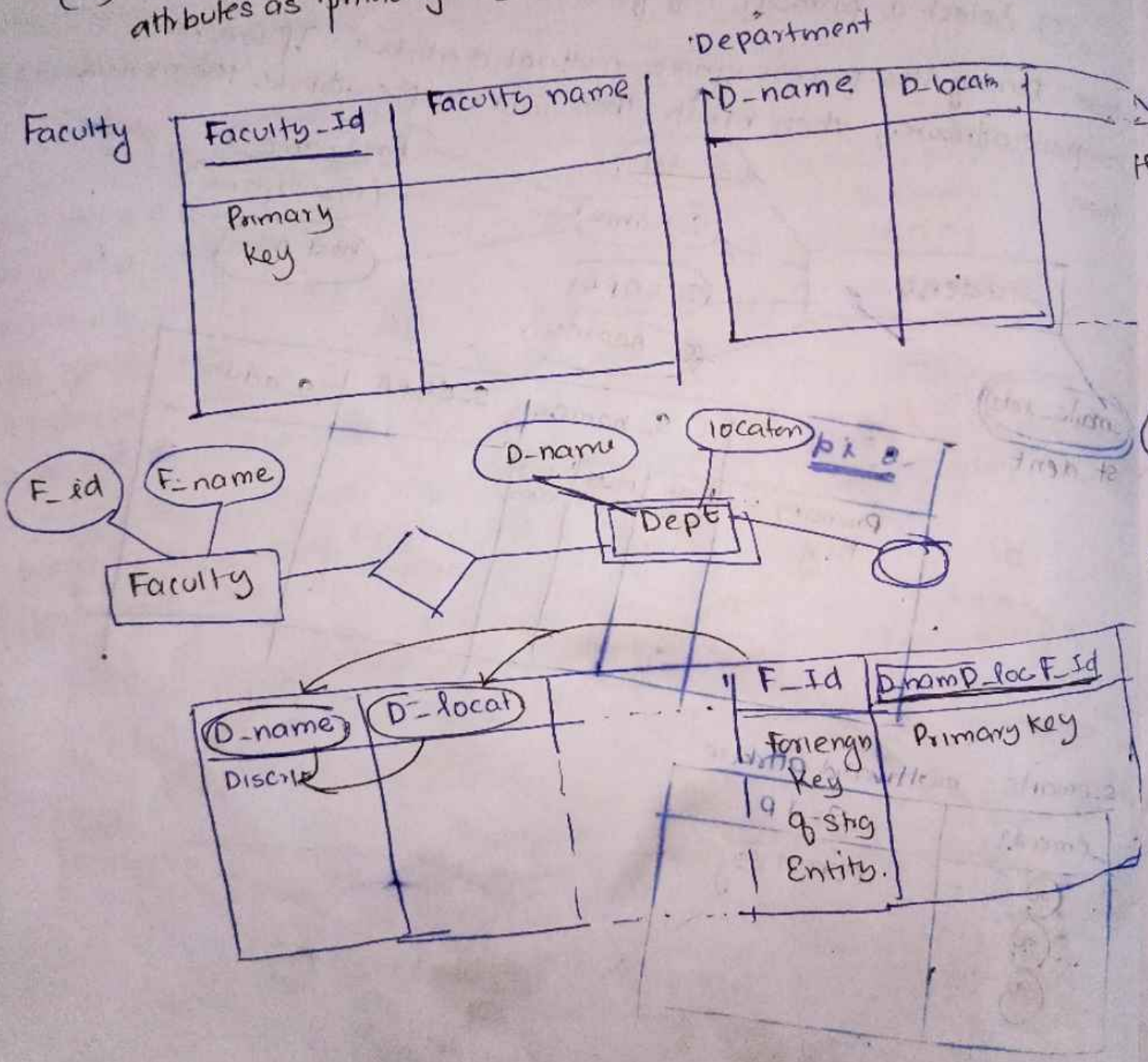
(2) Conversion of Weak entity set into equalvalent table

(a) For each weak entity set, create a separate table with same nam

(b) include all attributes in table

(c) Also include primary key of strong entity set as Foreign key

(d) Declare the combination of Foreign Key and Discriminator attributes as primary key.

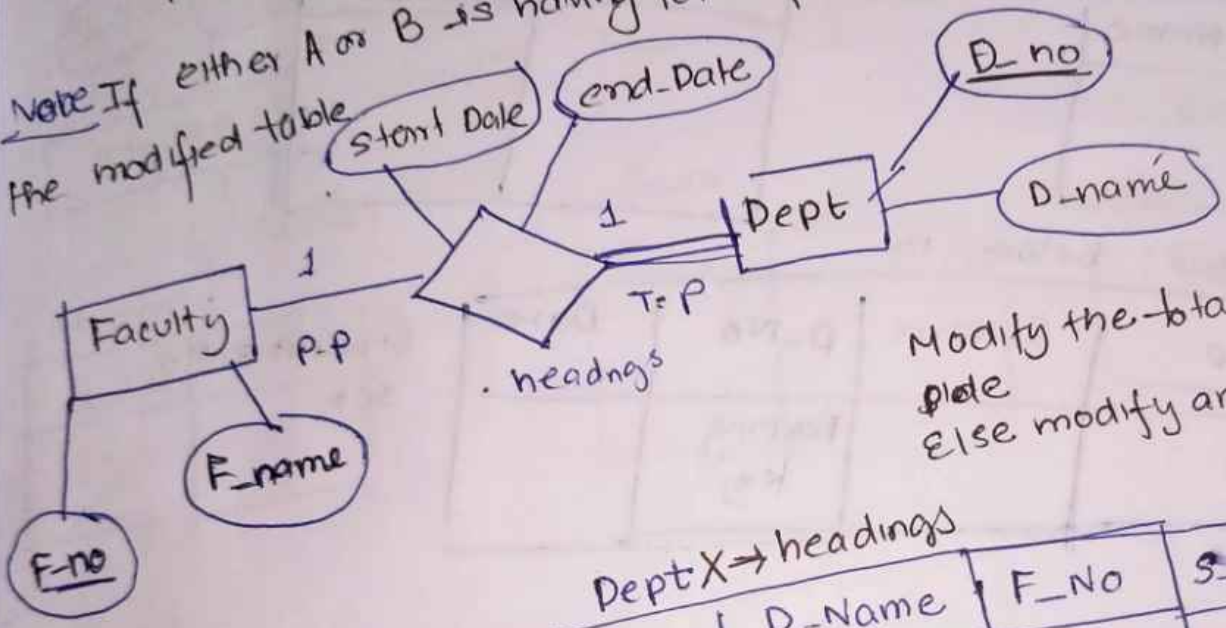


(3) Conversion of 1-to-1 relationship

(a) For each one to one relationship say between A and B modify either A or B and include primary key of other tabel as a foreign key

(b) If the relation p consist of of attributes then include them also in modified table

Note If either A or B is having total participation then that should be the modified table



Modify the total participation side
Else modify any one

Faculty

F_no	F_name
Primary key	

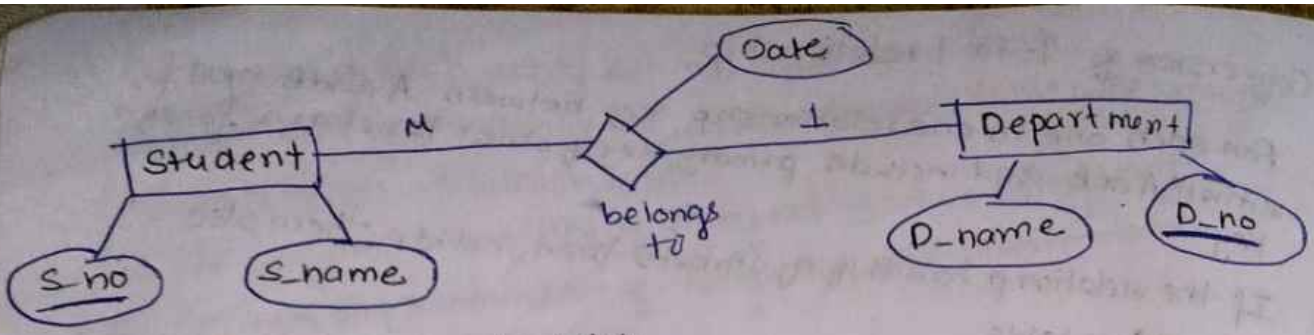
Dept X → headings

D_NO	D_Name	F_NO	S_Date	E_Date
Primar Key		Foreng Key		

(4) Convension of 1 to Many Relationship

(a) For each one to many relationship say b/w A and B, modify M side table and include primary key of one side table as foreign key

(b) If the relationship consists of attributes, include them also in the modified table.



modify M side table

S_No	S_Name

D_name	D_no

Modified table: belongs to

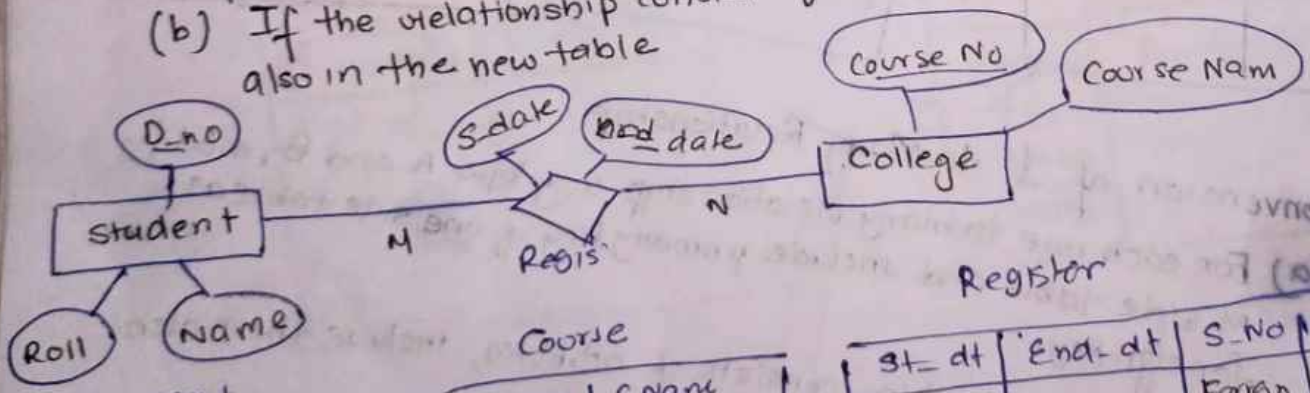
S_NO	S_Name	D_NO	Date
		Foreing Key	

Student entity set

(5) Conversion of Many to Many Relationship.

(a) For each M-M relationship, Create a separate table and include primary keys say A and B as Foreign keys

(b) If the relationship consist of attributes include them also in the new table



Student

Roll	Name	D-no
Primary key		

Course

C_No	CName
Primary Key	

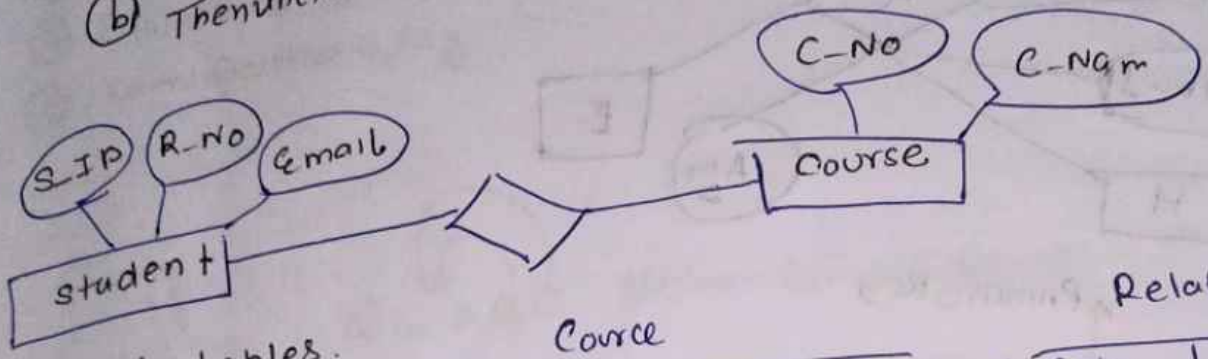
Register

St_dt	End_dt	S_NO	C-No
Foreing Key		Foreign key	Foreign key

(6) Conversion of Multivalued Attributes

(a) For each mVA create a speparate table and include multiple values of this attributes as simple attribute

(b) Thenunclud R.key of orignal table as Foriegn key of Newtable



4 tables.

student

S_ID	S_Roll	S

Cource

C_No	CName.

Relation

CNO	SID

Email

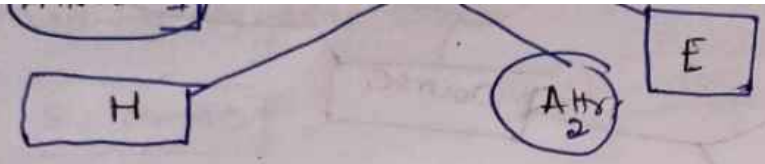
ids	S_ID

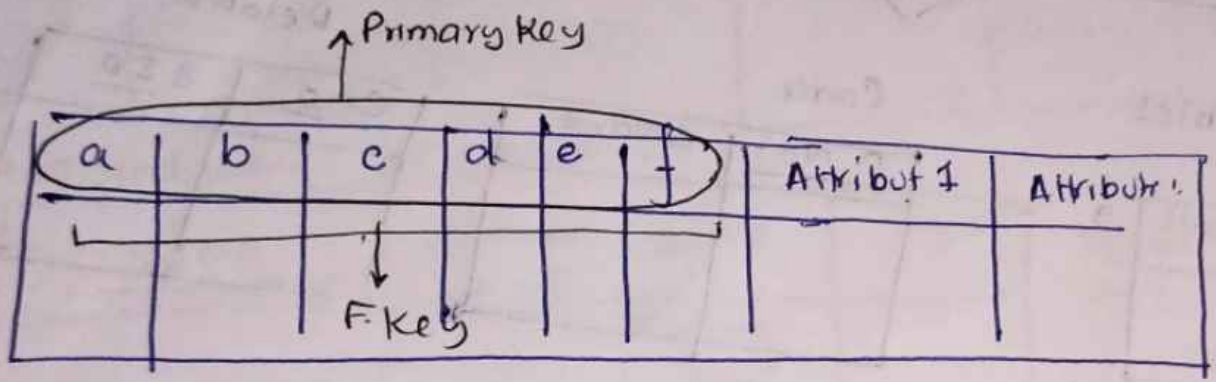
(7) Conversion of N-ary Relationsp

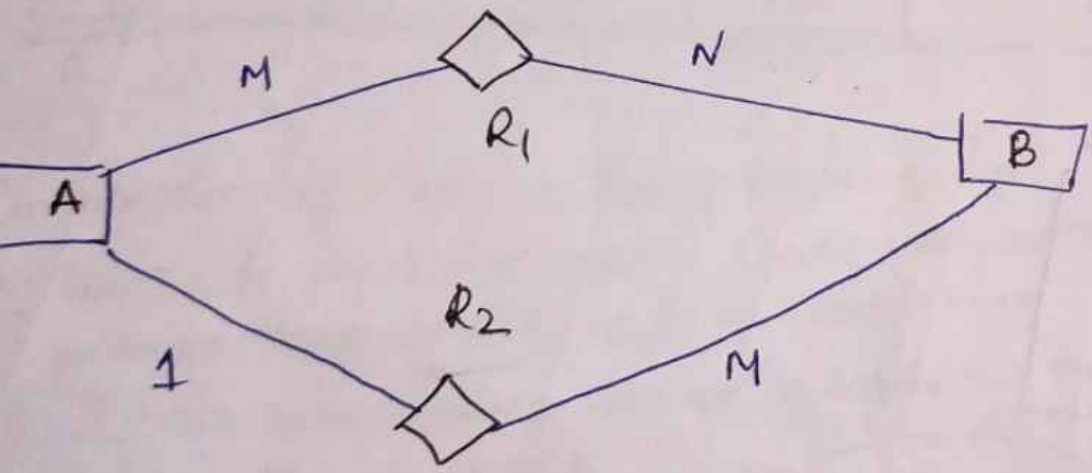
(a) For each n-arry relationship create a sepeate table and include P.Keys of all tables as F.Key

(b) If the relationship consist of attribute include them in newtable

(c) Declare combination all Foriegn keys in new table as Primary Key





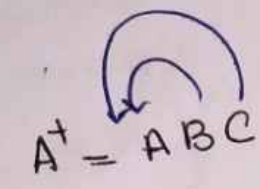


in: 3 tables

ax: 4 tables.

Closure Set of Attributes

① Identification of addnal f.d
② Equivalence of FDs
③ Identification of Key



$A^+ = ABC$ Attributes B and C depends on A

$A \longrightarrow B$

$B \longrightarrow C$

Procedure:

R (A B C D E)

FP : { $AB \rightarrow C$
$BC \rightarrow AC$
$D \rightarrow E$
$CE \rightarrow B$ }

$[AB^+] \rightarrow \underline{A\ B}\ C\ D\ E$

Any combination of attributes can be arived from AB

$(CE)^+ \longrightarrow CEBAD$

R (A B C D E)

$A \longrightarrow BC$

$CD \longrightarrow E$

$B \longrightarrow D$

$C \longrightarrow A$

C

$B^+ \longrightarrow BD$

$(BC)^+ \longrightarrow BCDEA$

Identification of addn f-d :

⇒ R(A B C D)

Fd: { A → B
BC → D }

Does AC derives D AC → D

A → AB

AC → ABCD ⟹ AC derives B valid.

⇒ R (A , B C D E F 4H)

Fd { A → BC
CD → E
E → C
D → AEH
ABH → BD
DH → BC }

check :

BCD → H ?

AB → BC ?

BH → DEH

$(BCD)^+$ → BCDEAH Yes

$(AD)^+$ → ADBCEH. Yes

$(BH)^+$ → BH NO

Identification of addn f-d :

⇒ R(A B C D)

Fd: { A → B
BC → D }

Does AC derives D AC → D

A → AB

AC → ABCD ⟹ AC derives B valid.

⇒ R (A ,B C D E F 4H)

Fd { A → BC
CD → E
E → C
D → AEH
ABH → BD
DH → BC }

check:

BCD → H ?

AB → BC ?

BH → DEH

(BCD)$^+$ → BCDEAH Yes

(AD)$^+$ → ADBCEH. Yes

(BH)$^+$ → BH NO

R (A, B, C)

$A^+ = ABC$

$B^+ = BCA$

$C^+ = CAB$

→ All are candidate keys

R (A, B, C)

↓

check all attributes all one by one

C.K = {A, B, C}

P.K = {A}

A.K = {B, C}

S.K = {A, B, C, AB, BC, AC, ABC}

R (A B C D)

{ A B → C D

C → A

D → B }

$A^+ \rightarrow A$

$B^+ \rightarrow B$

$C^+ \rightarrow CA$

$D^+ \rightarrow DB$

$AB^+ \rightarrow ABCD$

$AC^+ \rightarrow ACB$

$AD \rightarrow ADBC$

$(BC)^+ \rightarrow BCAD$

$(BD)^+ \rightarrow BD$

$(CD) \rightarrow CDAB$

Candidate Ky = {AB, AD, BC, [illegible], CD}

P.K = {AB}

A.K = {AD, BC, CD}

S.K = {ABC, ADC, BCD, CAD...}

Q1 R(A1, A2 . . . . An)

F.D = ϕ |SK^S| = ?

~~No Super Key~~ 1 super Key

All attribute form

Q3: R(A1, A2 . . . An)

① Key = A1 Total = 2^{n-1}

|S.K^S| = ?

② Keys = A1 A2

so SKey = $2^{n-1} + 2^{n-1} - 2^{n-2}$ (no of Key Attributs)

= $2^n - 2^{n-2}$

③ Key A1A2 , A2A3 SKey = $2^{n-2} + 2^{n-2} - 2^{n-3}$ (no of Key AU (A1 A2 A3))

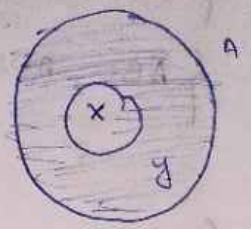
|S.K^S| = ?

Keys = {A1A2, A2A3, A3A5}

$2^{n-2} + 2^{n-2} \rightarrow 2^{n-1}$

FUNCTIONAL DEPENDENCIES

A is a relation A = XUY



X is a key attribute known as determinant

Y is called dependant

It specifies the relationship among attributes and if is treated for input for normalisation process (bigtable → manysmall table)

Roll No.	Name	Age
101	Ram	20
102	Ram	21
103	Sam	20

Roll → Name ✓ [Primary key]

Roll → Age ✓

Name → Roll ✗

Roll, Name → Age ✓ [Super key]

A	B	C
10	b_1	c_1
10	b_2	c_2
11	b_4	c_1
12	b_3	c_4
13	b_1	c_4
14	b_3	c_4

A → B

B → C

C → B

B → A

C → A

C → x c_4 & b_1

Attribute specific:

~~$b_1 \rightarrow c_1$~~ when b_1 repeats the attribute if point to is $b_1 \rightarrow c_1$ so it is functionally dependent correctly

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3

A → B ✓

B → C ✗

AC → B ✓

AB → C ✗

Trivial F.D: A functional dependency is said to be trial, if the right handside is present also in the left handside of the functionality dependency relation.

$X \longrightarrow Y$

$ABC \longrightarrow BC$

$AC \longrightarrow A$

$BDC \longrightarrow DC$

Non-Trival FD: A left handside and right hand side of a functional dependency are indepedent in such case the functional dependancy is said to be Non-trival

examples $A \longrightarrow B$

eg
$AC \longrightarrow BP$
$A \longrightarrow C$
$A \longrightarrow B$
$B \longrightarrow AB$

$AB \longrightarrow CD$
$AB \longrightarrow BC$
$ABC \longrightarrow ABCD$
$ABCD \longrightarrow ABC$
$ABC \longrightarrow BP$

FD1:
$A \longrightarrow B$
$B \longrightarrow C$
$D \longrightarrow EF$

Each function of functional dependencies to generate the perfect table as output

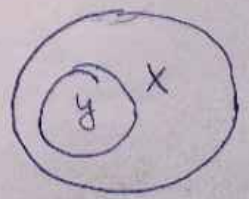
FD1 + FD2 $\xrightarrow{I/O}$ Normalization $\longrightarrow$ Perfect table as o/p.

INFERENC RULES

Reflexive Rules

$X \{Name, Gender\} \longrightarrow \{Name\} Y$

Trival is always reflexive



If $\{y_1, y_2, y_3 \ldots y_m\} W$

$\{x_1, x_2, x_3 \ldots x_n\}$

Augumentation Rule

X {Name, Gender} ⟶ {Name} Y

Trivial is always reflexive. If an attribute is added to right hand side of the FD the augumentation is used

If $\{x_1, x_2, x_3 \dots x_m\} \longrightarrow \{y_1, y_2 \dots y_n\}$

Transitive Rules

{Numbers} ⟶ {Name}

{Name} ⟶ {Gender}

Additive / Union Rule

{Number} ⟶ {Age} {number} ⟶ {name, Age}

Decomposition Rule

{Number} ⟶ {Name, Age}

{Number} ⟶ {Name}

{Number} ⟶ {Age}

FD1

(1) A ⟶ B

(2) B ⟶ C

(3) D ⟶ E, F

(4) F ⟶ G

FD2

(1) (1) and (2) are transitive A ⟶ C

(3) D ⟶ E (6)

(4) D ⟶ F (7)

(5) D ⟶ F (7) From (3) and (7)

(6) D ⟶ G

R (A B C D E)

FD = { AB → C
C → D }

Keys = $(ABE)^+ \longrightarrow ABECD$

CK = ABE
PK = A

R (A B C D E)

FD = { A → B
BC → D
D → BC }

$AE^+ \longrightarrow ABE$ X

$(AEB)^+ \longrightarrow AEB$ X

$(AEC)^+ \longrightarrow AECBD$

$(AED)^+ \longrightarrow AEDBC$

CK = AEC
AED

PK = AEC

A.K = AED

S.K = AECB , AECD .....

Irreducible Set

FDs = { A → B
C → B
D → A
D → B
D → C
AC → D }

Some depend^n may be redundant/invalid

↓

we remove them.

Remove duplicate/invalid/redundant FDs.

R(A B C D)

FD: {A → B, C → B, D → ABC, AC → D}

step 1: Write in simplest form (RHS of attributes in simplest form)

A → B

C → B

D → ABC ⇒ D → A, D → B, D → C] → Decomposition rule

AC → D

Step 2: Remove redundant attribute on LHS

C → BC [Before A removed] → A is not redundant

AC → CBDA [After removing A]

For A check is redu

A⁺ → AB [Before removing C]

AC⁺ → ABDC [After removing C] → C is not redundant

Step 3: Check which FD is redundant

① A → B
② C → B
③ D → A
④ D → B
⑤ D → C
⑥ AC → D

Remove 1 and check if A → B using ② to ⑥ [Rest FD) and similarly for each step. by finding A⁺

A → B (Not redund)

A⁺ → A

Closure for C → B (fro 1, 3, 4, 5, 6)

C → C (Not redund

for $D \rightarrow A$

$D^+ \longrightarrow DBC$ [Not redundant]

for $D \rightarrow B$

$D \rightarrow ABCD \longrightarrow (D \rightarrow B)$ Redundant

Remove (4)

for $D \rightarrow C$ (1, 2, 3, 6)

$D \rightarrow BAB$ [Not redudant]

Step 4 : Apply Union Rule

$A \rightarrow B$

$C \rightarrow B$

$D \rightarrow A$

$D \rightarrow C$] $\longrightarrow (D \rightarrow AC)$ Union Rule

$AC \rightarrow D$

Final set :

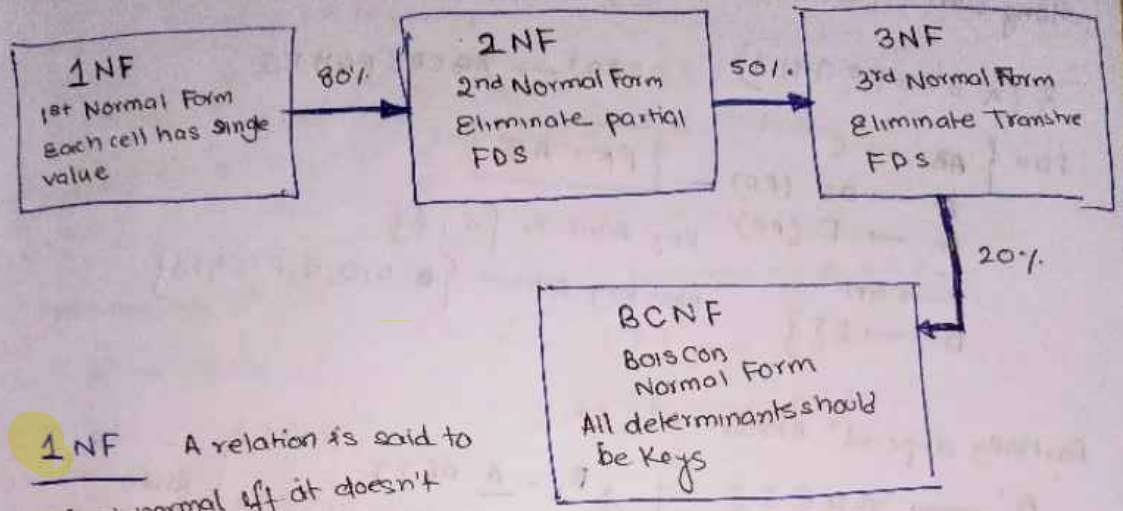
$A \rightarrow B$

$C \rightarrow B$

$D \rightarrow AC$

$AC \rightarrow D$

Normalization It is a process of reducing redundancy from universal table and hence insertion / deletion / modification problems can be elemination



1NF A relation is said to be first normal iff it doesn't contain any repeating groups

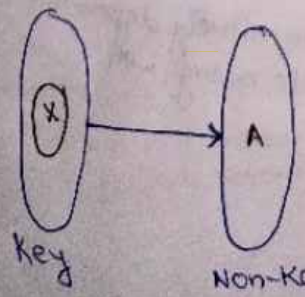
R1

Roll	Name	Course
101	Ravi	RDBMS
		OS
		CN
102	xyz	RDBS
		OS
		CN

After 1NF

Roll	Name	Cour
101	Ravi	RDB
101	Ravi	OS
101	Ravi	CN
102	xyz	RDBMS
102	xyz	OS
102	xyz	CN

2NF: A relation or table is said to be in 2NF if it already in 1NF and should be free from partial funal dependencies



X ⊂ Key → X is part of key

Part of the Key determines non-key attributes (Partial functional dependencies)

P.K = ABC R (A B C D E)

A → DE (P.f.d) AB → B (Not pfd

If there is a partial funcn depnd., then remove partially depend attributes in original table and place them in a separate table along with copy of their determinants

R (A B C D E F G H I J)

$AB^+ \longrightarrow$ ABCDEFGHIJ

FDs { AB → C
A → DE (PD)
B → F (PF)
F → GH
D → IJ }

PK = AB

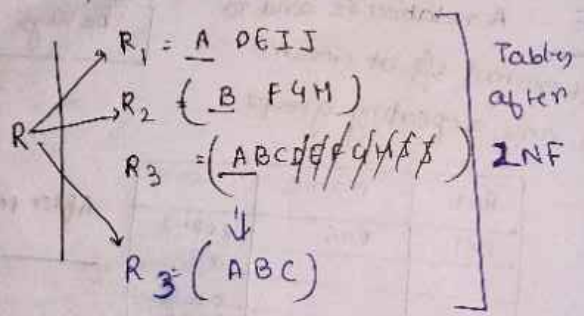
Key Attributes {A, B}

Non Key Attributes {C, D, E, F, G, H, I, J}

Partially dependent Attributes A^+, B^+

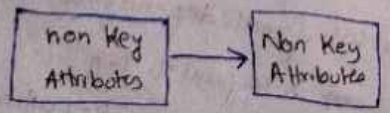
$A^+ \longrightarrow$ A D E I J

$B^+ \longrightarrow$ B F G H



3NF A Relation or Table, is said to be in 3rd Normal Form if it is already in 2NF and free from transitive functional dependencies

Transitive FD:



Remove transitively dependent attributes. in a s

If there is transitive dependency, remove transitively dependent attributes from 2NF and place in a separate table along with copy of its determinants

R (A B C D E F G H I J)

FD: { AB → C
A → DE (P.f.d)
B → F (P.f.d)
(t.f.d) F → GH
(t.f.d) D → IJ }

$(AB)^+$ → ABCDEFGHIJ

Key attrb. {A, B}

Non Key attr: {C D E F G H I J}

Decomposing to 2NF

A^+ = ADEIJ

B^+ = BFGH

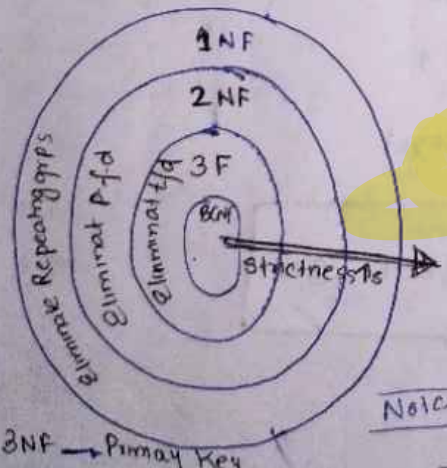
R → R_3 = ABC
R → R_1 = ADEIJ
R → R_2 = BFGH

Decomposing to 3NF

$(F)^+$ → FGH

$(D)^+$ → DIJ

R → R_1 = ADE
R_2 = BFGH
R_3 = ABC
R_4 = FGH
R_5 = DIJ



3NF → Primary Key

BCNF → Focus on Candidate keys (more strictness)

BCNF: Table should be in 3NF
All determinants should be key
Boyce-Codd Normal form.
A relation of table is said to be in BCNF if it is already in 3NF and all determinants are Keys.

Note: 3NF cannot handel the following situations effectively

(a) If table consists of multiple composite candidate keys eg: AB, CD, BC etc

(b) If there is overlapping attributes among Candidate Keys

R (A B C D E F G)

FDs { determinants
BCD → A
BC → E
A → F (t.d)
F → G (t.d)
C → D (P.f.d)
(t.d) A → G }

$(BCD)^+$ → BCDAEFG (S.K)

$(BC)^+$ = BCDAEFG (S.K)

A^+ — AFG

F^+ → FG

C^+ → CD

Check which determinants are Keys

Identify Primary Key

Convert to 2 NF

Convert to 3 NF

Primary Key : BC

2NF

R → R_1 = (B C D E F A)

R → R_2 = CD

~~R_3 = AFG~~

3 NF

R_1 = BCEA

R_2 = CD

R_3 = AF

R_4 = FG

R → R_1 = BCEA

R → R_2 CD

R → R_3 AF

R → R_4 FG

BCNF (All determinants should be keys)

Determinants — Key

BCD (S.K) — BC

BC — C

A — A

F — F

C — C

All determinants are key

Here BCNF = 3NF

BCNF

R = A B C D E F G H

FD's : { A B → C E F G H
A → D (P.f.d)
F → G (TA)
F B → H
H B C → A D E F G (S.K)
F B C → A D E } (S.K)

AB^+ → A B C E F G H D

A B C → A B C A D E F G H

F B C → A D E F B C G H D

B'

2 NF

R_1 = A B C D E F G H

R_2 = A D

3 NF

R_1 = A B C E F G H

R_2 = A D

R_3 = F G

BCNF

Determinants	Keys
AB	AB
A	A
F	F
FB	
HBC (S.K)	
FBC (K.N)	

$(FB)^+$ → F B H G

R_1 = A B C F H

R_2 = A D

R_3 = F G

R_4 = F B H

Properties of Decomposition

- Converting Universal table into
- Smaller subtables

(1) During decomposition of table we are eliminating insertion deletion and modification problems but beyond certain level in addn to reduce such problem, we get few problems decribed by

(1) Loss less join Property
(2) Dependency Preserving property

→ Both condn satisfy → Good Decomposition

Lossless ~~Decon~~ join. A decomposition is said to be lossless if natural join of all fragments is equal to original table

$$R = \Pi_{R_1}(R) \bowtie \Pi_{R_2}(R) \bowtie \cdots \Pi_{R_n}(R)$$

R → R_1, R_2, R_3

1000 ↓ no. of records

100 ↓ records (Rows)

Combining more than 1 table (Loss less)

Lossy Join: Natural join of decomposed fragments is more and is superset of original table

$$R \subset \Pi_{R_1}(R) \bowtie \Pi_{R_2}(R) \bowtie \cdots \bowtie \Pi_{R_n}(R)$$

R_1 AB

eg:

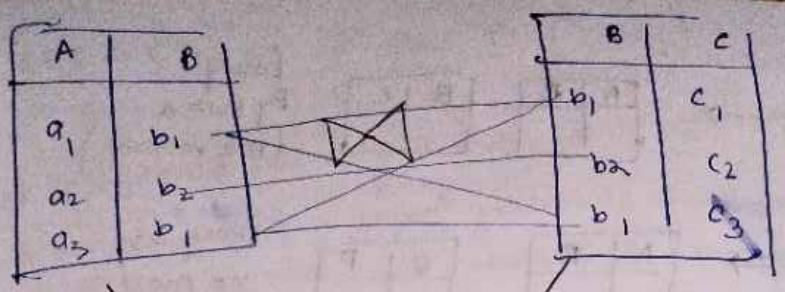
R	A	B	C
	a_1	b_1	c_1
	a_2	b_2	c_2
	a_3	b_1	c_3

R(A,B,C)

R_2 BC

(a)

(b)



Join Using Common Attributes

$\Pi_{R_1}(R) \bowtie \Pi_{R_2}(R)$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_3
a_2	b_2	c_2
a_3	b_1	c_1
a_3	b_1	c_3

→ No of Rows ↑s after natural join

Procedure to check Lossless join Decomposition.

(a) Check whether all attributes of original relation are present in decomposition or not If not the decomposition is lossy decomposition

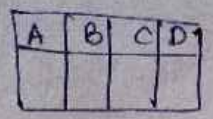
* Between 2 not all

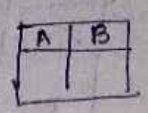
(b) $R \rightarrow R_1, R_2$

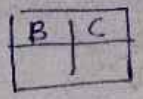
$R_1 \cap R_2 \neq \phi$ → It is lossless

$(R_1 \cap R_2)^+ \rightarrow R_1$ or $(R_1 \cap R_2)^+ \rightarrow R_2$ → It is Lossless

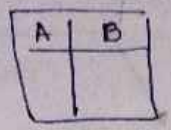
All cond^n satisfy → Lossless

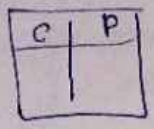






Lossy
By Rule a
(D is missing)

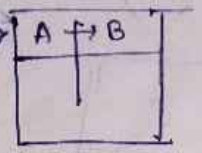


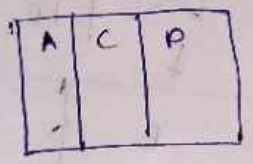


Lossy as
no common
Attribute

R (ABCD)

FD { A → B
B → C
C → D }





$(AB \cap ACD)^+ = (A)^+ = ABCD$

$R_1 \cap R_2 \longrightarrow R_1$ (Lossless)

So Lossless

eg: R (A B C D)

FD = { A → B , B → C , C → D }

D = { AB, AD, CD }
R_1 R_2 R_3

(1) All attribute present?

$AB \cap AD = A$

$(A^+) \longrightarrow (A)(B)C(D) \longrightarrow$ Determine AB and AD → Resultant join → ABD

ABCD

(ABD ∩ CD

(b) Dependen

A decomp

all origin

directly

eg R

FD:

D :

AB

A^+ —

B →

C

Indirect

let α —

D = { R

Result =

for

Result :

ste

$(ABD \cap CD)^+ = D^+ = D$ (lossy)

(b) Dependency Preserving Property

A decomposition is depen preserving if all fundn depen of all original relation are preserved by the decomposition directly or indirectly

eg R(A B C D)

FD: {A → B, B → C, C → D, D → A}

D: {AB, BC, CD}

Yes Dependony Preserving

AB

$A^+ \longrightarrow AB$ (Fun depn is preser)

$B \longrightarrow C \Longrightarrow$ Preserved by BC

$C \longrightarrow D \Rightarrow$ Preserved by CD

→ Preserved directly

Indiret Preservation

let $\alpha \longrightarrow \beta$

$D = \{R_1, R_2 \ldots R_K\}$

Result = α while (chase to the result existe)

for i = 1 to K {

result = result $\cup$ $[(result \cap R_i)^+ \cap R_i]$ }

if the result contain β, then $\alpha \longrightarrow \beta$ is preserved.

result = D

#step1

result : $D \cup [(D \cap CD)^+ \cap CD]$

$D \cup (DABC \cap CD) - CD$

→ doesn't cont A

Result- CD

$$\text{Result} = CD \cup \left[(CD \cap BC)^{+} \cap BC\right] = BCD$$

Result- BCD

$$= BCD \left[(BCD \cap AB)^{+} \cap AB\right]$$

$$= BCD \left[\{B\}^{+} \cap AB\right]$$

$$= BCD \left[ABCD \cap AB\right]$$

$$= ABCD$$

A is present in this result

Dependency ~~Decomposition~~ is preserved indirectly

Relation

- SQL
  - DDL → Create, Alter, Drop, Truncate
  - DML → Insert, Update, Delete
  - DQL → Select
  - DCL → Revoke, Deny, Grant
  - TCL → Commit, Rollback, Save

Delete memory spaces are not erased

Truncate memory space for records deleted

1) Logical Operators (<, >, <=, >=, AND, OR)

2) Range Operators (Between, Not Between)

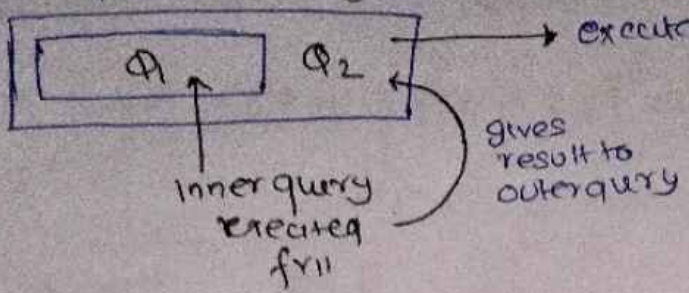
3) Set members : (In, Not In)

4) Pattern Maching Operators (Like, Not Like) → use for pattern mach

* Reg Expression → A% ← Followed by any sequence of charcter
↑ starts with A

A__b% u ← ends with u
↓ any single charcter

Sub Queries Query within another query



write a query to get details of employee who earns highest salary in organisam and who earns second highest query.

SELECT * FROM employe WHERE sal = (SELECT max(sal) form employee)

(2)

2nd highest

select * from emplay whex sal = (select ~~max~~(sal) from exmplay

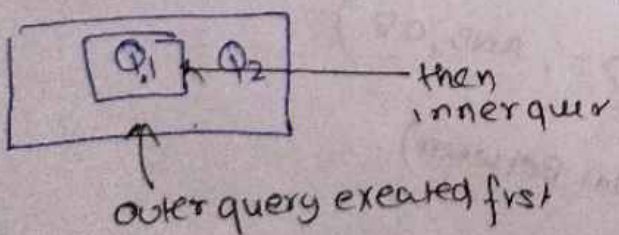
wher sal2 < max(salary) AND sal2 == max(sal)

2nd highest

sect * from emplay wher sal = (select max(sal) fron employee

where sal < (select max(sal) from employee))

Correlated Query:



eg: Select * from employee E
where 0 = (select count(Dist sal)
from emp where emp.sal > E.sal

Relational Algebra: Procedural query Language

(1) Native RA Operators
- Select (σ)
- Closure (Π)
- Division ($\div$)
- Renaming (ρ)
- Assignment ($\leftarrow$)
- Join ($\bowtie$)

Extende RA oper
- Minimum (MIN)
- Maximum (MAX)
- Count (COUNT)
- Sum (SUM

(2) Set Theory Relation Operat
- Union ($\cup$)
- Intersetor ($\cap$)
- Cartesian Product ($\times$)
- Minus ($-$)

Query select * from student where branch = cse and b>15

σ(studen) { branch == CSE, age > 15

Select (σ)

$\sigma(R)_{<condition>}$

no. of attributes = degree.
↑ no change in degree

* Select operators Rows but not colums
* Select only rows
* Commutative in nature

$\sigma_{<C_1>}(\sigma_{<C_2>}(R)) = \sigma_{<C_2>}(\sigma_{<C_1>}(R))$

$\sigma_{<C_1>}(\sigma_{<C_2>}(R)) = \sigma(R)(<C_1> \text{ and } <C_2>)$

Project operators (Project Columns)

Π (R) ⟨attribute list⟩

Q: Select name and age from student table

RA: Π (studen) ⟨name, age⟩

* Degree = no. of attributes in attributes list
* Cardinality remains same
* Filters Colouns not rows
* Not Commutative in nature

Select Rollno, Name, and age from student where branch is CSE and age > 15

Q: ~~Select * from student wher~~

Select Name, age, Rollno from student where branch == CSE and age > 15

~~Π σ (student) ⟨age > 15 and branch == 15⟩~~

Rollno, name, age

σ ($\Pi_{roll, name, age}$ (R)) X

branch == cse and age > 15

$\Pi_{roll, name, age}$ (σ(R) branch == CSE and age ≥ 15) ✓

→ inner query executed first

Select * from student

σ (~~students~~) RA : student

Division $\left(\frac{\div}{\div}\right)$

Suitable for queries forall / for every

$R \div S = \frac{AB}{B} = A$ Resultant relation has only 1 attribut A

R(AB) $S \subseteq R$

S(B)

A	B
a_1	b_1
a_1	b_2
a_3	b_1
a_4	b_2

R

S:

B
b_1
b_2

consider only attribute in A which has bot b_1 and b_2

$R \div S$:

A
a_1

$R \div S = \frac{ABC}{C}$

A	B
a_1	b_1

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_2
a_1	b_2	c_1

S:

C
c_1
c_2

$$\deg(R \div S) = \deg(R) - \deg(S)$$

$$\text{Card} = 0 \leq |R \div| \leq \frac{|R|}{|S|}$$

$$R \div S = \prod_{A}(R) - \prod_{A}\left(\left(\prod_{A}(R) \times S\right) - R\right)$$

$$= \prod_{R-S}(R) - \prod_{R-S}\left(\left(\prod_{R-S}(R) \times S\right) - R\right)$$

A
1
5

Rename (ρ) operator.

$\rho_S(R)$: Rename relation to S OR $S \leftarrow R$

$\rho_{P,Q}(R)$: Rename attributes of R to P and Q

$\deg(R) = \deg(S)$

$\text{Car}(R) = \text{Car}(S)$

SET THEORY OPERATIONS ([redacted], ∪, ∩, −)

For above cases, relation must be union compatible

must have same domain

R

A	B
1	3
5	7

S

S	D
1	3
6	8

R ∪ S

A	B
1	3
5	7
6	8

All records must be there

must have same degree

for Union

$deg(R \cup S) = deg(R) = deg(S)$

$\max(|S|, |R|) \leq |R \cup S| \leq |S| + |R|$

R ∩ S

A	B
1	3

Select the common Record

For Intersection

$deg(R \cap S) = deg|R|$

$0 \leq |R \cap S| \leq \min(|R|, |S|)$

R − S

A	B
5	7

Remove the common record from R

For minus

$deg(R - S) = deg(R) = deg(S)$

$0 \leq |R \cap S| \leq |R|$

Cartesian Product (×)

It is used to combine more than relation (2 relation)

A B C and B C D are different attributes in the given case.

R:

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

S:

B	C	D
a_2	c_2	d_2
a_3	c_3	d_3

R × S :

R			S		
A	B	C	B	C	D
a_1	b_1	c_1	a_2	c_2	d_2
a_1	b_1	c_1	a_3	c_3	d_3
a_2	b_2	c_2	a_2	c_2	d_2
a_2	b_2	c_2	a_3	c_3	d_n

$$\deg(R \times S) = \deg(R) + \deg(S)$$

$$|R \times S| = |R| \times |S|$$

Join Operator ($\bowtie$)

Used to retrive data from moltiple table based on certain condition

- $\bowtie$ Inner Join → (May be Natural join ($\bowtie$) / on < (condⁿ / using < (cond
- $\bowtie$ Left outer Join (unmanched record from left)
- $\bowtie$ Right outer Join (unmatched record from right)
- $\bowtie$ Full outer Join. (unmtched from both side)

Natural Join

$R \bowtie S$: Retriving data from R and S based on common attributes.

R:

A	B	C
a_1	1	c_1
a_2	2	c_2
a_3	3	c_3

S:

A	B	D
a_1	1	e_1
a_3	2	e_3
a_4	3	e_4

→ only if common join

Common in R and S

$R \bowtie S$

A	B	C	D
a_1	1	c_1	e_1

only one comman attributes

* $deg(R \bowtie S) = deg(R) + deg(S) - No\ of\ Common\ attributs$

* $0 \leq |R \bowtie S| \leq |R| * |S|$

• Innen Join / θ-Join

R innerjoin S | $R.B < S.B$ | $R \bowtie_{R.B<S.B} S$

R:

A	B	C
a_1	1	c_1
a_2	2	c_2
a_3	3	c_3

S

A	B	E
a_1	1	e_1
a_3	2	e_3
a_4	3	e_4

$R.B < S.B$

Here A is different for R and A

Filter from Cartesian proddct base on given condn ($R.B < S.B$

$$R \bowtie_{R.B<S.B} S$$

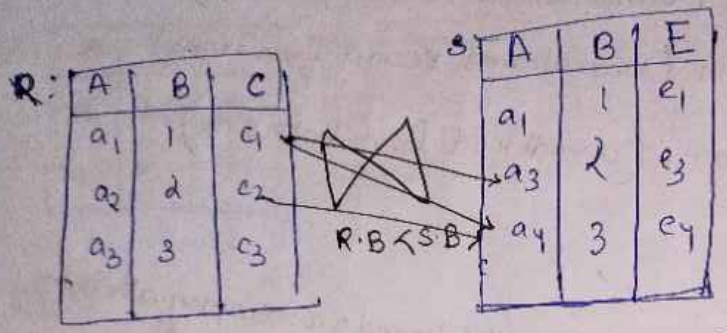
A	B	C	A	B	E
a_1	1	c_1	a_3	2	e_3
a_1	1	c_1	a_4	3	c_4
a_3	3	c_3	a_4	3	e_4

$$\deg(R \bowtie_\theta S) = \deg(R) + \deg(S)$$

$$0 \leq |R \bowtie_\theta S| \leq |R| * |S|$$

• Innen Join / θ-Join

R innerjoin S | $R.B < S.B$ | $R \bowtie_{R.B<S.B} S$



Here A is different for R and A

Filter from Cartesian proddct base on given condn (R.B < S.B

$$R \bowtie_{R.B<S.B} S$$

A	B	C	A	B	E
a_1	1	c_1	a_3	2	e_3
a_1	1	c_1	a_4	3	c_4
a_3	3	c_3	a_4	3	e_4

$$deg(R \bowtie_\theta S) = deg(R) + deg(S)$$

$$0 \leq |R \bowtie_\theta S| \leq |R| * |S|$$

TRC and DRC

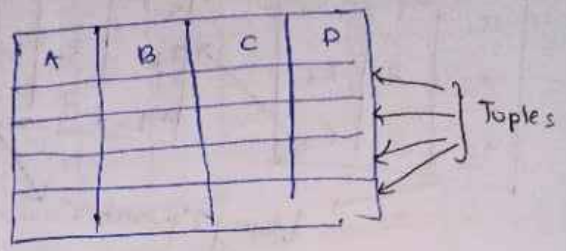
TRC: Tuples Relational Calculus → Nonprocedural query lang.

DRC: Domain Relational Calculus

TRC: $\{t \mid P(t)\}$ set of tuple t true is true for t

t → represents a row

t[B]: Value of tuple t for attribut B



SQL: Select * from instructor where salary > 8000

RA: $\sigma_{sal > 8000}$ (Instructor)

TRC: $\{t \mid t \in Instructor \wedge salary > 8000\}$ → Retrive all columns.

(Logical and)

For selecting columns, we need to use Quantifiers.

Use existential / "There exists" quantifier → $\exists t \in R(P(t))$

There exists a tuple t in a relation R such that predicate P of t is true

Query: Find the instructor id for each instructor with salary grater than 80,000

$$\{t \mid \exists s \in Instructor\ (t[ID] = s[ID]) \wedge s[sal] > 80{,}000\}$$

It represents set of tople t such that there exists a tuple in relation instructor for which the values of t and s for id (instructor id attribute) are equal or same and values of s for salary is grater than 80,000

Find all students who have taken all courses offered in CSE department

For all Quantifier $\forall t \in R\,(P(t))$

Domain Variables: Domain Variables are

Domain Relational Calculus represent domain variables that take values from Domain rather than values from entire tuple

$$\{\langle x_1, x_2, \ldots x_n \rangle \mid P(x_1, x_2 \ldots x_n)\}$$

Find instructor details for instructor where Salary > 8000

Instructor (ID, name, dep-name, sal)

$$\{\langle i, n, d, s \rangle \mid \langle i, n, d, s \rangle \in \text{Instructor} \wedge s > 8000\}$$

Find all instructor name where salary graten than 80,000

$$\{\langle n \rangle \mid \exists\, i, d, s\,(\langle i, n, d, s \rangle \in \text{Instructor} \wedge s > 8000)\}$$

Find details of employees coming from hydrabad.

SQL: Select * from employee where home = hydrabad

RA: $\sigma_{\text{home = hydrabad}}(\text{employe})$

TRC: $\{t \mid t \in \text{Employee} \wedge t[\text{city}] = \text{hydrabad}\}$

DRC $\{\langle \text{eno}, \text{name}, \text{age}, \text{sal} \rangle_{\text{city}} \mid \text{Emp}(\text{empno}, \text{name}, \text{age}, \text{sal})_{\text{city}} \wedge \text{city} = \text{hydrabad}\}$

File Organisation.

Records are stored in tables which are stored in Blocks of memory.

(1) Unsorted File Organisation: based on non-ordering key-field

150	X	CS
100	Y	IT
121	Z	CE

Not in sorted order

sid	name	branch
150	X	CS
100	Y	IT
121	Z	CE

Database table

Searching/update/deletion time complexity is $O(n)$ in worst case

Insertion occurs in $O(1)$

(2) Sorted File Organisation: Ordering key field.

150	X	CS
100	Y	IT
121	Z	CE

sid	name	branch
100	Y	IT
121	Z	CE
150	X	CS

Searching / Update / deletion time complexity $O(\log_2 n)$

Insertion is at $O(n)$

Indexing

We know that data is stored in the form of records. Every record has a key feild which helps it to be recognised uniquely.

Indexing is a data structure to retrive records from efficiently from database files based on some attributes on which indexing has been done.

- Indexing
  - Primary Index
    - Both data and index pages are sorted.
    - ordering Key feild based
  - Cluster Index
    - Non ordering Key feild based
    - Both data and index pages are sorted
    - Perform on column having group of value
  - Secondary Index
    - Based on non ordering non Key feild
    - Index pages are sorted but not data pages are sorted.

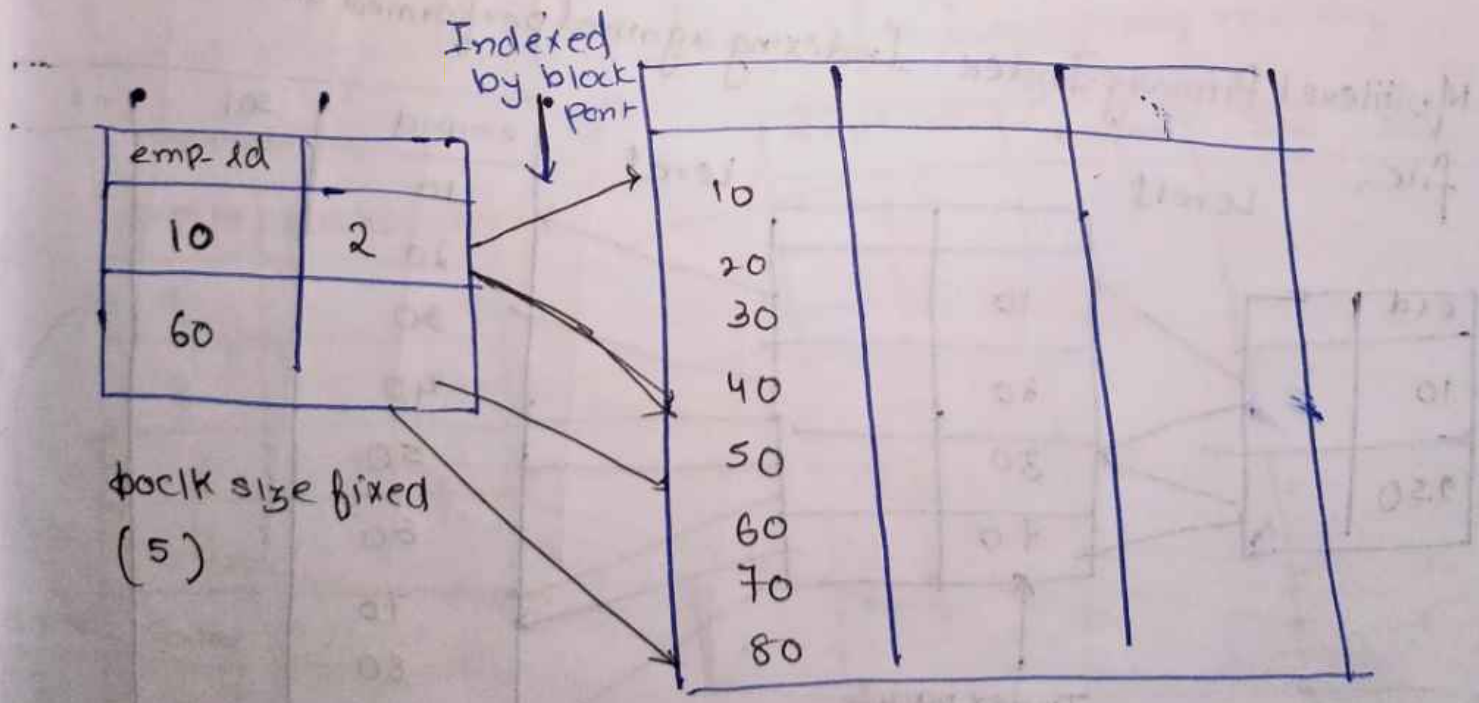
Primary Index: It is created based on ordering Key feild (Primary Key of the table) therefore only one primary index per table is allowed.

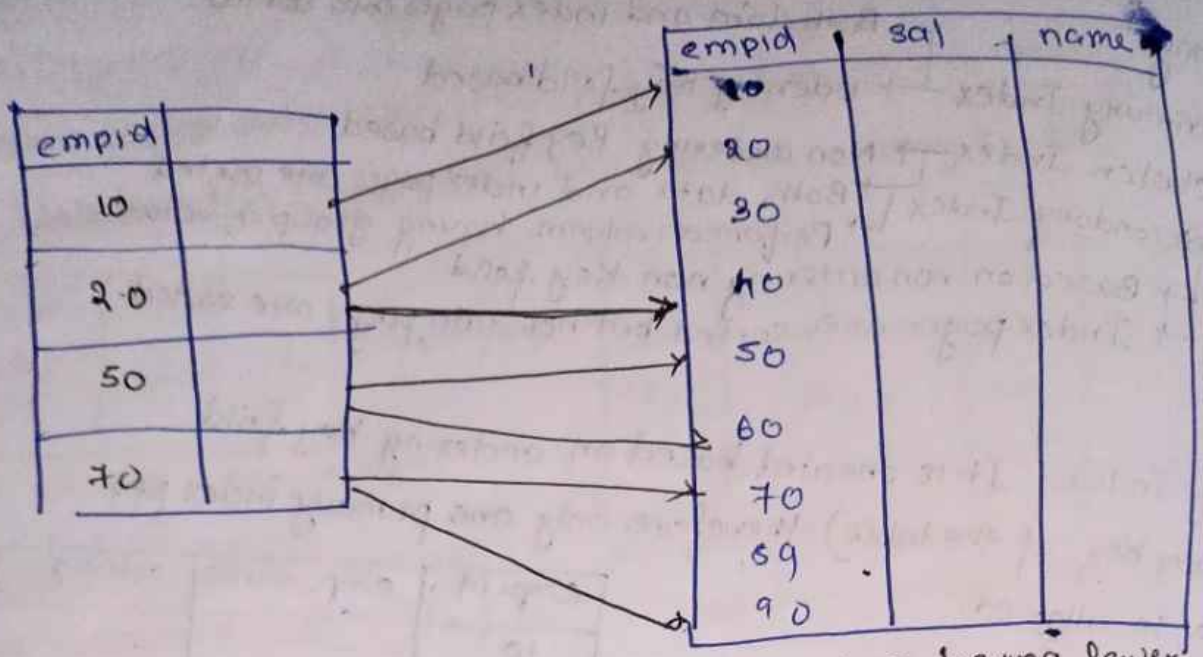
Ordering Key Field	Block Pointer

emp_id	emp_name	salary
10	—	—
20	—	—
30	—	—
40	—	—

Ordering Key → Sorted

Blocking factor is no of record per block

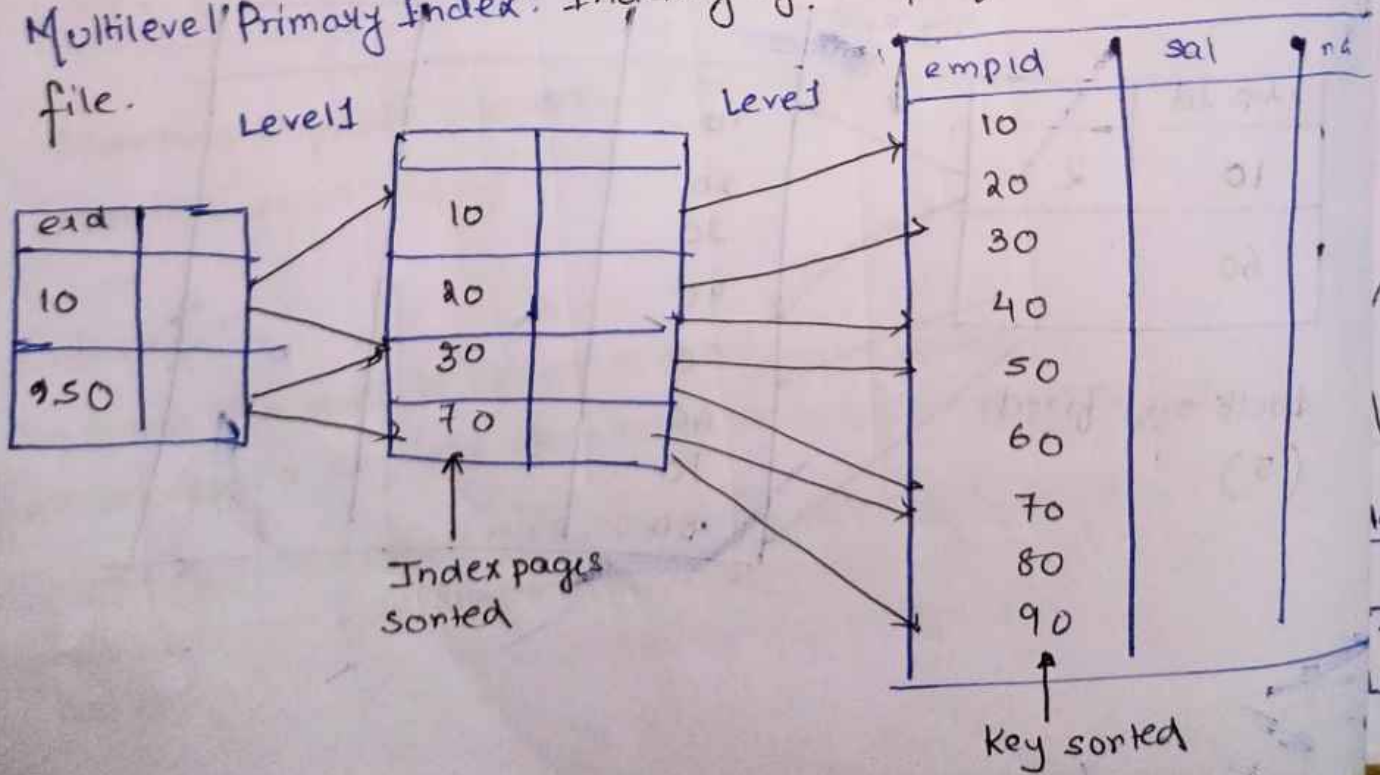




Sparse Index : Index that contains entries having fewer than orginal record

Dense Index : Index that contain entries for each record of original file

Multilevel Primary Index : Indexing agained performed on indexed file.

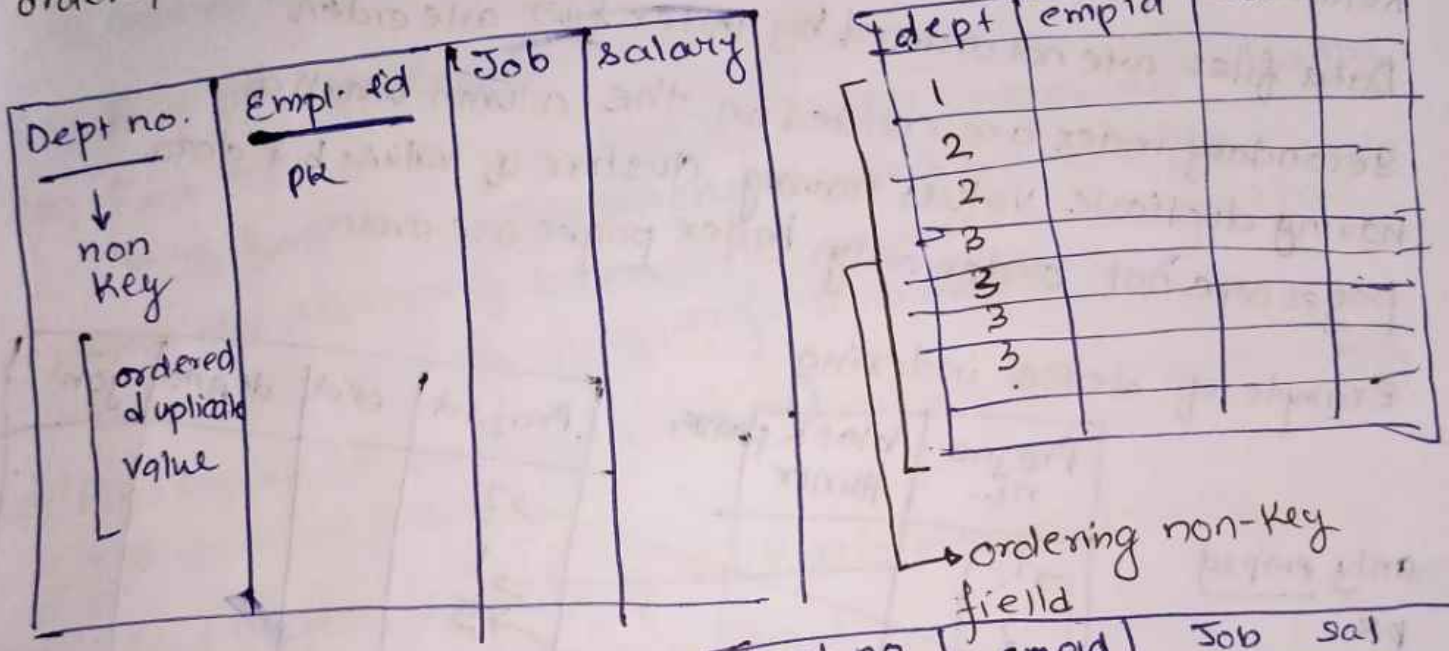


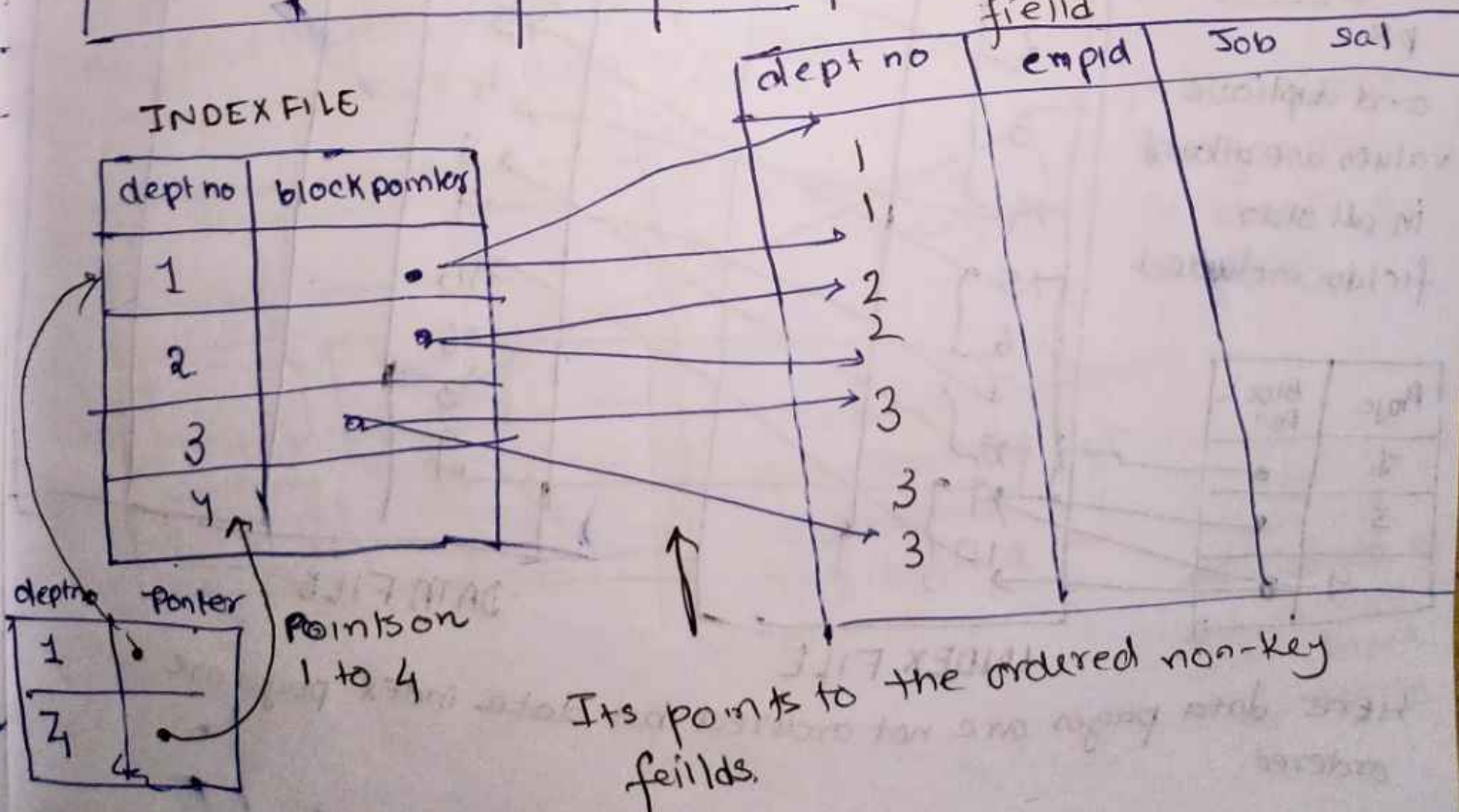
Clustered Index

Based on ordering non-key field.

It is created based on ordering non-key field. that is clusetered index is created on a column which of having a group of values or clustered values.

Both data and index pages are ordered. Since data pages are orders, we have only one clustered index per table.





The difference is that it is not mapped on primary index / key values like in primary indexing instead mapped on

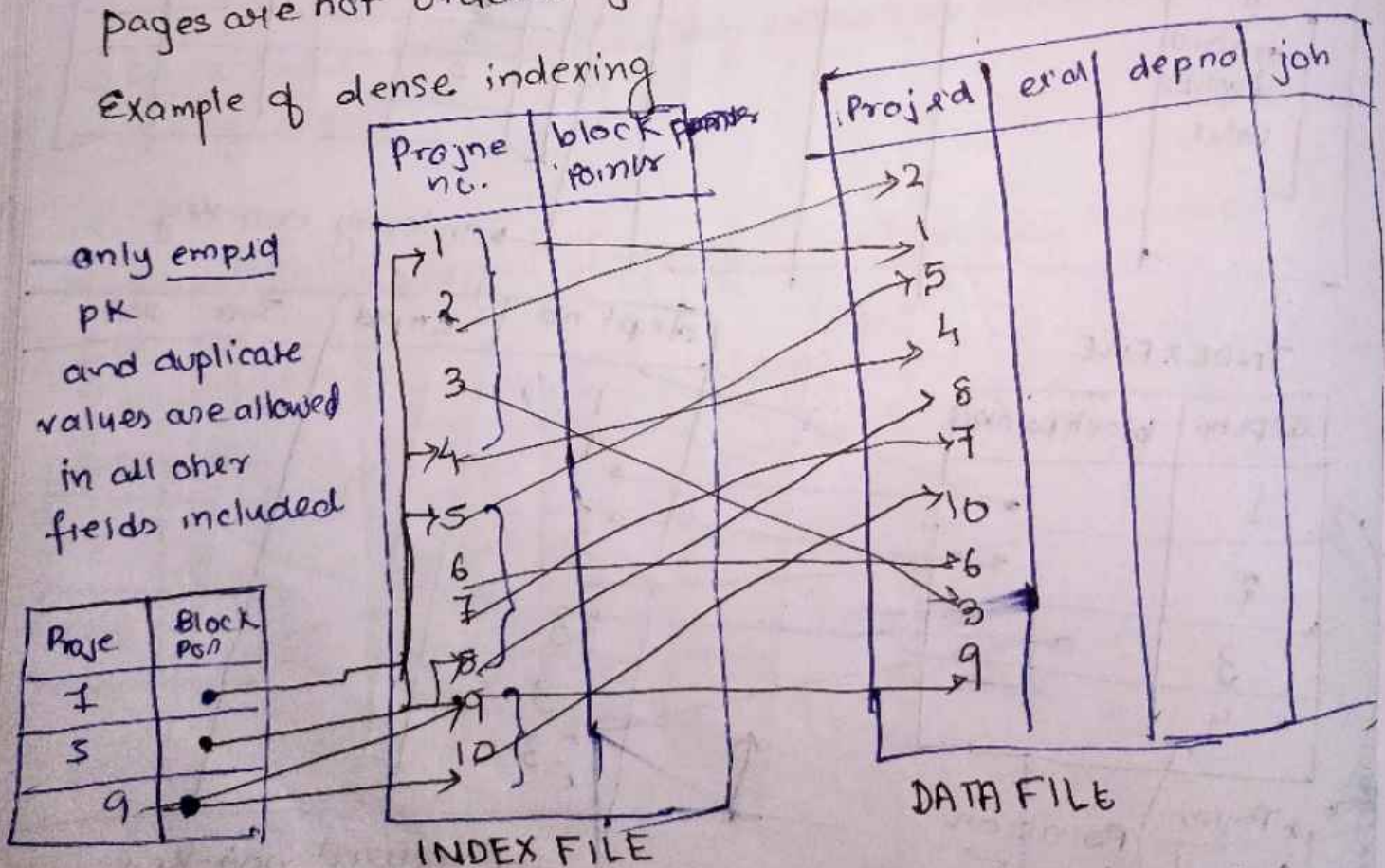
Secondary Indexing → created on non ordering no key field

Related to ~~sparse~~ dense indexed.

Data files are not ordered by index files are orders.

Secondary index are created on the column which of having duplicate values having clustre of valus. but data pages are not order only index pages are order

Example of dense indexing



only empid PK and duplicate values are allowed in all other fields included

Proje	Block Poi
1	
5	
9	

Here data pages are not ordered but ~~data~~ index pages are ordered.

Assuming 4 index records per blocks.

Que Consider a disc block of size 512 bytes. A block pointer is 6 bytes and record pointer is 7 bytes long. A file has 30,000 employee records of fixed length. with each record taking 100 bytes, assuming that an unspaned file organisation is used to store and SSN text 9 bytes. suppose the file is ordered by the key feild SSN.

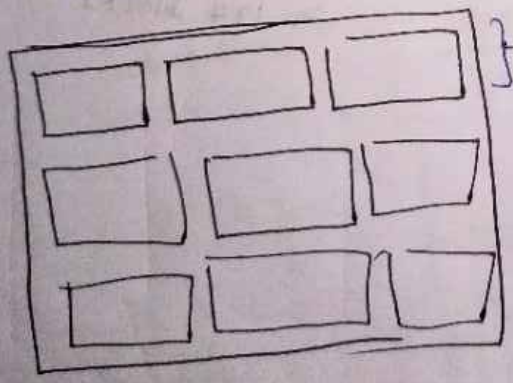
(a) Find the blocking factor index file

(b) Find the number of first level index entries and also the no. of first level index blocks required for index files

(c) Find the number of block axis required to retrive the record from the file using single level index.

(d) Find the no levels required to transform the primary index into multilevel index.

(e) Find the total number of block axis required using multilevel index to find any record.



Blocks of memory

2ndary memory is organised in blocks of memory

Unspanned Data Base File not accomodated with block of memory

Spanned DB File is accomodated within block of memory

Block Size = 512 bytes
Record Size = 100 bytes (of each row)

datafile

(i) Blocking factor of employee = $\lfloor \frac{512}{100} \rfloor$ = 5 employee record per block of memory

(ii) No. of blocks required for total employee = $\frac{30,000}{5}$ = 6000

(a) $\left\lfloor \frac{512}{(9+6)} \right\rfloor = \left\lfloor \frac{512}{15} \right\rfloor = \lfloor 34.13 \rfloor = 34$

SSN	block pon
9 byte	6 byte

blocking ~~factor~~ index records = 34 per block of memory

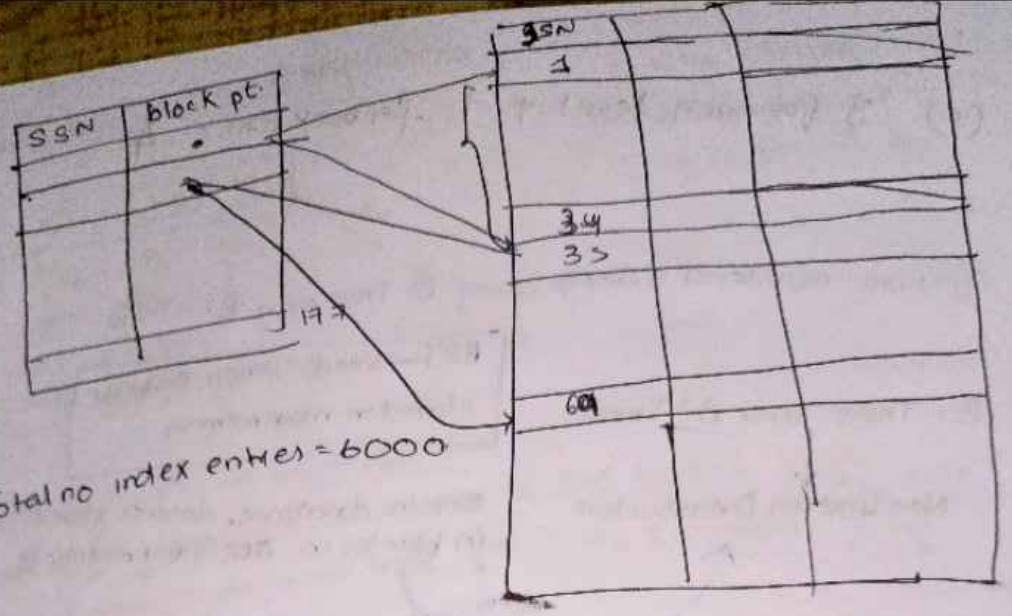
(b)

SSN	
1	
6000	

No. of index blocks are at first level are $\lfloor \frac{6000}{34} \rfloor$

= 177 blocks

(c)



Total no index entries = 6000

(c) $\log_2(177) = 7$ → search axis on binary block

+1 → 1 for search employee block.

(d) level 1 = $\left\lfloor \frac{6000}{34} \right\rfloor = 177$

level 2 = $\left\lfloor \frac{177}{34} \right\rfloor = 6$

level 3 = $\left\lceil \frac{6}{34} \right\rceil = 1$

1 + 6 + 177 = 184 block

max levels are 3

(e) 3 for each level + 1 for box axis = 4

Dynamic multilevel indexes using B- Tree and B+ - Tree

B- Tree and B+ Tree

↓

Non Linear Data Structure

BST → used when Data is stored in main memory

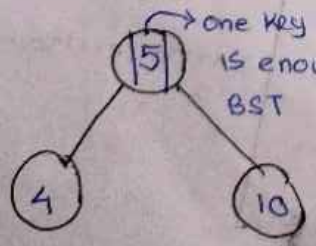
But in database, data is stored in blocks in Auxillary memory

Disadvantages of indexed sequential File organisation

Performance degrades as the file grows, both for index lookup and sequential scan for the data. (Time consuming)

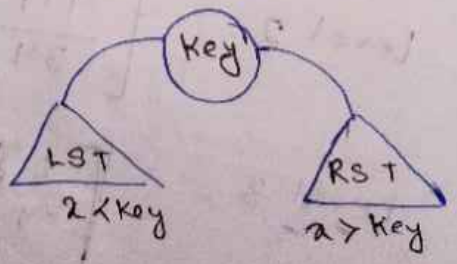
B-Tree

Multiway binary search tree whose objective is to minimise height of tree as much as possible to enable faster / rapid access of data items.

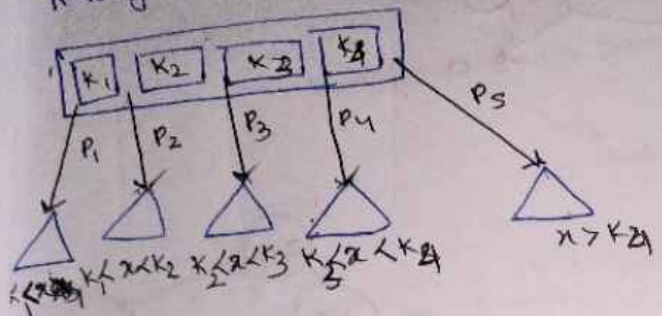


one key value is enough in BST

In B-Tree, more than one key are require



K-way BST → K no. of key values



K-way subtree
no. of key values = K-1
no. of children = K

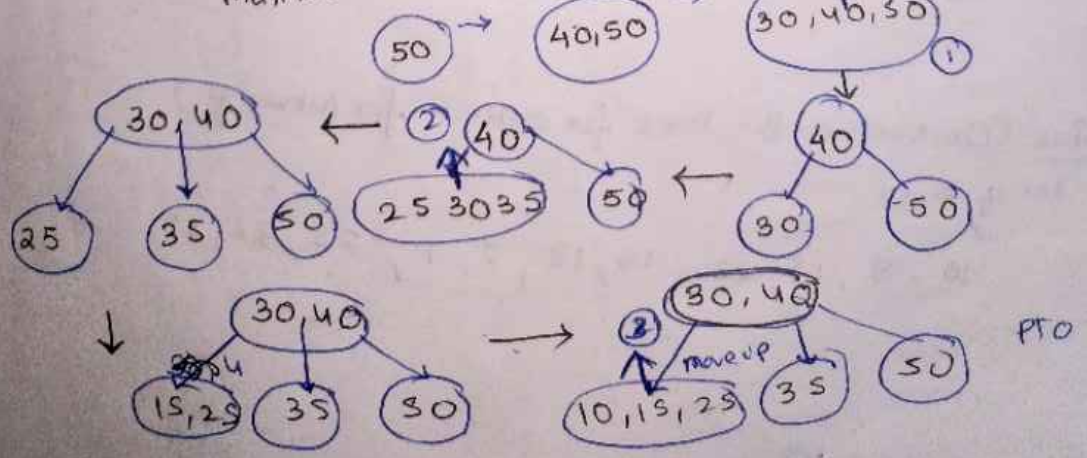
Properties of B-Tree of degree / order K

1. The root must have 2 to K children ie 2-1 to K-1 keys
2. Every internal node must have $\lceil K/2 \rceil$ to K children
   ie $\lceil \frac{K}{2} \rceil - 1$ to K-1 key
3. Every leaf node must have $\lceil \frac{K}{2} \rceil$ to K null links
4. All leaf nodes should be at same level

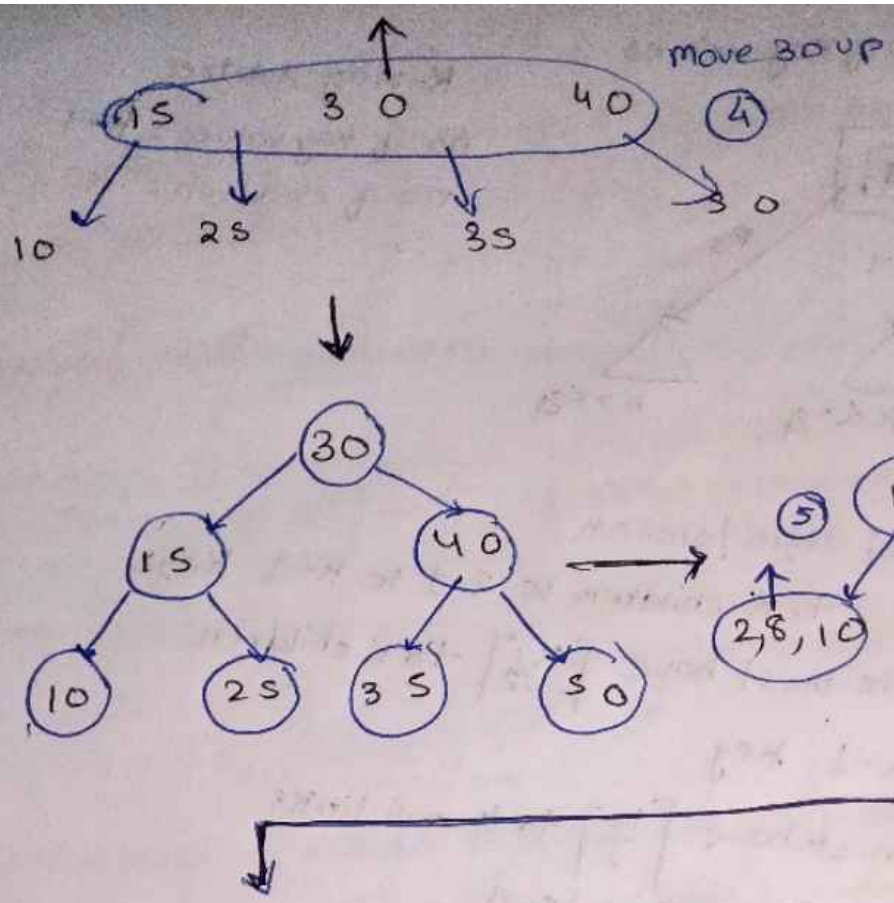
Que Construct a B-Tree of order 3 for the following set of each
50, 40, 30, 25, 35, 15, 10, 17, 8, 2

at root minimum : 2
maximum : 3

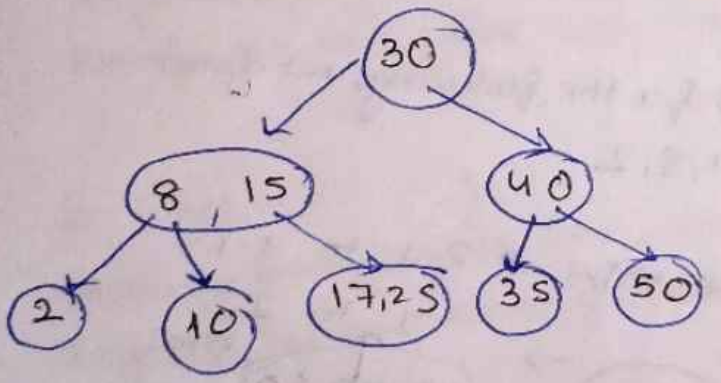
key value = 2-1 to 3-1
1 to 2 more



PTO



Total 5 splits



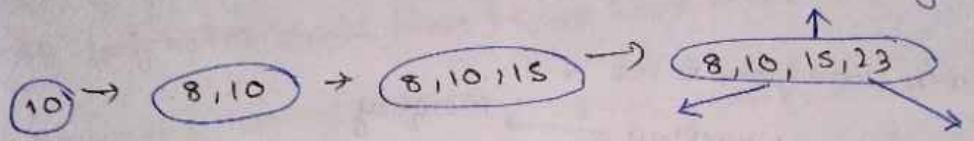
Que Construct a B-Tree for order 4 for following set of keys

10, 8, 15, 23, 16, 18, 2, 1, 25, 32, 21

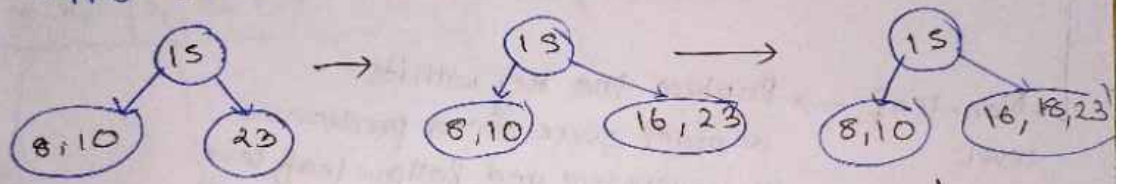
In case order is add, move | split middle element but incase of even element, fallow left basis | right basice

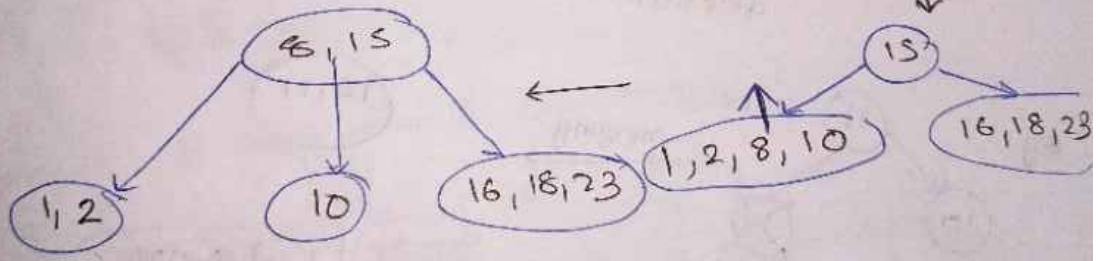
Insertion → Overflow → odd order → move the middle element up

order even → Left basis
Right basis

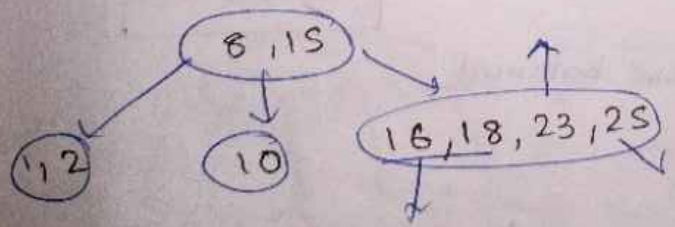


Apply left bas.





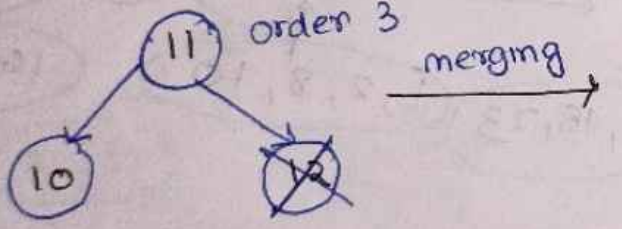
8,15,23



Deletion of Node from B tree →

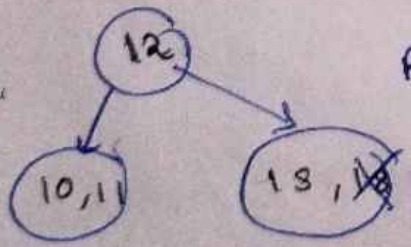
- Leaf Node → Underflow → Merging
  - Redistribution
- Non-leaf level → Replace the key with its in order successor or predecessor or predecessor and follow leaf level deletion

eg



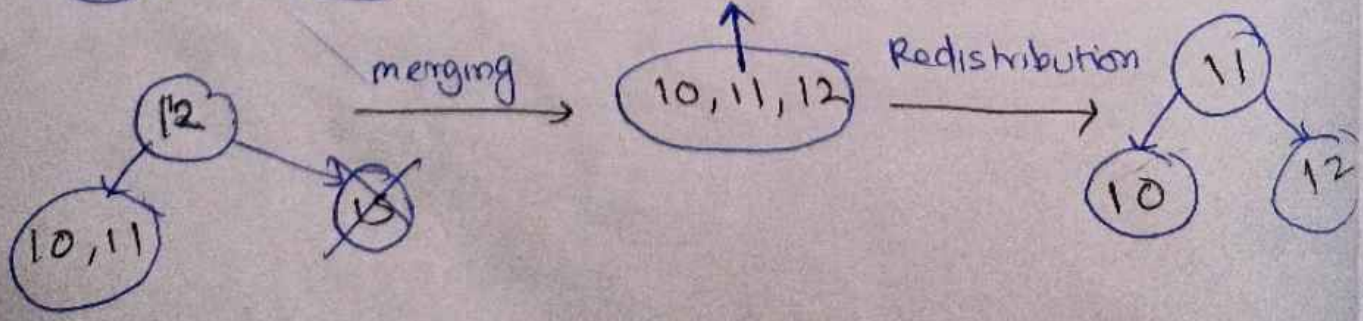
Imp* Question related to nsition

eg:



Remains balanced

eg



Properties of B+ tree order K

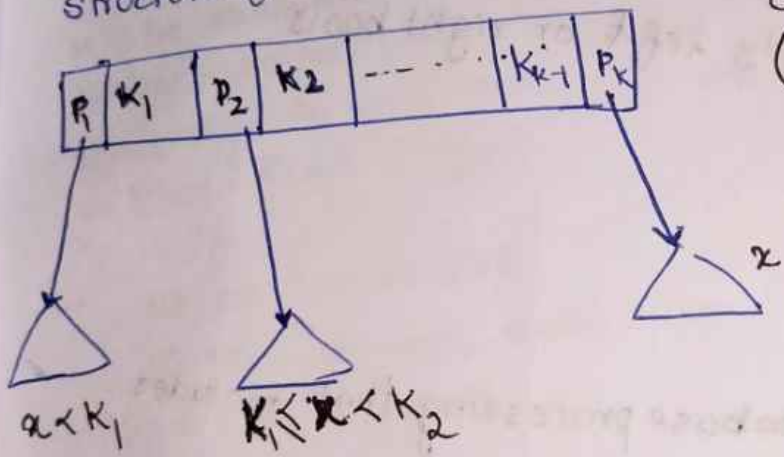
Root must have 1 to K children

Every internal node must have $\lceil \frac{K}{2} \rceil$ to K children

Every leaf node must have $\lceil \frac{K}{2} \rceil - 1$ to K-1 key / record pointer.

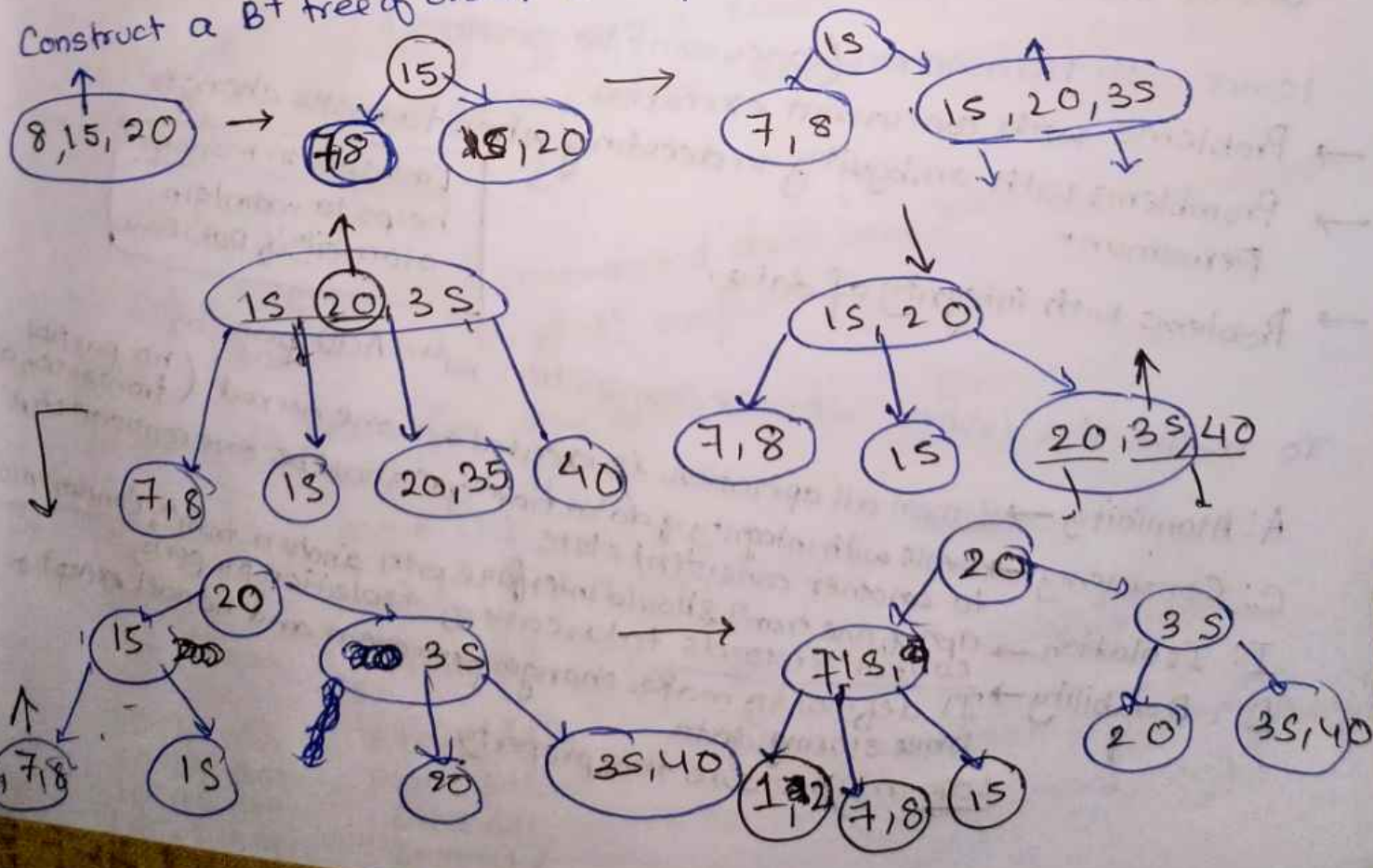
All leaf nodes should must be at same level.

Structure of internal node.



B tree varies from B+ tree on basis of internal structe (≥ 2)

Construct a B+ tree of order 3 <15, 20, 8, 7, 35, 40, 1, 2>



Total no of splits: 05

Rules for Insertion

- Odd Order
  - Leaf level → Copy middle element to its parent
  - non leaf level → move middle element to its parent
- Even Order → Apply left or right basis

Transaction Management

Transaction: Logical unit of database processing that includes one or more database operation

Issues with transaction (concurrent transactions)

→ Problems with concurrent execution
→ Promblems with ambguity in deciding when to make changes Permanant
→ Problems with integrity of data.

Transaction manager helps to maintain atomicity of Database

To resolve this issues we use transaction with ACID property

A: Atomicity → Either all operation is excuted or none occred (no partial transactional)
C: Consistency → Deals with integrity of data. tran^n opn should be one consistent step to another consistent state
I: Isolation → Opn of one tran^n should interfare with another tran^n. Concurrency control Protocols taken care of isolation property
D: Durability → It defines to make changes permanant and handel exception while storing data
Logs in DB ensure this property

Isolation Problems

1.

Name	Dept	Salary
John	Marketing	500
Hary	Finance	600
Sally	Retail	500

Here as Tx2 commit2 will be executed before commit 1 of Tx1 ~~transtam~~ : So here ther is Write-Read Conflict

Tx1 ——————————

10:00 Am Start Tx1

10:10 Am Set Hary.Salary = 700 (write)

10:20 Am Commit 1

Tx2 ——————————

10:05 Start Tx 2

10:15 Read Salary (Read)

10:15 Commit

WR-conflict / Dirty Read / Reading Uncommitted Data

2.

10:00 start TX1

10:10 set Hary.salay = 700

10.20 Commit

10:00 start Tx2

10:05 Read Salary ⟶ Reads Hary.Sal = 600

10:25 Read salary ⟶ Reads Hary.sal = 700] conflict

10:25 Commit 2

In First case we read uncommited data and then read committed data. Hence it is a R-W-R conflict

Unreapatable Reads (at Value ~~level~~ level) Conflict.

3.

10:00 Am Start Tx1

10:10 Am Insert (Jeff, sales, 1000) ⟶ Phantom Row

10:20 Am Commit 1 (Unrepeatable reads at Row level)

10:05 Am start TX2

10:05 Am Read sal ⟶ reads (500, 600, 500)

10:25 Am Read sal ⟶ reads (500, 600, 500, 1000)] Conflict

10:25 Am Commit 2

4. Race Condition

10:00 Am	Start Tx1	
10:10 Am	Set Mary = 700	Here Commit 1 races with Commit 2
10:20 Am	Commit 1	
10:10 Am	Start TX2	So it is a race condition.
10:15 Am	Set Mary = 800	
10:25 Am	Commit	

5. Incorrect Summary problem

10:00Am	start Tx1	
10:10Am	Set Mary 700	
10:20	Commit1	
10:05	start TX2	
10:05	sum (salary) → sum = 1600	Here incorrect summary is getting updated
10:15	sum (salary) → sum = 1700	
10:17	Commit 1	

6. Unstable Cursor

10:00 Am	Tx1	
10:10 Am	Red Salary	
10:15 Am	Read Salary	→ Gives error as we have dropped the salary column
10:10	start Tx2	
10:11	Drop Salary	
10:12	Commit	

Scheduling Transaction

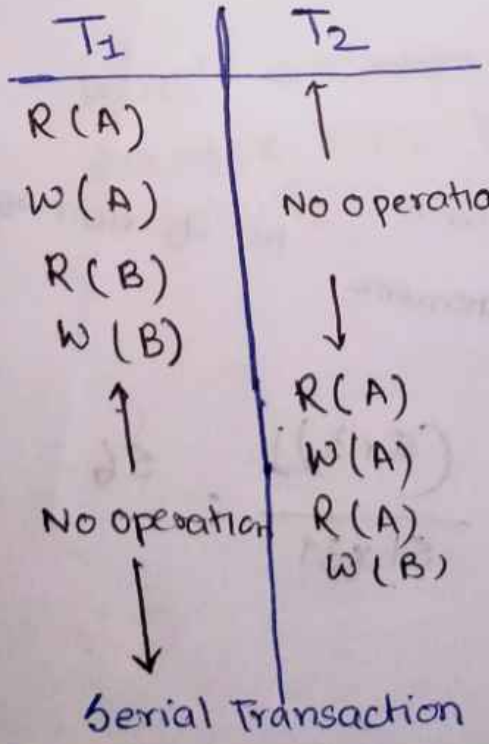
Transaction scheduling helps to solve above 6 isolation problems

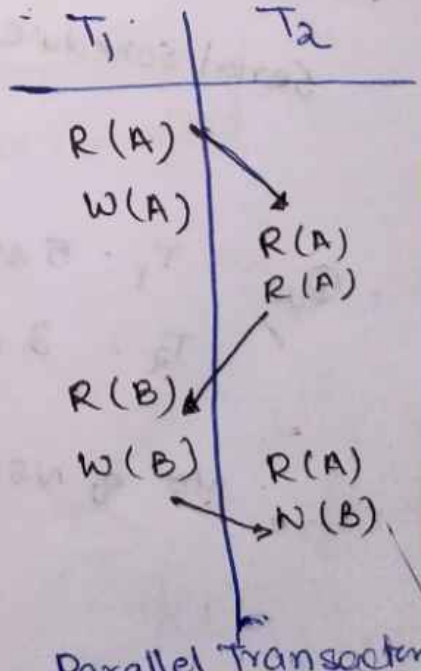
- Scheduling: The order of execution of operation for various transaction is called scheduling.
  Alternatively, when transactions are executing concurrently in an interleaved fashion then the order of execution of operation in various transaction is know as schedule.
  - Serial Schedule
  - Non Serial Schedule
    - Recoverable Schedule
      - Cascading
      - Non Cascading
        - Strict Recoverable
        - Non strict recoverable
    - Non recoverable

Serial Schedule: A schedule when operation of each transaction are executed conjugatively without any interleaved operations from other transaction.

Non Serial Schedule: A schedule where operations from a set of concurent transactions are interleared

eg S: T_1, T_2





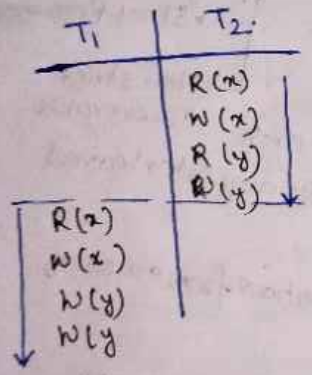
Schedule: If no of transactions = n

No of serial schedule = $n!$

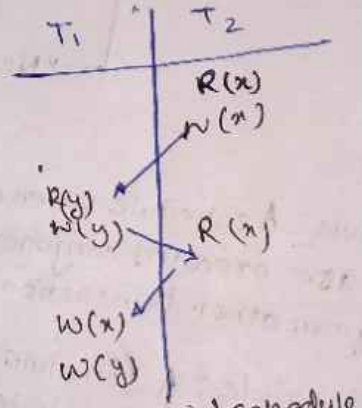
No of non-serial schedule $= \frac{(n_1 + n_2 + \ldots + n_n)!}{(n_1! * n_2! * \ldots * n_n!)}$

n_i = number of operations in ith transaction.

* Serial schedule is reliable than non-serial schedule
* Non-serial schedule, CPU utilization is more



Serial schedule



Non-serial schedule

Que: T_1 : 5 statements
T_2 : 3 statements

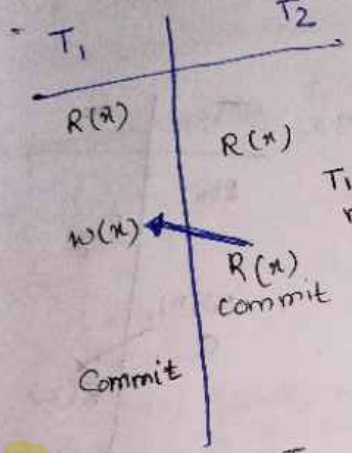
no. of non ~~Ser~~ serial arrangements.

No. of NSS $\frac{(5+3)!}{5! \times 3!} = 56$

Recoverable Schedule / Recoverability

A schedule is said to be recoverable if ~~transaction~~ is no need to role back a transaction

Let T_i and T_j be two transactions. If T_j reads a data item x previously written by T_i, then commit of T_i must appear before commit / abort operation of T_j

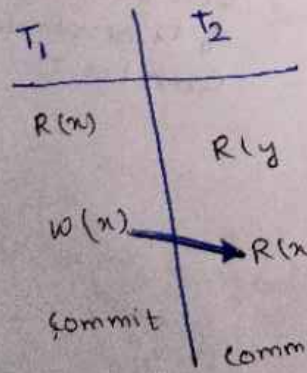


T_1 wrote data item x which is read by T_2

As per defination commit of T_1 must appear before T_2

Hence it is a non-recoverable schedule.

If T_1 is abort T_1, T_2 must be rolled back, but since T_2 is already commit, it cannot be rolled back, so it is non recoverable



Here it is a ~~non~~ recoverable schedule since T_1 is committed before T_2

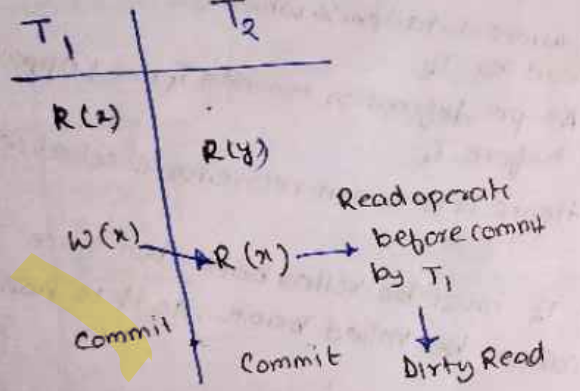
Recoverable

- Cascaded → Presence of dirty read operation
- Non Cascaded → No dirty read.
  - → Strict → Recoverable, and allow commit read and write op
  - → Non-strict → Recoverable + Committed Rea

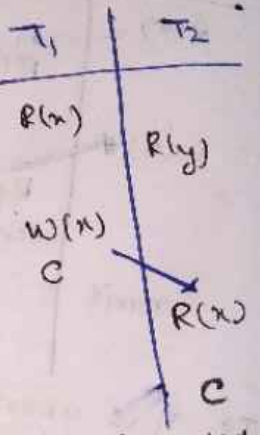
Dirty Read Operation: Reading uncommitted data is called dirty read.

eg:

S1:

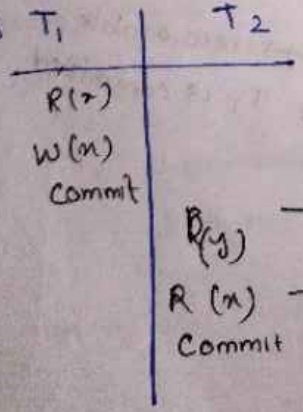


S2:



Non-cascaded Recoverable Schedule since we R(x) of T2 occurs after Commit in T1.

S3:



→ Serial schedule

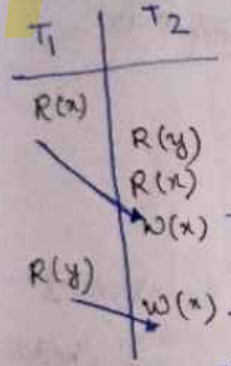
→ Cascadeless

Serialisation / Serializability

A non-serial schedule (S') with n transactions is said to be serializable if it is equivalent to a serial schedule (S) with n-transaction.

Serializability
- Conflict Serializability
- View Serializability

Conflict Serializability: Two operation are said to conflict serialized if
1. They must access same data item
2. Atleast one must be write opn
3. Operation must be from diffrent transactio

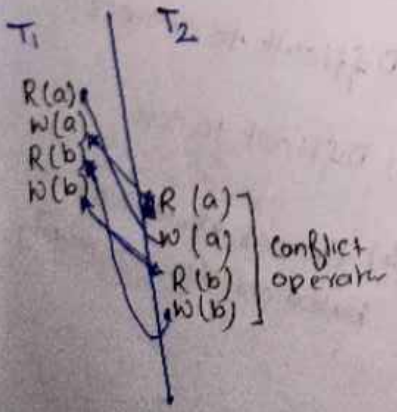


conflict operation.

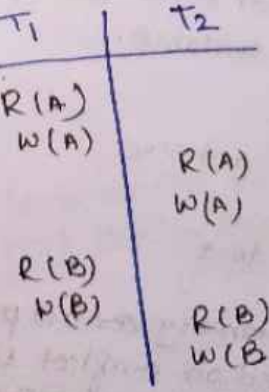
To qualify conflict operation there must be
1. Operations must be from 2 different transation
2. They have to access same data item.
3. Atleast one operation must be write operation

R(x) → x from T1
W(x) → x from T2

$R_i(x)$ and $S_j(x)$ is Conflict iff $x = y$, $i \neq j$ and R or S must be write



conflict operation



View Serializability: A serial schedule S and non serial Schedule S' is said to be view equilavent if they satisfy all the fallowing 3 condition

1. If T_1 reads initial dataitem in S, the T_1 should also read initial datavalue in S'
2. If T_2 perform final write operation on dataitem in S then T_2 also perform final write operation same dataitem in S'
3. If T_2 in S read the dataitem produced by T_1 then T_2 in S' must also read dataitem produced by T_1

If alleast one serial schedule from n! serial schedule is equivalent to S', then it is view serialisable.

Difference b/w Conflict and View Serializability.

Conflict Serializability	View Serializability
(1) All conflict serializable schedule are view seriable.	(1) All view serializable schedule may not be conflict serializable schedule.
(2) Easy to achieve	(2) Difficult to achieve
(3) Easy to test	(3) Difficult to test
(4) All concurrency control protocols are based on conflict serialisability except Thomas Write Rule.	(4) Thomas Write Rule is based on view serialisibity

Graph based method : Used to check conflict serialisability

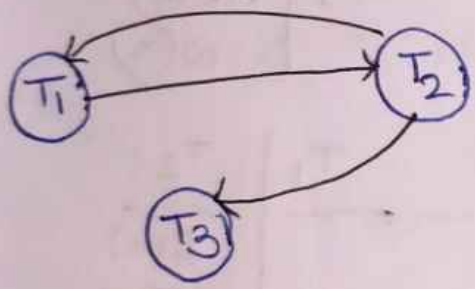
In a graph G:

no of nodes = no. of transactions in a schedule.

no of edges = Conflict operation. in a schedule.

Once the graph is drawn verify the graph for cycle.

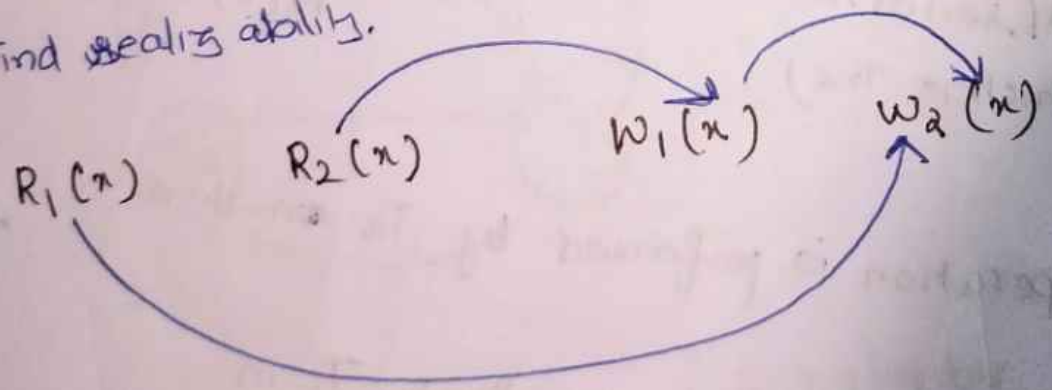
If graph has a cycle then it is not conflict equivalent, otherwise it is conflict equivalent (serializable)

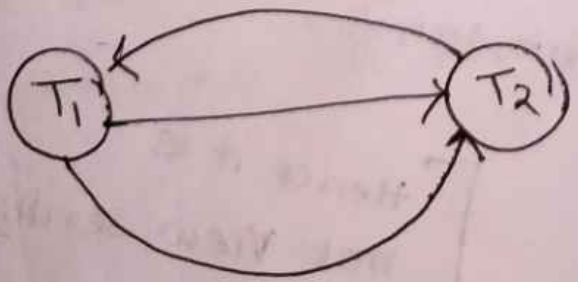


Cycle exists and hence is not conflict serializable

eg: $T_1 : R(x)$ $T_2 (R(x)$ $T_1 : W(x)$ $T_2 : W(x)$

Find serializability.





→ loop exists so not conflict-equivalent

T_1 : R(x) T_2 : R(x) T_1 : W(x) T_2 : W(x)

→ Not Conflic Serialisation

→ Hence we check for view serialisation

For View equiv
atleast 1 serial
schedule needs to
be equivalent to non-s
sche

(1) Caclulat all possible
serial schedule

T_1	T_2
R(x)	
	R(x)
W(x)	
	W(x)

For Secrial Schedule S)

(2) Take first Serial Sechud.
If T_1 Heads mitial data item
In non-serial, serial schedu T(1)
read x (which is True)

(1)

T_1	T_2
R(x)	
W(x)	R(x)
	W(x)

(2)

T_1	T_2
	R(x)
	W(x)
R(x)	
W(x)	

(3) Final Write operation is performed by T_2 on item x in

(4) T_2 reads data item which is not written by T_1 in Non Serial (3rd condition not satisfied

For Serial schedule T_2
1st condition not satisfy

Hence it is
not View serilizable
equivalent

eg:

T_1 : W(x)

T_1

All con

eg. T_1 :

T_1 : R(x) T_2 : R(x) T_1 : W(x) T_2 : W(x)

→ Not Conflic Serialisation

→ Hence we check for new serialisation

For View equiv
atleast 1 serial
schedule needs to
be equivalent to non-s
sche

(1) Caclulat all possible serial schedule

T_1	T_2
R(x)	
	R(x)
W(x)	
	W(x)

For Secrial Schedule S)

(2) Take first Serial Sechud.
If T_1 Heads mitial data item
In non-serial, serial schedu T(1)
read x (which is True)

(3) Final Write operation is performed by T_2 on item x in

(4) T_2 reads data item which is not written by T_1 in Non Serial (3rd condition not satisfied

For Serial schedule T_2
1st condition not Satisfy

Hence it is not View Serilizable equivalent

(1)

T_1	T_2
R(x)	
W(x)	R(x)
	W(x)

(2)

T_1	T_2
	R(x)
	W(x)
R(x)	
W(x)	

eg:

T_1 : W(x)

T_1

All con

eg. T_1 :

T_1	T_2	T_3
R(x)	R(y)	W(z)
	R(y)	W(y)
W(x)		
	R(y)	

Possibli

T_1	T_3	T_1	T_2	T_2	T_3
T_2	T_1	T_3	T_1	T_3	T_2
T_3	T_2	T_2	T_3	T_1	T_1

1st Possibli

T_1	T_2	T_3
R(x)	R(y)	W(x)
	R(y)	W(y)
W(x)		
	R(y)	

Here final write by T_1

Initial R by T_1

T_1	T_2	T_3
Rx		
W(x)		
	R(y)	
	R(y)	
	R(y)	
		W(x)
		W(y)

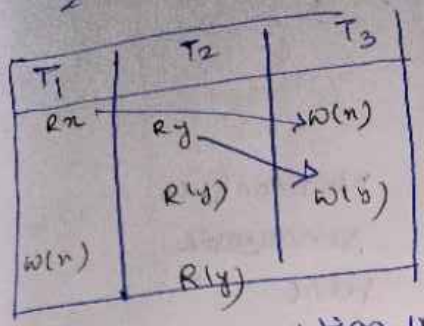
Final write by T_3

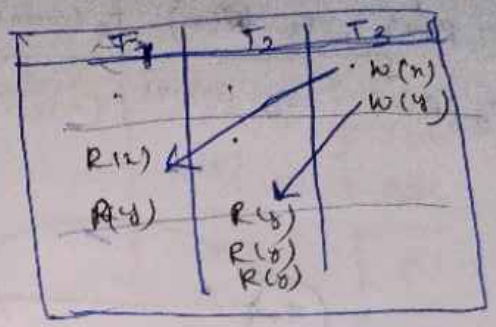
2nd

T_1
Rx
W(x)

T_1

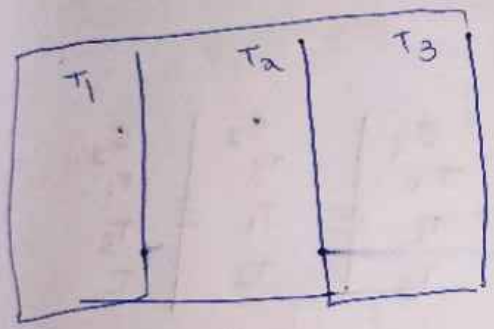
2nd Possibility

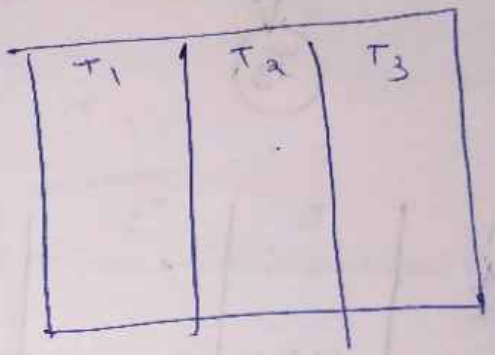




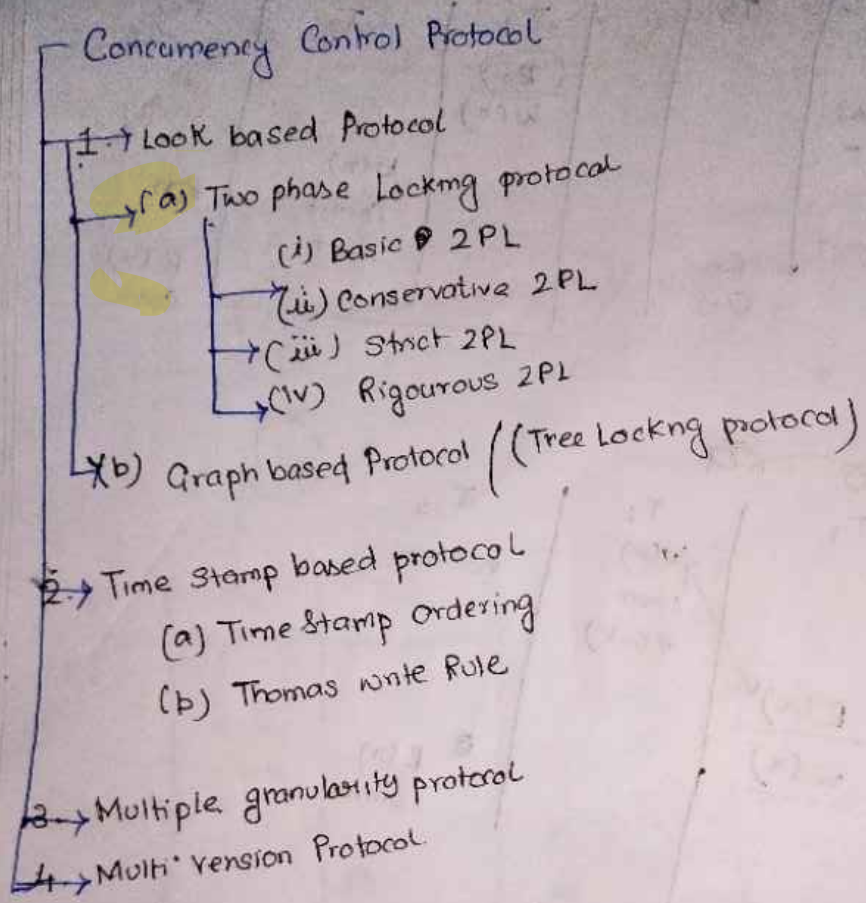
3rd condition invalid

Hence it is not checl.





If we check all the serial schedule we will find that none of the serial seduele is equivalent



CCP: Main objective is to ensure concurrent transaction by the RDBMS.

(I) Lock Based Protocol: One of the requirement for serializability is that data item must be accessed in mutual exclusive manner. ie, when one transaction accessing one data item at the same time no other transaction should modify the same data item

To implement this requirement, allow a transaction to access data item only if it is currently holding a lock on that data item.

Lock
2 modes to lock data items

(S) → Shared mode lock (S) $T_i \rightarrow Q$
If transaction T_i obtained shared mode lock on x in such case it can only read data item x but cannot write on data item x

Exclusive mode lock (X) $T_i \rightarrow Q$ → If exclusive lock is granted lock on q the T_i can perform both read and write operation on q

• Two phase Locking Protocol : Ensures conflict serializability

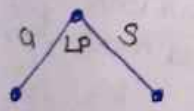
→ Any transaction obtains new lock in 2 phase,
(i) Growing phase
(ii) Shrinking phase.

Each transaction issue lock and unlock 2 phase

Any transaction may obtain locks but may not release any lock in growing phase

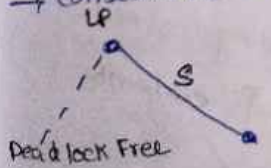
Any transaction may release locks but may not obtain any new locks.

→ Basic 2 PL : Has 2 phases (i) Growing (ii) Shrinking phase



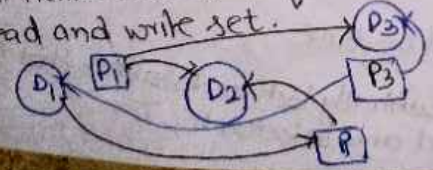
G: Transaction obtain Locks (R and W Locks)
S: Transaction release locks but donot obtain any new locks

→ Conservative 2PL : Here there is no growing phase but only shrinking phase hence it only release or unlock read and W locks



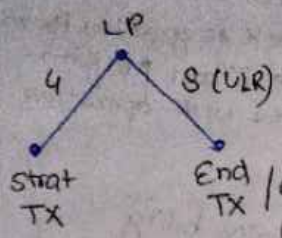
Before Transaction operation only, W-R locks are pre-declared.

Lock all items needed before execution begins by pre-declaring its read and write set.



No process is available any data and hence it is a deadlock.

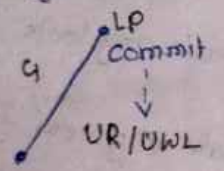
→ Strict 2PL: Has two phases. In growing phase, Tx issue read and write lock. but in shrinking phase it only unlocks read locks.



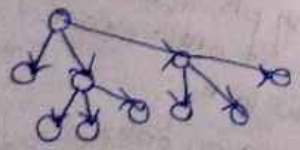
But after commit operation unlocks W-Locks Tx. doesn't release its W-Locks until after it aborts or commits

It is not deadlock free but ~~great~~ gurantees recoverable schedule. (Most popular version of 2PL)

→ Rigourous 2PL: It has only Growing phase, which is obtain R-W. Locks
After commit all R-W Locks are released.
No Lock is released until abort or commit



• Graph based Protocol: Also known as Tree lock protocol is applicable when DB is form trees of Data items
The simplest graph based protocol is Tree locking protocol and employs exclusive mode locks.



Here each transaction T_i can lock a dataitem atmost once and must obey following roles

(a) All Locks are exclusive locks
(b) First Lock by T_i can be any data item including root node
(c) Transaction T_i can lock data item q only if T_i currently locks parent of q.
dataitem may be unlocked at anytime
nsaction T_i cannot sequentially lock a dataitem that has be locked and unlocked by T_i

(2) Timestamp based Protocol: Unique identifiers timestamp is created to identify a dataitem.

timestamp: Unique identifier created by DBMS to identify a transaction.

It is assigned in the order in which transactions are submitted to the system. $T(S) = Ti \rightarrow$ Unique.

For every every tx, unique timestamp is assigned.

T_1 : 10:00AM

T_3 : 10:05AM

T_2 : 10:04 AM.

$TS(T_1) < TS(T_2) < TS(T_3)$

→ Time Stamp Ordering: Transaction submitting in order of occring.

W-TS(Q): Largest time stamp of any transaction that execute write dataitem Q successivly

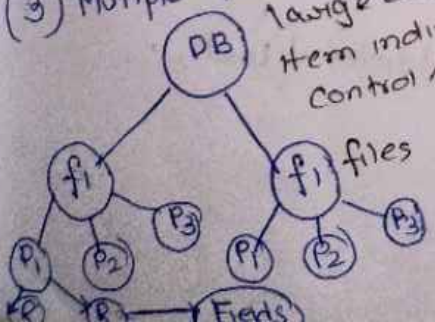
R-TS(Q): Largest timestamp of any trxn that execte Read data item Q successively

→ Thomas Write Rule: Based on view-serializability

It is a modified version of timestamp ordering protocol

It relaxes conflict sevializability and provide greater concurrency by rejecting absolute write operation.

(3) Mutiple Granuttality Protocol: Fallowed when dealing with large data. Here we have to unlock each item individually. The previous cocurrency control schems we used each individual data item as single unit on which synchronisation is the proble. So it is advantages to group the data item and create them as a single unit.



Case I: If Tx T_i needs to access entire DB, and uses locking protocols, then T_i must lock each individual data items which is time-consuming.

Disadvantge: Very Time consuming

Solution: If T_i issue single lock request to lock the entire Database and then perform oprn.

Case II: If trnx^n T_i needs to access few items then it should not require to lock the entire data. In such case an item can be one of the following

(a) DB-Record
(b) Field Value of DB-Record
(c) A disk block
(d) Whole file
(e) Whole DB (Highest size granule)

Size of Granule ↓

Data item Granularity: The size of database item is called database granularity

Granularity → Fine granularity: overall item size
→ Coarse Granularity: Large item size

Here there are 3 types of locks. We used shared and exclusive lock when a trxn locks a node, all descendant of the node in the same lock node

To make multiple granularity level, additional lock, intention lock is used.

Intension Lock

- Intension Shared (IS)
- Intension Exclusive (IX)
- Shared Intension Exclusive (SIX)

IS: Shared Lock is requested on some desendant node

IX: Exclusive lock is requseted on some dependant node

SIX: Current node is locked in shared mode but exclusive lock will be requested.

CCP - 10mrks + Txn = 15 mrks
RQ/SQL - 10mrk
ER - 15mrks
FDs: 10mrks + 10mrks
~~Txn Managment~~ : 10mrks
Indexing: + B|B+ Trees = 10

7-questions
↓
out of 5