

G **T** **S** **G** **T**

5 **E** **5** **E**

N **O** **N** **O**

O **S** **O** **S**

G **T** **G** **T**

5 **E** **5** **E**

N **O** **N** **O**

O **S** **O** **S**

G **T** **G** **T**

5 **E** **5** **E**

Books :

Principles of database - JD Ullman

Database System Concepts - HF Korth & A Silber

Database Management Systems - Raghuramakrishnan

Raw Data -

Name
A. Shukla

Field(s) like Roll No., Name

Records Collection of field

Similar kind

Data File → Collection of Record

e.g. teacher file, student file

Some relationship exists

Collection of Interrelated Datafile

DBMS ⇒ Software can create Data & Manipulate

Allows us

What are the advantages of creating file & manipulation over HLL (File creation & Manipulation)?

① Data Redundancy & Inconsistency

Roll No. , Name , HALL , Game Name , Fees

Possibility of Redundancy

(Roll No., Name, Hall) (G.Name , Fees) (Roll , G.Name)

Attributes repeated to make connections

→ When we remove redundancy, inconsistency also removed.

② Easy Access of Data

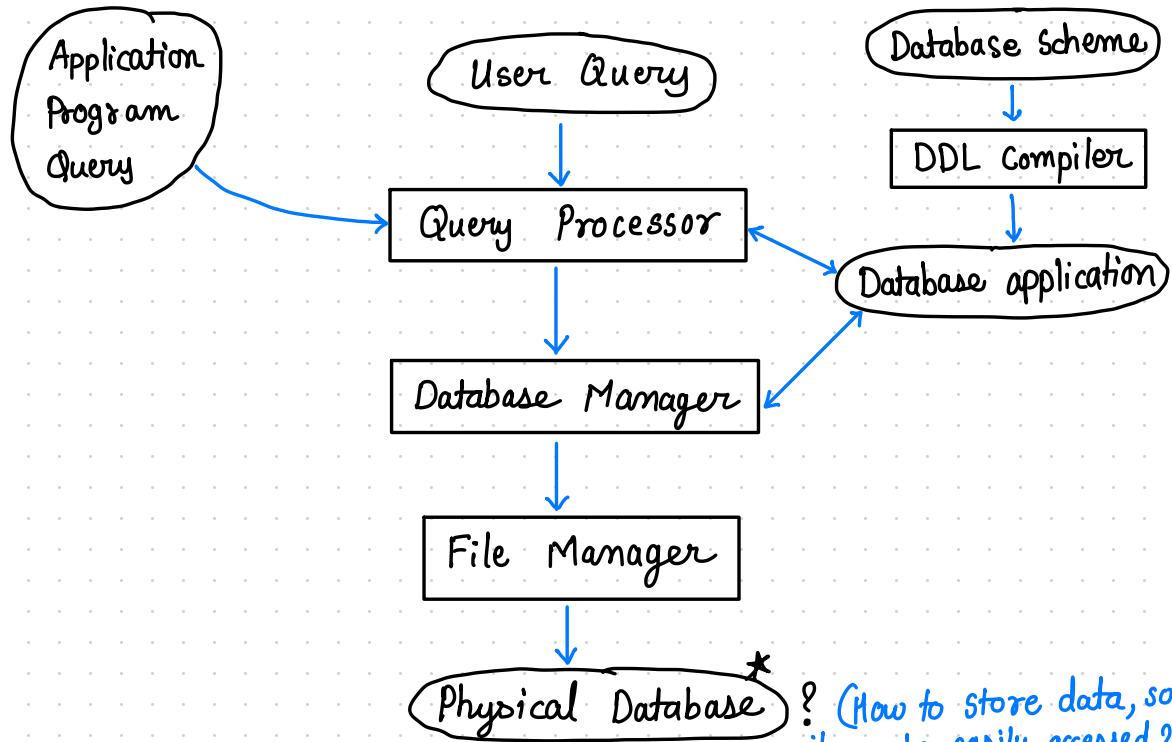
③ Multiple Users

④ Security Problems

⑤ Integrity Problems → like in Railway System - Lower birth to Senior Citizen or Ladies

⑥ Synchronization
 [→ Multiple reader
 [→ But only one can manipulate at a time to avoid inconsistency.]

⑦ Recovery Process (To prevent Info Loss)



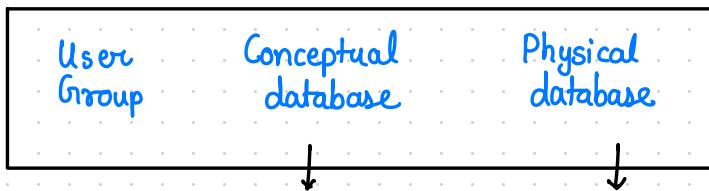
DDL - Data Definition Language

DML - Data Manipulation Language

? (How to store data, so it can be easily accessed?)
 (This to study)

- Database Design
- Query Processing
- File Organization

* Levels of Abstraction :



(The way designer) wants (How Physically stored)

Object Based Logical Models

Entity Relationship Model

Binary "

Semantic "

Infological "

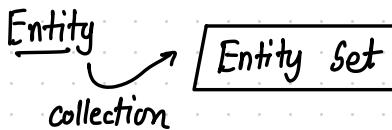
Record based logic Models

Relational Model

Network Model

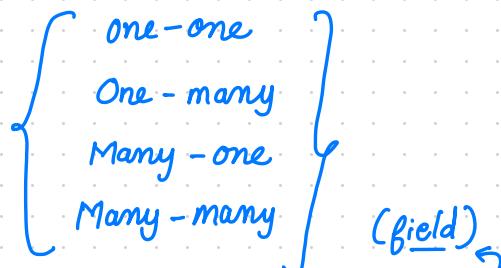
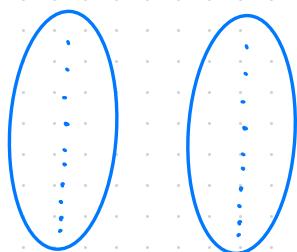
Hierarchical Model

Entity Relationship Model (E-R)



e.g. Customer (C.Name, P.No., C.City)

Employee (emp.name, p.No.)



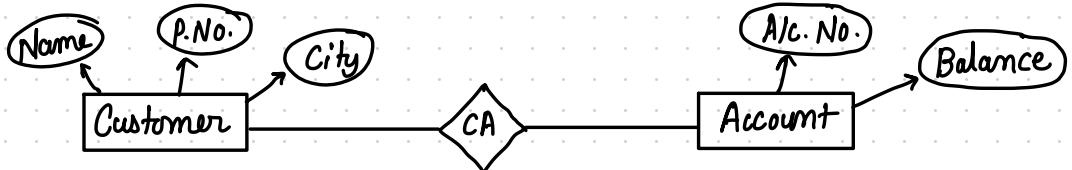
Primary Key → Collection of one or more than one attribute whose value will uniquely identify only one record.

PK

Minimum No. of attribute which uniquely identify.

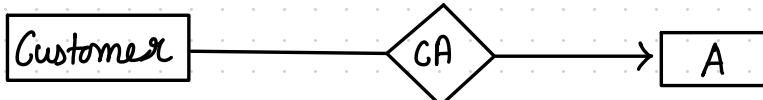
Notations



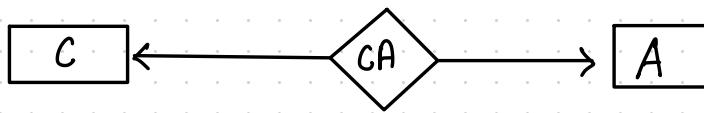


Many-Many (No arrow on both sides)

Customer have an account & an account may belong to multiple customer.



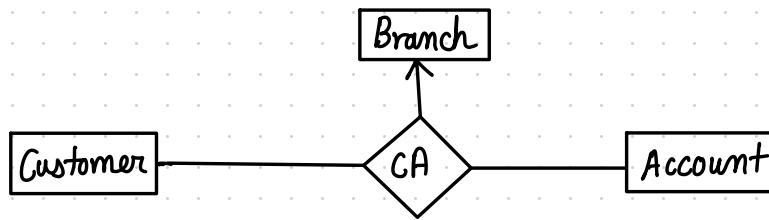
Many-one



One-one



One-many



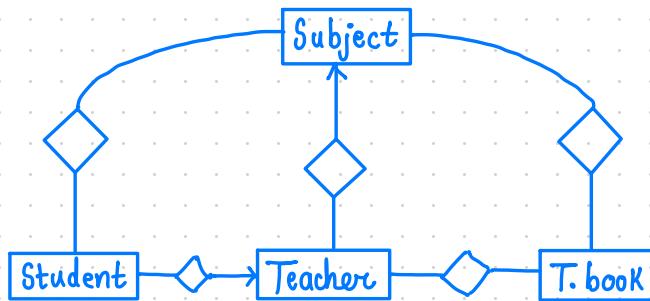
One customer may have multiple Acc., One acc. belong to many customer but in one branch.

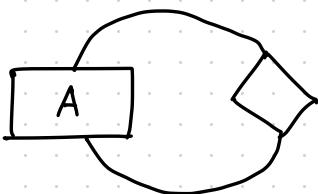
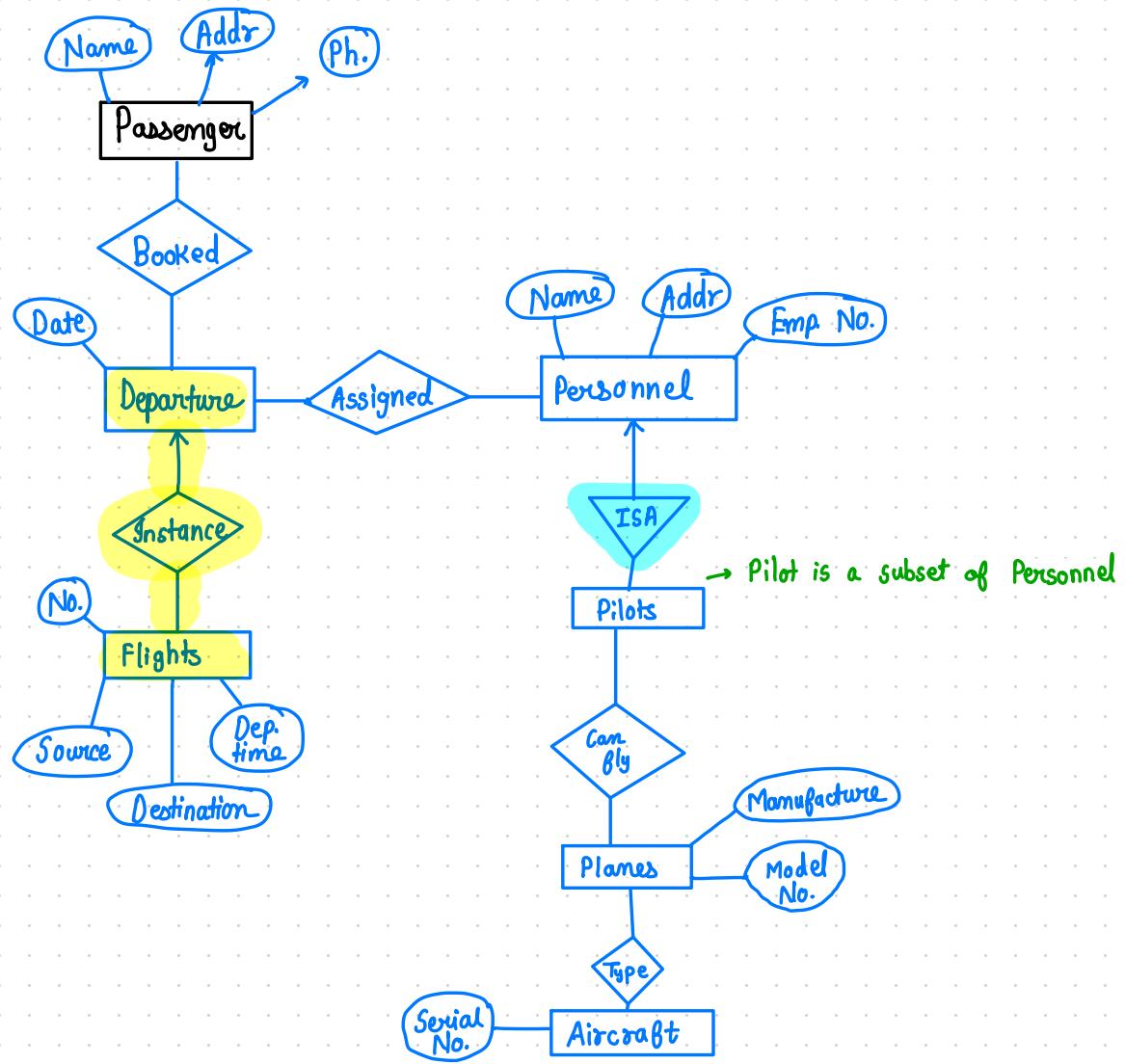
⇒ Draw one E-R diagram for Database System.

(Textbook , Subject , student , Teacher)

- ① For each Subject each student is taught by a single Teacher.
- ② Each teacher teaches only one subject.
- ③ Each Subject is taught by several teachers.
- ④ Teachers choose their respective text book.

Ans:





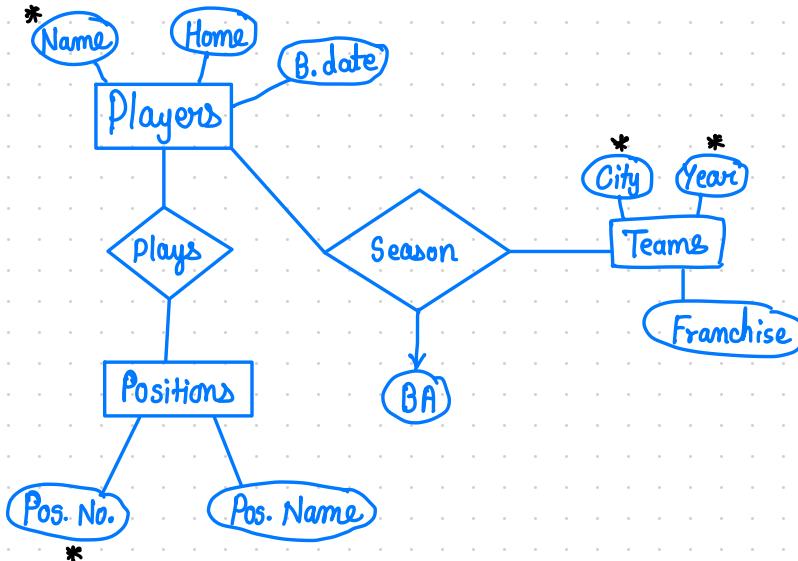
why we need to make a relationship?

E1	1000	M1		
:	:			
M1	900	-	-	
E1	1000	M1	M1	900
-	-	-	-	-

Relational Data Model
 Hierarchical " "
 Network " "

* Relational Data Model

* = KEY



Players (Name, Home, B.date)
 Positions (Pos. No., Pos. Name)
 Teams (City, Year, Franchise)

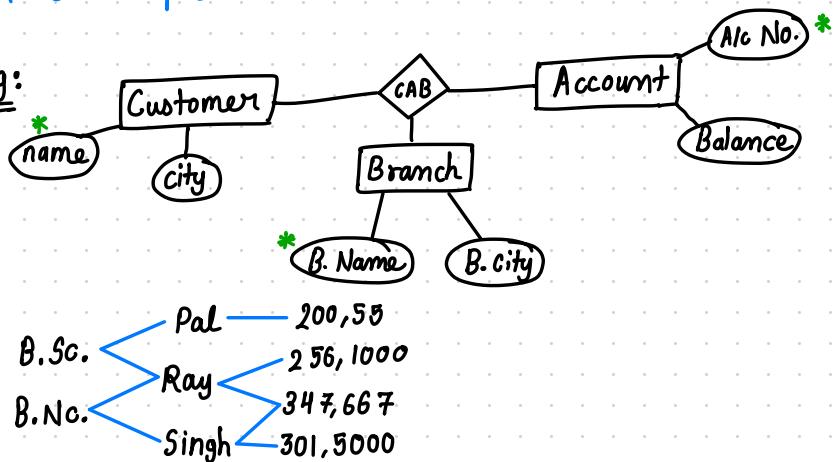
KEY of the Relationship
Plays (Name, Pos. No.)

Season (Name, City, Year, BA)

Q. Player X, which position?

→ Go to positions table

Eg:

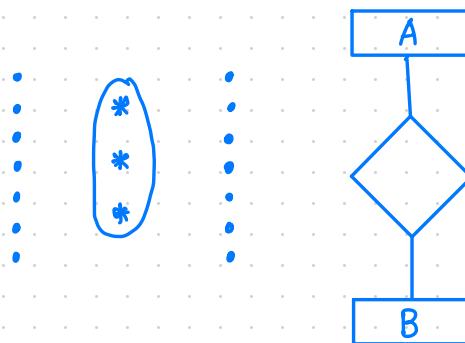


CAB

<u>name</u>	<u>B.Name</u>	<u>Ac. No.</u>
Pal	B.Sc.	200

Network Model

→ ER model where relⁿ is restricted to binary & many-one



Create a
Temporary set such that
its many one with
A & B.

Branch

B.name

B.Sc.

B.Nc.

B.City

—

—

Customer

C.Name

Pal

Ray

Singh

C.City

—

—

Account

Ac.No.

200

256

347

301

Balance

—

—

—

—

CAB

C.name

Pal

Ray

Ray

Singh

B.name

B.Sc

B.Sc

B.Nc

B.Nc

Ac.No.

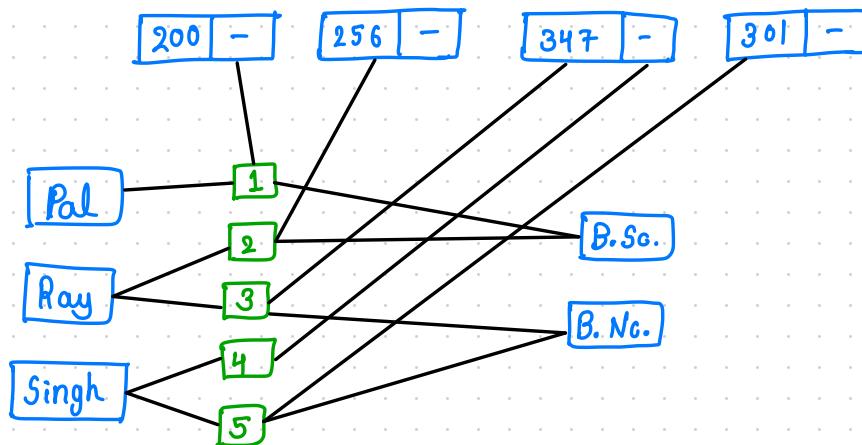
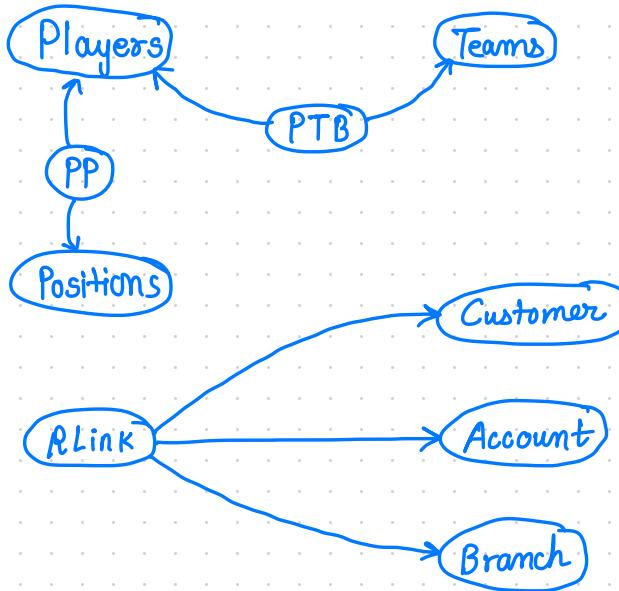
200

256

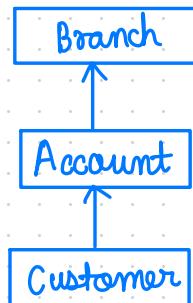
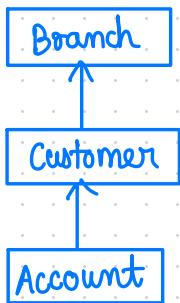
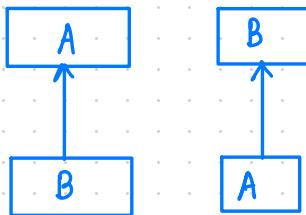
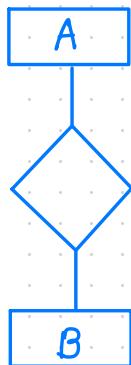
347

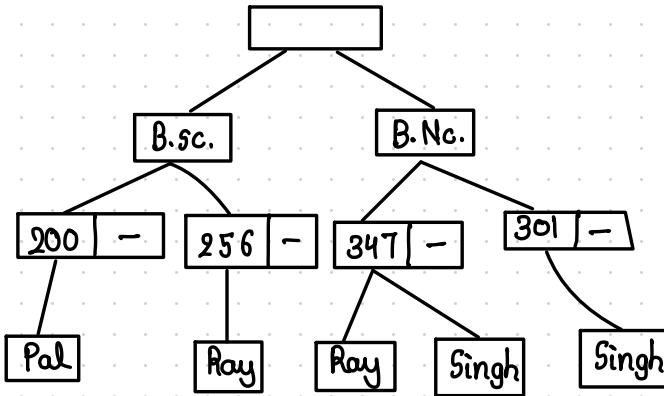
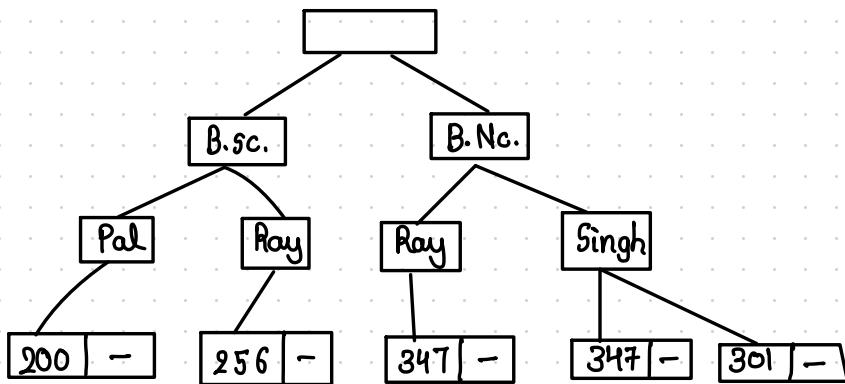
301

Example :



Hierarchical Model





Do problems from book

★ Database Design :-

* Relational Database Design :-

→ Design is good if it is free from certain things:

1) Redundancy

2) Updation Anomalies

3) Insertion Anomalies

4) Deletion Anomalies → only 1 student for hockey & he says NO, if you delete, your data is lost about fees of hockey.

<u>S. Name</u>	<u>RN</u>	<u>HALL</u>	<u>GAME</u>	<u>FEE</u>
----------------	-----------	-------------	-------------	------------

(S.Name, RN, Hall)

(RN, Game)

(Game, Fee)



Redundancy in Key only
& Nowhere else.

Functional Dependency (FD)

$U \rightarrow$ Universal set of attributes

$U_j \subset U$: U_j is subset of U .

$X \rightarrow Y$; X & Y are subset of U_j

X functionally determines Y

Y is functionally dependent on X .

X = determinant



Y = Dependent attribute

Trivial FD.

$X \rightarrow Y$, where
 Y is a subset of X .

Eg: $s_id \rightarrow s_id$

$LHS \cap RHS \neq \emptyset$
 $s_id \rightarrow sname$

Non-trivial : $X \cap Y = \emptyset$
 $s_id \rightarrow sname$

tuple = Record
in database

Eg: $\begin{matrix} s_id \\ \downarrow \\ \text{student} \end{matrix} \rightarrow s.name$

X determines Y

Relation $R : X \rightarrow Y$ $\mu_1 \& \mu_2$ (tuples)

If $\mu_1[x] = \mu_2[x] \Rightarrow \mu_1[y] = \mu_2[y]$

$\mu_1 \rightarrow$	X	Y
	x	y
$\mu_2 \rightarrow$	x	y

$R(A, B, C, D) \Rightarrow$ find FDs.

A	B	C	D
a ₁	b ₁	c ₁	d ₁
a ₁	b ₂	c ₁	d ₂
a ₂	b ₂	c ₂	d ₂
a ₂	b ₃	c ₂	d ₃
a ₃	b ₃	c ₃	d ₄

$A \rightarrow C$
 $D \rightarrow B$ ✓ → If x val. is diff., there
 can be any y value.
 $G \rightarrow A$
 $AB \rightarrow D$
 $B \rightarrow D$ ✗

$A \rightarrow A$ ✓ (Trivial FD)

$A \rightarrow B, B \rightarrow C$ logically \Rightarrow A \rightarrow C elements of a closure

$\mu_1(a) = \mu_2(a) \rightarrow \mu_1(b) = \mu_2(b)$
 & b b
 G.: a a c c
 c c

⇒ Closure of a set of FDs

$F \xrightarrow{+} F^+$

$$F^+ = \{X \rightarrow Y \mid F \xrightarrow{+} X \rightarrow Y\}$$

$R(A, B, C)$; $F = \{A \rightarrow B, B \rightarrow C\}$

$$F^+ = \{A \rightarrow C, A \rightarrow ABC, AB \rightarrow ABC, \dots\}$$

* Primary Key

→ Minimum No. of attributes which determines one _____

$$R(A_1, A_2, \dots, A_n)$$

$$X \subseteq A_1, A_2, \dots, A_n$$

1) $X \rightarrow A_1, A_2, \dots, A_n$ is in F^+

2) $Y \subseteq X \ \& \ Y \rightarrow A_1, A_2, \dots, A_n$ is in F^+

* Armstrong's Axioms :

Trivial

Determining itself

1) Reflexivity : If $Y \subseteq X \subseteq U$, then $X \rightarrow Y$

2) Augmentation : If $X \rightarrow Y$ and $Z \subseteq W$,
then $XZ \rightarrow YZ$

(Augmenting Z on both sides)

** 3) Transitivity : If $A \rightarrow B$, $B \rightarrow C$, then $A \rightarrow C$

$$R(\text{CITY}^{\text{Street}}, \text{ST}, \text{PIN})$$

$$F = \{ \text{CITY ST} \rightarrow \text{PIN}, \text{ PIN} \rightarrow \text{CITY} \}$$

What are Key? Tell logically using axioms :

Ans.

$$\text{CITY ST} \rightarrow \text{PIN}$$

$$\therefore \text{CITY ST} \rightarrow \text{PIN CITY ST} \quad (\text{2}^{\text{nd}} \text{ ax. - Augmentation})$$

$$\text{PIN} \rightarrow \text{CITY}, \text{ PIN ST} \rightarrow \text{CITY ST}$$

$$\therefore \text{PIN ST} \rightarrow \text{CITY PIN ST}$$

→ You may have more than 1 Key.

THUMB RULE \rightarrow If an attribute is not in RHS of any FD then that attr. must be a part of Key.

* Inference Rules

1. Union Rule : $\{x \rightarrow y, x \rightarrow z\} \models x \rightarrow yz$

logic : $x \rightarrow xz, xz \rightarrow yz$
 $x \rightarrow yz$

2. Pseudotransitivity Rule : $\{x \rightarrow y, wy \rightarrow z\} \models xw \rightarrow z$

3. Decomposition Rule : If $x \rightarrow y \ \& \ z \subseteq y$, then $x \rightarrow z$

OR: If $x \rightarrow yz$ then $x \rightarrow y \ \& \ x \rightarrow z$

4. If $x \rightarrow yz$, then $x \rightarrow y \ \& \ x \rightarrow z$

* Closure of a set of attributes :

$$F \quad x^+ \quad x \rightarrow A$$

LEMMA : $x \rightarrow y$ follows from Armstrong's ax, iff $y \subseteq x^+$

Theorem : Armstrong's ax are sound & complete. (You can't derive any extra FDs)

Book of Ulman \rightarrow see proof of both above

* X^+ algorithm of Bernstein :

1. Let $N=0$ and $X(N) = X$
 2. If there is an FD: $A \rightarrow B$ in F
whose LHS A is contained in $X(N)$
but RHS B is not contained in $X(N)$
then $X(N+1) = X(N) \cup B$
otherwise terminate.
 3. $N = N+1$, Go to step 2
-

$$R = (A, B, C, D, E, G)$$

$$F = \{ AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, D \rightarrow EG, BE \rightarrow C \\ CG \rightarrow BD, CE \rightarrow AG \}$$

$$(BD)^+ = ?$$

Ans.

$$N=0 ; X(N) = BD$$

$$D \rightarrow EG ; X(1) = BDEG$$

$$BE \rightarrow C ; X(2) = BDEGCA$$

$$C \rightarrow A ; X(3) = BDEGCA$$

Family Income Make of Cor Salary Spouse's Salary Job City of Residence Luck Education Effort Sincerity
Eg: R (FI, MOC, SAL, SS, JOB, COR, LUCK, ED, EFF, SEN)

$F = \{ FI \rightarrow MOC, SAL \rightarrow SS, JOB \rightarrow COR, LUCK \rightarrow ED, ED \rightarrow JOB, JOB \rightarrow SAL \}$
 (HINT : Thumb Rule)

Ans : $(LUCK, SS, JOB, SEN, EFF)^+ \rightarrow$ will give you all the attributes.
 $(LUCK, SS, ED, SEN, EFF)$

Partial Dependency :

$X \rightarrow Y$ $X^1 \subset X$ s.t. $X^1 \rightarrow Y$
 then elementary FD
 is present

$AB \rightarrow C$
 $A \rightarrow C$

$F \rightsquigarrow f: X \rightarrow A$ $B \subseteq X$
 $X^1 \leftarrow (X - \{B\}) \rightarrow A$ is in F^+
 extraneous attribute

$F = \{ AB \rightarrow DEF, AC \rightarrow G, A \rightarrow C \}$
 $\Rightarrow (A)^+ = ACG$ $\underbrace{(AC - \{C\}) \rightarrow G}_{f} \checkmark$
 $(AC)^+ = ACG$ extraneous attribute

$F' = \{ AB \rightarrow DEF, A \rightarrow G, A \rightarrow C \}$
 $\{ AB \rightarrow DEF, A \rightarrow CG \}$

$X \rightarrow Y$ $\{X - B\} \rightarrow Y$
 $B \subseteq X$ Partial dependency
Eg: $AB \rightarrow C ?$ $A \rightarrow C$
 $B = \text{extraneous attribute}$

Redundant FD

f in (F) if $(F-f)^+ = F^+$

$f: X \rightarrow A$ } Redundant FD

→ A set of FDs F is Minimum if there is no set G with less FDs than F s.t. $G^+ = F^+$ [There is no redundant FD] & it has least possible no. of FDs.

Is this set minimum? $\{A \rightarrow B, A \rightarrow C\}$ $\{A \rightarrow BC\}$

$\begin{matrix} F \\ x \end{matrix}$ $\begin{matrix} G \\ \checkmark \text{ min.} \end{matrix}$

L-minimum

left-hand side

: A set of FDs F is L-min. if
1) F is minimum
2) There exist NO partial dependency between _____

Is it L-min?

$\{ABC \rightarrow D, A \rightarrow B\}$

→ 1st check is it minimum? → YES

B is redundant

⇒ $\{AC \rightarrow D, A \rightarrow B\}$

LR - Minimum : A set of FDs F is L-min. & also for each FD it is minimum on the RHS.

$A \rightarrow (AB)$, Not min.

$R(A, B, C, D, E, F)$

$$F = \{AB \rightarrow E, AC \rightarrow F, AD \rightarrow B, B \rightarrow C, C \rightarrow D\}$$

Solⁿ:

$AB \rightarrow E$	$: NR(AB)^+ = ABCDF$	<small>E is not there</small>	<small>1) check for minimum</small>
$AC \rightarrow F$	$: NR(AC)^+ = ACDBE$	<small>F is not there</small>	<small>NR: Non-Redundant</small>
$AD \rightarrow B$	$: NR(AD)^+ = AD$	<small>B is not there</small>	<small>No Redundancy</small>

$$AB \rightarrow E; A^+ = A, B^+ = BCD$$

$$AC \rightarrow F$$

$$C^+ = CD$$

No partial dependency

$$AD \rightarrow B$$

$$D^+ = D$$

Can I reduce the No. of FDs? What changes we will make?

Eg: $F = \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, D \rightarrow EG, BE \rightarrow C, CG \rightarrow BD, CE \rightarrow AG\}$

Find L-R Min.

→ (1) Rewrite the FD s.t. on the RHS of each FD there is only one attribute.

$$\begin{array}{ll}
 \text{NR } AB \rightarrow C & \text{NR } BE \rightarrow C \\
 \text{NR } C \rightarrow A & \text{NR } CG \rightarrow B \\
 \text{NR } BC \rightarrow D & * R \text{ } CG \rightarrow D \\
 * R \text{ } ACD \rightarrow B & \text{NR } CD \rightarrow D \\
 \text{NR } D \rightarrow E & * R \text{ } CE \rightarrow A \\
 \text{NR } D \rightarrow G & \text{NR } CE \rightarrow G
 \end{array}$$

(2) check for minimum

1) Find Redundant FDs of this set.

ACD gives you $ACDEG$ (B)

Similarly,

$$\begin{array}{l} CG \rightarrow D \\ \& CE \rightarrow A \end{array}$$

Hence $ACD \rightarrow B$ is Redundant

$AB \rightarrow C$	$BE \rightarrow C$
$C \rightarrow A$	$CC_1 \rightarrow B$
$BC \rightarrow D$	$CD \rightarrow D$
$D \rightarrow E$	$CE \rightarrow C_1$
$D \rightarrow G_1$	

2) Check for each FD whether
it's partial dependency or
not.

$AB \rightarrow C \Rightarrow$ find A^+ & B^+

\downarrow

A

\downarrow

B

No partial dependency ✓

3) Is it minimum or can reduce it?

→ If for any 2 FDs if LHS is same,
merge the RHS.

→ Each FD represents one table, we will have to carry
less data.

Decomposition of a Relational Scheme :

$$R = (A_1, A_2, \dots, A_n)$$

$$P = \{R_1, R_2, \dots, R_k\} \rightarrow k \text{ subsets}$$

$$R = R_1 \cup R_2 \cup \dots \cup R_k$$

It must satisfy 2 properties when we decompose :

1) Lossless Join property → else it will lose information when
we join them.

$$R : R_1, R_2, \dots, R_k$$

D is a set of dependencies

All the tuples of Relation R $= \Pi_{R_1}(x) \bowtie \Pi_{R_2}(x) \bowtie \dots \bowtie \Pi_{R_k}(x)$

Natural Joint

A	B	C
a ₁	b ₁	c ₁
a ₂	b ₂	c ₂
a ₃	b ₃	c ₃

R(A, B, C)

$\pi_R(A, C)$

A	C
a ₁	c ₁
a ₂	c ₂
a ₃	c ₃

Theorem : If $\delta = (R_1, R_2)$ is a decomposition of R

& F is a set of dependencies

then δ is lossless join decomposition w.r.t F

iff.

$$(R_1 \cap R_2) \rightarrow (R_1 - R_2)$$

or

$$(R_1 \cap R_2) \rightarrow (R_2 - R_1)$$

This FD exists in F or F^+ .

Proof in Ullman

→ Their intersection determines their subtraction.

Example : R(A, B, C) $F = \{A \rightarrow B\}$

$$R_1(A, B), R_2(A, C)$$

Does it satisfy lossless join property?

$$R_1 \cap R_2 = A$$

$$R_1 - R_2 = B$$

$(A \rightarrow B) \Rightarrow$ can be in F or F^+

YES

$$R_1(A, B), R_2(B, C)$$

NO

$$\gamma = \{a_1, b_1, c_1, a_2, b_1, c_2\}$$

$$\pi_{AB}(\gamma) = \{a_1, b_1, a_2, b_1\}$$

$$\pi_{BC}(\gamma) = \{b_1, c_1, b_1, c_2\}$$

$$\pi_{AB}(\gamma) \bowtie \pi_{BC}(\gamma) = \{a_1, b_1, c_1, a_1, b_1, c_2, a_2, b_1, c_1, a_2, b_1, c_2\}$$

should be equal to $\underline{\underline{\gamma}}$, but its not.

2) Preservation of Dependencies :

$R \Rightarrow R_1, R_2, \dots, R_k$ $F \rightarrow$ set of dependencies
 Union $\leftarrow \bigcup_{i=1}^k \underbrace{\Pi_{R_i}(F)}_{\text{dep. which exists in } F \text{ where attributes}} = F$
 R_i are involved.

→ we decomposed into k - sub-relations s.t. when we join them we get original Relation.

→ if I don't preserve some FDs while decomposing
 WHAT WILL HAPPEN ?

I may not be able to answer all queries.

BUT

→ if loss-less join is not there , our answer to query will be wrong. So, its more imp. than preserving property.

Example : $R(C, S, P)$

$$F = \{CS \rightarrow P, P \rightarrow C\}$$

$$R_1(S, P) \quad R_2(C, P)$$

⇒ 1) Is it satisfying loss-less join property ?

YES

2) Is it satisfying preservation of dependency ?

$$\Pi_{R_1}(F) = \{S \rightarrow S, P \rightarrow P\}$$

$$\Pi_{R_2}(F) = \{P \rightarrow C\} \rightarrow \text{does not preserve dependency.}$$

Example :

$R(A, B, C, D)$

$F = \{A \rightarrow B, C \rightarrow D\}$

$R_1(A, B)$

$R_2(C, D)$

\Rightarrow Not loss-less

\Rightarrow preserve dependency ?

$$\begin{aligned}\pi_{R_1}(F) &= \{A \rightarrow B\} \\ \pi_{R_2}(F) &= \{C \rightarrow D\}\end{aligned}\left.\right\} \text{preserve dependency } \checkmark$$

1. Candidate Key : → minimum , its closure can give all attri.
→ Anything extra added , then it becomes Super Key.

These are individual columns in a table that qualifies for the uniqueness of all rows

→ Prime attri. = that which makes the candidate Keys.
Non-prime attri = remaining others.

2. Primary Key or KEY : RN

is the columns you choose to maintain uniqueness in a table.

3. Foreign KEY : RN , Hall

is a Collection of field(s) in one table that uniquely identifies a Row of another table.

primary KEY of table T2 but it is there in table T1 also.

4.) Super Key : RN , S. Name

If you add any other column or attribute in a primary key then it will become a super key.

* Normal Forms :

(Normalisation) is the process by which when you decompose it, that will be free from anomalies.

* Boyce Codd Normal Form (BCNF)

$$R \quad F: X \rightarrow Y$$

(Not just a member but
↑ whole KEY)

- If Y is not a subset of X and if X is a Key then $X \rightarrow Y$ is in BCNF.
- If all FDs satisfy this property then Relation R is in BCNF.

Eg: R (PAN, PI, DI, DRUG_i, QTY, COST)

$$F = \{ PAN \rightarrow PI, PI \rightarrow DI, PI DRUG_i \rightarrow QTY, \\ DRUG_i QTY \rightarrow COST \}$$

Is it BCNF?

⇒ KEY : {PAN, DRUG_i} , PAN → PI is not in BCNF

Q. How to decompose relation R into subrelation s.t. each sub-relⁿ is in BCNF?

① R₁ (PAN → PI) , R₂ (PAN, DI, DRUG_i, QTY, COST)
 sub. relⁿ sub relⁿ ↗ all remaining attributes except RHS of R₁
 ↘ (PI) FDs available in F or F⁺:

Not a KEY (DRUG_i QTY → COST)

Find out FDs involved in attributes of R₂ only, those may be in F or F⁺.

→ Decompose R₂ in sub-relⁿ:

② $R_3(DRUG, QTY, COST)$ $R_4(PAN, DI, DRUG, QTY)$

↓ possible FD

$PAN \rightarrow DI$ { it is in F^+ , not in F , we can use.

③ $R_5(PAN, DI)$

$PAN \text{ DRUG} \rightarrow QTY$

④ $R_6(PAN, DRUG, QTY) \rightarrow$ contains KEY → you can find out any attribute

decomposition of R is R_1, R_3, R_5 & R_6

↓
that satisfies the
loss-less join property.

Theorem: BCNF decomposition produces loss-less join decomposition always.

we have lost $PI \rightarrow DI$ ⇒ BCNF may not preserve dependencies.

Example: $\{C \rightarrow A, AE \rightarrow B, BF \rightarrow C, CD \rightarrow EF, EF \rightarrow AD\}$

$R(A, B, C, D, E, F)$

{C, A}

{B, C, D, F, F}

✓

{B, F, C}

{B, F, D, F}

✓

{B, D, E}

{B, F, D}

✓

Also KEY

CD → Key

B F D E
+
B F C D E A

BFD → EF
BD → E

BF → C CD → EF EF → AD
AE

★ Prime Attribute

A \Rightarrow prime if A is a member of a key of R, otherwise an attribute we call it as Non-prime attribute.

$$R(A, B, C, D), F = \{AB \rightarrow C, B \rightarrow D, BC \rightarrow A\}$$

\Rightarrow Key : AB, BC

★ Third Normal Form: (3NF)

If : R $X \rightarrow A$: A is not in X

then either X is a key of R or A is prime attribute
 $\xrightarrow{\text{in the Key}}$

BCNF $>$ 3NF

★ Bernstein's 3NF Algorithm

Input : A set of attributes

Output: decomposed sub-relation which is free from anomalies

- 1) Rewrite the FDs s.t. there is only 1 attribute on the RHS of each FD.
- 2) Rewrite the list of FDs s.t. \exists no redundant FD.
- 3) Rewrite the FDs s.t. No proper subset of LHS functionally determines RHS. (Removing partial dependency)
- 4) Combine the dependencies with same LHS using the Union Rule.

→ then if $X \rightarrow Y$ is in the list, make a decomposition with attributes $\{X, Y\}$ \Rightarrow This will ensure preservation of dependencies.

5) Calculate the keys of the original relation.

If NO key is included in any of the decomposed sub-relation then create a new sub-relation with the attributes of key only. \Rightarrow for the loss-less join property.

6) If any of the sub-relation is a sub-set of the other, remove the smaller sub-relation.

Example:

A (PAN, PI, DI, DRUG_i, QTY, COST)

$F = \{ PAN \rightarrow PI, PI \rightarrow DI, PI \text{ DRUG}_i \rightarrow QTY,$
 $\Rightarrow DRUG_i \text{ QTY} \rightarrow COST \}$

1) ✓

2) ✓

3) ✓

4) decomposed sub-relation :

R₁ (PAN, PI) R₂ (PI \rightarrow DI)

R₃ (PI, DRUG_i, QTY)

R₄ (DRUG_i, QTY, COST), R₅ (PAN, DRUG_i)

Ex: $R(A, B, C, D, E, F, G, H, I)$

$$F = \{A \rightarrow BCD, AE \rightarrow FG, F \rightarrow AEG, G \rightarrow HI\}$$

\Rightarrow 3NF decomposition?

\Rightarrow 1) $A \rightarrow B$ $AE \rightarrow F$ $F \rightarrow A$ $C \rightarrow H$
 $A \rightarrow C$ $AE \rightarrow G$ $F \rightarrow E$ $C \rightarrow I$
 $A \rightarrow D$ \cancel{R} $F \rightarrow G$

2) $A \rightarrow B$ $AE \rightarrow F$ $F \rightarrow A$ $C \rightarrow H$
 $A \rightarrow C$ $F \rightarrow E$ $C \rightarrow I$
 $A \rightarrow D$ $F \rightarrow G$

3) ✓

4) $A \rightarrow BCD$ $F \rightarrow AEG$
 $AE \rightarrow F$ $C \rightarrow HI$

5) KEY of original :

$R_1(A, B, C, D)$, $R_2(A, E, F)$ X

$R_3(C, H, I)$, $R_4(A, E, F, G)$

$$\begin{array}{ll} A \rightarrow B & F \rightarrow A \\ A \rightarrow C & F \rightarrow E \\ A \rightarrow D & F \rightarrow G \\ AE \rightarrow F & C \rightarrow H \\ AE \rightarrow G & C \rightarrow I \end{array}$$

* B 3] No partial dependency.

$$A \rightarrow BCD, AE \rightarrow F, F \rightarrow AEG, C \rightarrow HI$$

$$\Rightarrow R_1(A, B, C, D) , R_2(A, E, F)$$

$$R_3(A, E, F, G) , R_4(C, H, I)$$

$$\Rightarrow R_1(A, B, C, D) , R_3(A, E, F, G) , R_4(C, H, I)$$

1 NF

2 NF - No partial dependency but transitive dependency .

* Multivalued dependency :

MATH	T1	B1
	T2	B2
	T3	

DSA	T11	D1
	T12	D2
		D3

u	MATH 1	T1	B1
t	MATH 1	T1	B2
v	MATH 1	T2	B1
g	MATH 1	T2	B2

$(c, t_1, x_1), (c, t_2, x_2)$ appears

then $(c, t_1, x_2), (c, t_2, x_1)$ will also appear.

$X \rightarrow\!\!\! \rightarrow Y$ holds in R

if f have 2 tuples : $t, s : t[x] = f[x]$

then you must obtain : u, v s.t. :
2 more tuples

$$1) u[x] = v[x] = t[x] = f[x]$$

$$2) u[y] = t[y] \text{ and } u[R-x-y] = f[R-x-y]$$

$$3) v[y] = f[y] \text{ and } v[R-x-y] = t[R-x-y]$$

X	Y	R-X-Y
$t[x]$	$t[y]$	$t[R-x-y]$
$f[x]$	$f[y]$	$f[R-x-y]$
$u[x]$	$u[y]$	$u[R-x-y]$
$v[x]$	$v[y]$	$v[R-x-y]$

There is a Multivalued dependency between X & Y.

$R(X, Y, Z)$

If $X \rightarrow\rightarrow Y$ then $X \rightarrow\rightarrow Z$

$$\{X \rightarrow\rightarrow Y, Y \rightarrow\rightarrow Z\} \models X \rightarrow\rightarrow (Z - Y)$$

$$\{X \rightarrow Y\} \models X \rightarrow\rightarrow Y$$

★ 4th Normal Form :

R is a relⁿ with a set of dependencies D .

R is in 4NF if there is an MDF of form
 $X \rightarrow Y$ where Y is not contained in X &

$X Y$ does not include all the attributes of R .

and then X is a key of R .

$$X \rightarrow Y$$

R, D

$$XY \quad Y \notin X$$

$R(X, Y, Z)$

X is a key of R

Eg: $R(A, B, C, D, E, F, G)$

$$D = \{A \rightarrow\rightarrow B, B \rightarrow\rightarrow C, B \rightarrow\rightarrow EF, CD \rightarrow E\}$$

\Rightarrow

KEY : ACD

$\checkmark R_7(A, F), \checkmark R_8(A, C, D)$

KEY
↓
stop here

$A \rightarrow\rightarrow B$ not in 4NF.

$\checkmark R_1(A, B), R_2(A, C, D, E, F, G)$

$CD \rightarrow E$
(Not in 4NF)

$\checkmark R_3(C, D, E), R_4(A, C, D, F, G)$

$A \rightarrow\rightarrow G$

$\checkmark R_5(A, G), R_6(A, C, D, F)$

$A \rightarrow\rightarrow F$

★ Relational Algebra

→ Operations:

- 1.) Union (U): Set of tuples t which will be either in R or in S
or in both

R ∪ S

A	B	C
a	b	c
d	a	f
c	b	f

R

D	E	F
b	g	a
d	a	f

S

No heading

a	b	c
d	a	f
c	b	f

R ∪ S

{ first write R, then see
in S what's not here,
add that }

- 2.) Set difference (-) : Set of tuples t which will be in R
but not in S.

R - S

a	b	c
c	b	d

R - S

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

- Take first tuple of R, join with all tuples of S.
 - Take next tuple.

Number of tuples : $K_1 * \underline{\underline{K_2}}$

4.) Projection (π) :

$$\pi_{G,A}(R)$$

$$\prod_{i_1, i_2, \dots, i_m} (R)$$

Integer value (col. No.)
or attribute
name

→ Taking out from each tuple, only the attribute values in (i_1, i_2, \dots, i_m)

$$\pi_{2,3}(s)$$

E F
g a
e b

5.) Selection (σ) : F is a formula \rightarrow operands \hookrightarrow

\vee Arithmetic operators
 \wedge Comparative operators
 \top Logical operators
 \Rightarrow attr names
 \Rightarrow column names etc.
 \Leftarrow constant value

\Leftarrow $\sigma_F(R)$ } from reln R we will take the tuples which will satisfy the formula F .

Eg: $\sigma_{2>3}(R)$

$$\sigma_1 = 's' \vee 2 = 'p' (R)$$

$$\sigma_B = 'b' (R)$$

A	B	C
a	b	c
c	b	d

Additional operators :

1.) Intersection (\cap) : Set of tuples which are in R as well as in S .

$$R \cap S = R - (R - S)$$

d	a	f

2.) Quotient (\div) : $R \rightarrow$ attributes &

$$S \rightarrow \text{..} S$$

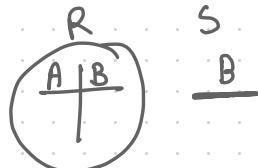
$$R^{(R)}, S^{(S)} \quad R > S, S \neq \emptyset$$

$R \div S \Rightarrow$ set of tuples with attr $(R-S)$ such that for all tuples u in S , t_u is in R

$R(A,B), S(B)$ t is tuple of $R \div S$

$$R \div S = \pi_A(R) - \pi_A(\pi_A(R) \times S - R)$$

All combⁿ
of AB



Eg: R

a	b	c	d
a	b	e	f
b	c	e	f
e \leftarrow c	d	c	d
e	d	e	f
a	b	d	e

c	d
e	f

$$\text{No. of attri : } r-s = 4-2 \\ = 2$$

<u>R \div S</u>	
a	b
e	d

If c replaced by e,

Now ed can also come.

b c won't come
because

but bcef is there
but bcod is not there

3.) Θ -join :

$R \bowtie_{i \theta j} S$

$\hookrightarrow i$ is in some
relation with j .

$\sigma_{(i \theta (r+j))} R \times S$

<u>R</u>		
A	B	C
1	2	3
4	5	6
7	8	9

$R \bowtie_{B < D} S \quad \} \text{ No. of columns} = 5$

(3+2)

<u>S</u>	
D	E
3	1
6	2

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

4.) Natural Join :

1.) $R \times S$

2.) for each attri. A which appears in both R & S , select those tuples from $\underline{R \times S}$ where,

$$R \cdot A = S \cdot A$$

3.) Remove $S \cdot A$

<u>R</u>		
A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

<u>S</u>		
B	C	D
b	c	d
b	c	e
a	d	b

→ 2 attri same name.

I have to remove.

∴ Resultant table will have 4 columns. A,B,C,D

$R \times S$

A	$R \cdot B$	$R \cdot C$	$S \cdot B$	$S \cdot C$	D
	✓	✗	✓	✗	

A	B	C	D
a	b	c	d
a	b	c	e
d	b	c	d
d	b	c	e
c	a	d	b

Ex: Employee (Emp. id. , Name, Salary, M-id)

↓

= 1 for Manager
= 0 otherwise

→ find out name of the employee whose salary is more than any manager (may not be All).

⇒ $\pi_2 [\sigma_{4 \neq '1' \wedge 8 = '1'} \wedge 3 > 7 (\text{Employee} \times \text{Employee})]$

Ex: Customer (c.name, street, c.city)

Deposit (B.name, ac.no., c.name, balance)

Borrow (B.name, loan no., c.name, amount)

Branch (B.name, B.city, asset)

* = KEYS

Client (c.name, emp.name)

(1) → find the c.names such that c.name & the emp. who handles that customer, his name is same.

$\Pi_1(\sigma_{1=2} \text{ Client})$

(2) → find all clients of emp. 'xyz' at corresponding cities of those clients.

$\Pi_{1,5}(\sigma_{2='xyz'} \text{ Client } \times \text{ Customer})$

If this condition was not there,
we could have used Natural Join.

(3) → find all customers who have both account and loan at 'KGP' branch.

$\Pi_3(\sigma_{3=7 \wedge 1='KGP'} \text{ Borrow } \times \text{ Deposit})$

OR

$3=7 \wedge 1='KGP'$
 $5='KGP'$

Deposit \times Borrow

OR (Natural Join)

$\Pi_3 (\sigma_{1='KOLP'} (\text{Borrow} \bowtie \text{Deposit}))$

(4) find all customers who have accounts in all the branches
in the city 'KOLKATA'.

$\Pi_{C.\text{name}, B.\text{name}} (\text{Deposit})$

$\div \Pi_{B.\text{name}} (\sigma_{B.\text{city} = 'KOL'} (\text{Branch}))$

All customers & their branch where they have account
All the Branches in Kolkata.

Eg: Lives ($p.\text{name}$, street, city)
person

Works ($p.\text{name}$, *company*, $c.\text{name}$, salary) -

Do of your own.

Located in ($c.\text{name}$, $c.\text{city}$)

Manages ($p.\text{name}$, $M.\text{name}$)
Manager

- (1) find the name & city for all employees who work for 'FCI' company.
- (2) find the name, street & city of all employees who work for 'FCI' comp. & whose salary is more than 50,000.
- (3) find all employees who live in the same city & the Company they work for.
- (4) find all employees who live in the same city & same street as their manager.
- (5) find all employees whose salary is more than every employee of 'XYZ' company.

Try :

$$\underline{\text{Soln}} : (1) \quad \Pi_{1,3} \left(\text{lives} \bowtie \left(\Pi_{p.name} \left(\sigma_{c.name = 'FCI'} (\text{works}) \right) \right) \right)$$

$$(2) \quad \Pi_{1,3} \left(\text{lives} \bowtie \left(\Pi_{p.name} \left(\sigma_{c.name = 'FCI'} (\text{works}) \right) \right) \right) \\ \text{Salary } \overset{\delta}{>} 50000$$

$$(3) \quad \Pi_1 \left(\sigma_{3=8} (\text{lives} \bowtie \text{works} \bowtie \text{located in}) \right)$$

$$(4) \quad P = \text{lives} \bowtie \text{Manages} \quad \left\{ \begin{array}{l} p.name, street, city, m.name \\ \hline \end{array} \right.$$

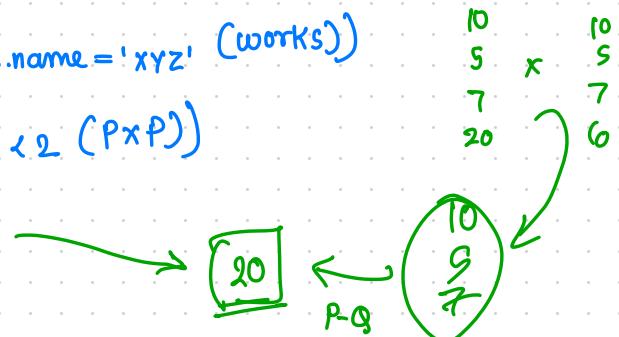
$$\Pi_1 \left(\sigma_{1=7 \wedge 2=4 \wedge 3=5 \wedge 1 \neq 4} (\text{lives} \times P) \right)$$

$$(5) \quad P = \Pi_3 \left(\sigma_{c.name = 'xyz'} (\text{works}) \right)$$

$$Q = \Pi_1 \left(\sigma_{1 < 2} (P \times P) \right)$$

$$R = P - Q$$

$$(\text{works} \times R)$$



→ we can solve any query except sorting, etc.

Tuple Calculus

* Propositional Logic

\neg (not), \wedge (and), \vee (or),

\Rightarrow (implication or If / then)

\equiv (equivalence or iff)

* Proposition:

A declarative statement which will be either true or false is called a proposition.

→ symbols used to represent prop. is called atom(s).

* Tuple Relational Calculus : → takes this form $\{t \mid \psi(t)\}$ where

→ t is a tuple variable of some fixed length.

δ $\psi(t)$ is a formula built from the atoms.

$R(\delta)$, $\delta \in R$

$\delta[i] \theta u[j]$

⇒ j^{th} attribute of tuple δ is in relation θ with j^{th} attri of tuple u .

- Existential quantifier (\exists)

→ There exist atleast one tuple x for which this formula will be true. $(\exists x) \Psi(x)$

- Universal quantifier (\forall)

→

$$\begin{array}{l} (\forall x) \Psi(x) \\ (\exists s) (R(s)) \end{array} \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{Formula}$$

There exist atleast one tuple s which is a tuple of the relation A.

$$(\forall s) (\Psi(s)) \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{Formula}$$

Any tuple s you take, it will be in Relation R.

$\rightsquigarrow R \cup S : \{ t \mid R(t) \vee S(t) \}$

$R - S : \{ t \mid R(t) \wedge \neg S(t) \}$

$\rightarrow \text{I have to use } 3 \text{ tuple variables}$

$R^{(r)} \times S^{(s)} : \{ t^{(r+s)} \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge$

$\begin{aligned} & t[1] = u[1] \wedge t[2] = u[2] \wedge \dots \wedge t[r] = u[r] \wedge \\ & t[r+1] = v[1] \wedge \dots \wedge t[r+s] = v[s] \end{aligned} \}) \}$

$\swarrow u \text{ belongs to } R$

$\searrow v \text{ belongs to } S$

$\rightarrow \text{I have to use } 2 \text{ tuple variables.}$

$\Pi_{i_1, i_2, \dots, i_k}(R) : \{ t^{(k)} \mid (\exists u)(R(u) \wedge t[1] = u[i_1] \wedge \dots \wedge t[k] = u[i_k]) \}$

$$\sigma_F(R) : \{t \mid R(t) \wedge F'\}$$

You are selecting tuples
s.t. it will be a member
of R only, so we don't need
2 tuple var, but only $t \rightarrow t$

denoting as F' bcz F & F'
will not be same, one
is Relational algebra one
is tuple calculus.

$$\text{Eg: } \Rightarrow \pi_{1,4} (\sigma_{2=3} (R^{(2)} \times S^{(2)})) :$$

$$\{t^{(2)} \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge$$

$\star F' \quad 2=3 \quad \pi_1 \quad \pi_4$

$$u[2] = v[1] \wedge t[1] = u[1] \wedge t[2] = v[2]\}$$

$$\Rightarrow R^{(2)} \bowtie S^{(2)} :$$

$$\{t^{(2)} \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge$$

$$u[2] = v[1] \wedge t[1] = u[1] \wedge t[2] = v[2]\}$$

Q. Find the branch name, loan No., customer name & amount
for loans over ₹ 10,000.

Customer (c.name, street, c.city) \vee

Deposit (b.name, ac.no., c.name, balance)

Borrow (b.name, loan no., c.name, amount) \vee

Branch (b.name*, b.city, asset)

Client (c.name*, emp.name)

\Rightarrow Borrow ✓

$\{t \mid t \in \text{Borrow} \wedge t[4] > 10000\}$

Q. Find all customers who have loan more than ₹ 10,000.

$\Rightarrow \{t^{(1)} \mid (\exists u) (u \in \text{Borrow} \wedge u[4] > 10000 \wedge t[1] = u[3])\}$

↳ have to use separate tuple variable

Q. Find all customers who have loans from "IIT" branch and the cities where they live.

$\Rightarrow \{t^{(0)} \mid (\exists u)(\exists v) (u \in \text{Customer} \wedge v \in \text{Borrow} \wedge$

$u[1] = v[3] \wedge v[1] = \text{"IIT"} \wedge$

$t[1] = u[1] \wedge t[2] = u[3]\}$

Without Cartesian Product

$\hookrightarrow \{t \mid (\exists u)(u \in \text{Borrow} \wedge u[1] = \text{'IIT'} \wedge t[1] = u[3]$

$\wedge (\exists v) (v \in \text{Customer} \wedge u[3] = v[1] \wedge t[2] = v[3])\}$

$\hookrightarrow \{t \mid (\exists u)(\exists v) (u \in \text{Borrow} \wedge v \in \text{Customer} \wedge u[1] = \text{'IIT'}$

with cartesian
product

$\wedge u[3] = v[1] \wedge t[1] = u[3] \wedge t[2] = v[3]\}$

Q. Find all customers having a loan or Account or Both at "IIT" branch.

\Rightarrow Deposit (B.name, ac.No., C.name, balance)

Borrow (B.name, loan no., C.name, amount)

$\Rightarrow \{t^{(0)} \mid (\exists u) (u \in \text{Deposit} \wedge u[1] = \text{'IIT'} \wedge t[1] = u[3])$

logical OR $\hookrightarrow \{(\exists v) (v \in \text{Borrow} \wedge v[1] = \text{'IIT'} \wedge t[1] = v[3])\}$

Instead :

Logical AND : have both account & loan at 'IIT' branch

AND NOT : have account but no loan at 'IIT' branch.

Q. Find all customers who have account at all branches in the city 'KGP'. \div

\Rightarrow Customer (c.name, street, c.city)

Deposit (d.name ac.no., c.name, balance) \checkmark

$\{ t \mid (\forall u)(u \in \text{Branch} \wedge u[2] = 'KGP') \Rightarrow$

$(\exists v)(v \in \text{Deposit} \wedge u[1] = v[1] \wedge t[1] = v[3]) \}$

If P then Q.

$$P \Rightarrow Q$$

$\neg P \vee Q$

$\rightarrow \{ t \mid (\exists u)(u \notin \text{Branch} \vee u[2] \neq 'KGP')$

$\vee (\exists v)(v \in \text{Deposit} \wedge u[1] = v[1] \wedge t[1] = v[3]) \}$

Till here : Syllabus for Midsem.

SQL :

Select A₁, A₂, ..., A_n
 from R₁, R₂, ..., R_n
 where P ;

correspond relational algebra

$$\pi_{A_1, A_2, \dots, A_n} (\sigma_P (R_1 \times R_2 \times \dots \times R_m))$$

→ Union intersect minus
 and or not

Q. Find the name of all branches in deposit relation.

⇒ → Select distinct b.name
 from Deposit ;

Q. find all customers who have account at IIT branch.

⇒ Select C.name
 from Deposit
 where b.name = 'IIT' ;

Q. find all customers having a loan, an account or both
 at IIT branch.

⇒ (Select c.name
 from Deposit
 where b.name = 'IIT')
 Union

```
( select c.name  
from Borrow  
where b.name = 'IIT');
```

Q. find all customers having a loan at any branch & their city.

→ Customer (c.name, street, c.city)
Deposit (B.name, ac. No., c.name, balance)
Borrow (B.name, loan no., c.name, amount)
Branch (B.name, B.city, Asset)
Client (c.name, emp. name)

Select c.c-name, c.city
from Customer C, Branch B
where C.c-name = B.c-name ;
(and b.name = 'IIT') → for only IIT branch

Select
from
where

Q. Find all customers who have both loan and account at IIT branch.

→ Select c.name
from Deposit
where b.name = 'IIT'

and c.name in



(Select c.name
from Borrow
where b.name = 'IIT')

Q. find all customers who have account at IIT branch but no loan from IIT branch.

⇒ Select c.name
from Deposit
where b.name = 'IIT'
and c.name not in
(Select c.name
from Borrow
where b.name = 'IIT')

Q. find all branches that have more assets than any branch located in KGP.

⇒ Select T.b.name
from Branch T, Branch S
where T.asset > S.asset
and S.b.city = 'KGP'

Select b.name
from Branch
where asset > any

> any
< any
<> any

(Select asset
from Branch
where b.city = 'KGP')

> all
< all

Q. find all customers with balance lies between 50000 & 60000.

⇒ Select c.name
from Deposit
where balance \geq 50000 and balance \leq 60000

Customer (c.name, street, c.city)

Deposit (B.name, ac.no., c.name, balance)

Borrow (B.name, loan no., c.name, amount)

Branch (B.name, B.city, Asset)

Client (c.name, emp.name)

Select c.name

from Deposit

where balance between 50000 and 60000

not between ✓

Q. Find in alphabetical order all the customers who have loan in IIT branch.

⇒ Select c.name

from Borrow

where b.name = 'IIT'

order by c.name

for descending order
order by _____ desc;

avg, min, max, sum, count

Q. find the average account balance at all branches.

⇒ Select b.name, avg(balance)

from Deposit

group by b.name

having avg(balance) > 10000

if we want only avg.
balance > 10000

Q. find no.of tuples in customer relation .

⇒ Select count(*)

from Customer

Q. find all customers who have deposit at IIT branch but
for whom there is no entry in customer relation.

⇒ Select c.name

from Deposit D

where b.name = 'IIT' and c.name not in

not exists

(Select c.name
from Customer)

(select *
from Customer C

where D.c.name = C.c.name)

⇒ After midsem , one problem → compulsory to
submit ✓

 Teacher (T.name, Dept., Tl.No., ^{Telephone} S.title)
Student (S.name, course, hall)
Study (S.title, ^{Subject name} S.name, level, status, marks)

Q. find the name of teachers of Math dept. who teach 200 Level subjects.

Q. find students of CSDP course living in VS or SN hall who study NO subject taught by Prof. XYZ.

⇒ Select S.name
from student
where course = 'CSDP' and
(hall = 'VS' or hall = 'SN')

and S.name not in
(Select S.name
from Teacher T, Study S
where T.name = 'XYZ'
and T.S.title = S.S.title)

Q. find the avg marks in the subjects taught by Prof. XYZ .

⇒ Select S.S.title , avg(marks)

from Study S, Teacher T

where T.s.title = S.S.title

and t.name = 'XYZ'

group by S.S.title

Q. find the name of the subjects such that all students studying the subject secure more than 80 marks.

⇒ Teacher (T.name, Dept., Tel.No., S.title)

Student (S.name, course, hall)

✓ Study (S.title, S.name, level, status, marks)

Select S.title

from Study

group by S.title

having min(marks) > 80

Q. find the name of students who secure equal marks in 2 different subjects.

⇒ Select S.name

from Student

where S.name in (select S.S.name

from Study S, Study T

where S.S.name = T.S.name

and S.S.title ≠ T.S.title

and S.marks = T.marks.

} you can
also write
this ONLY

Q. Find the number of students in each subject taught by Prof. XYZ.

⇒ Teacher (T.name, Dept., Tel.No., S.title)

Student (S.name, course, hall)

Study (S.title, S.name, level, status, marks)

Select S.s.title, count(S.name)

from Study S, Teacher T

where T.name = 'XYZ' and

S.s.title = T.s.title

group by S.s.title

⇒ R(A,B,C) | Is it in 3NF ?

F = {A → C, B → C}

⇒ 3NF decomposition

KEY : AB

R₁(A, C), R₂(B, C), R₃(A, B)

Yes

BCNF decomposition :

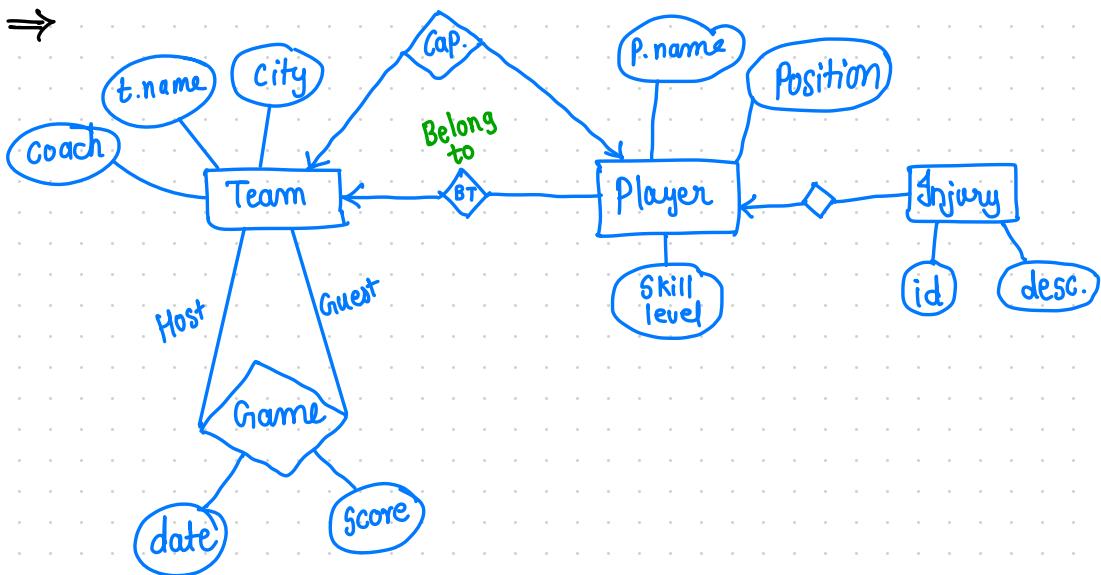
R₁(A, C), R₂(A, B)

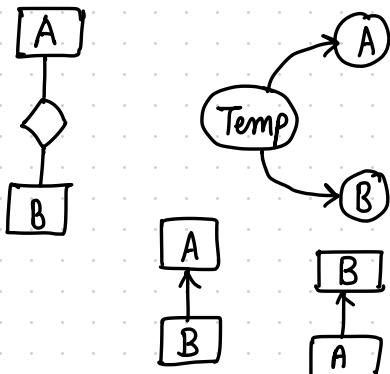
No

→ In SQL we can use nested queries.

- Q. Suppose you are given the following requirement for a database for IPL :
- 1) IPL has many teams
 - 2) Each team has a name city, coach, captain and a set of players .
 - 3) Each player belongs to only 1 team
 - 4) Each player has a name, a position , skill level & the set of injury records
 - 5) A team captain is also a player.
 - 6) A game is played between 2 teams referred as host team and guest team & has a date and a score.

Draw the E-R diagram ?





★ QWEL

format : range of t_1 is R_1 ,
range of t_2 is R_2 ,
:
range of t_n is R_n
retrieve $(t_{i1} \cdot A_{j1}, t_{i2} \cdot A_{j2}, \dots, t_{in} \cdot A_{jn})$
where P ;
 \sqsubset condition

↑ tuples ↑ Relations ↑ Attributes of t_i 's

Relational Algebra :

$$\{ \mid (\exists t_1)(\exists t_2)(\exists t_3)$$

Q. find all customers having an account at IIT branch:

Customer (c.name, street, c.city)

Deposit (B.name, ac.No., c.name, balance)

Borrow (B.name, loan no., c.name, amount)

Branch (B.name, B.city, Asset)

Client (c.name, emp.name)

⇒ range of t is Deposit

retrieve (t.c.name)

where t.b.name = 'IIT'

Q. find all customers having a loan at IIT branch & their cities.

Customer (c.name, street, c.city)

⇒ Deposit (B.name, ac.No., c.name, balance)

Borrow (B.name, loan no., c.name, amount)

Branch (B.name, B.city, Asset)

Client (c.name, emp.name)

⇒ range of t is Borrow

range of u is Customer

retrieve (t.c.name, u.c.city)

where t.c.name = u.c.name

and

t.b.name = 'IIT'

Q. Find all customers who have loan & account at IIT branch.

Customer (c.name, street, c.city)
Deposit (b.name, ac.no., c.name, balance)
Borrow (b.name, loan no., c.name, amount)
Branch (b.name, b.city, Asset)
Client (c.name, emp.name)

⇒ range of t is Borrow

range of u is Deposit

retrieve (u.c.name)

where t.c.name = u.c.name

and

t.b.name = 'IIT'

and

u.b.name = 'IIT'

Q. find all customers who have account or loan or both at IIT branch. (Union operation)

Customer (c.name, street, c.city)
Deposit (b.name, ac.no., c.name, balance)
Borrow (b.name, loan no., c.name, amount)
Branch (b.name, b.city, Asset)
Client (c.name, emp.name)

⇒ Here, we can't use nested queries like SQL, it's difficult.

Creating new file

range of u is Deposit

retrieve into temp ($u.c.name$)
where $u.b.name = 'IIT'$

Accounts at IIT

(Appending)

range of s is Borrow

append to temp ($s.c.name$)
where $s.b.name = 'IIT'$



loans at IIT

Storing in existing file
will create if not exist

range of t is temp

retrieve unique ($t.c.name$)
 II2
distinct → In SQL

Taking Unique

Q. Find all customers who have account at IIT branch but no loan at IIT branch.

⇒ Same

range of u is Deposit
retrieve into temp ($u.c.name$)
where $u.b.name = 'IIT'$

Accounts at IIT

range of s is Borrow

range of t is temp

delete (t)

where $t.c.name = s.c.name$

and

$s.b.name = 'IIT'$



Deleting those who have loans at IIT

range of t is temp
retrieve unique ($t.c.name$)

Q. Find all customer names in alphabetical order who have accounts at IIT branch.

Customer (c.name, street, c.city)

Deposit (b.name, ac.no., c.name, balance)

Borrow (b.name, loan no., c.name, amount)

Branch (b.name, b.city, Asset)

Client (c.name, emp.name)

⇒ range of u is Deposit

retrieve u.c.name

where u.b.name = 'IIT'

Sort by u.c.name

max sum
min count
avg

Q. Find the average account balance for all accounts at IIT branch.

→ Condition will come inside brackets.

⇒ range of t is Deposit

retrieve avg(t.balance where

t.b.name = 'IIT')

Q. Find all account numbers whose balance is more than the average balance at the branch where the account is held.

Deposit (B.name, ac.No., C.name, balance)

→ range of t_1 is Deposit

range of t_2 is Deposit

retrieve where

long method



range of t is Deposit

retrieve ($t.ac.no.$)

where $t.balance > \text{avg}(t.balance \text{ by } t.name)$

Q. find all customers who have account at all branches in the city "KGP".

→ Customer (C.name, street, C.city) ?

\checkmark t_1 ✓ Deposit (B.name, ^{all such} ac.No., ^{for all} C.name, balance)

Borrow (B.name, loan no., C.name, amount)

\checkmark ✓ Branch (B.name, B.city, Asset)

Client (C.name, emp.name)

⇒ range of t is Deposit

range of u is Branch

range of s is Deposit

retrieve (t.c.name)

where

countu (s.b.name where

s.b.name = u.b.name and

u.b.city = 'KorP' and

g.c.name = t.c.name)

Tomorrow : student teacher
δ File organization. → not in books

≠ Attend classes → different than books

= countu (u.b.name

where u.b.city = 'KorP')

Teacher (T.name, Dept., Tl.No., S.title)

28th Feb 2023

Student (S.name, course, hall)

Study (S.title, S.name, level, status, marks)

Q. Find the name of subjects such that all students studying the subject score more than 80 marks.

⇒ range of t is Study

retrieve (t.title)

where min(t.marks by t.title) > 80

Q. find the students of CSDP course living in 'VS' or in 'SN' hall, who study no subject taught by prof. 'XYZ'.

→ range of t is student
append to temp1 (t.sname)
where (t.course = 'CSDP' and
(t.hall = 'VS' or t.hall = 'SN'))

range of u is teacher

range of s is study

append to temp2 (s.sname)

where u.stitle = s.stitle

and u.tname = 'XYZ'

range of t₁ is temp1

range of t₂ is temp2

delete (t₁)

where t₁.sname = t₂.sname

range of t₃ is temp1

retrieve unique (t₃.sname)

.

Q. find the maximum marks secured in each subject of the CSDP course.

→ Teacher (T.name, Dept., Tl.No., S.title)

Student (S.name, course, hall)

Study (S.title, S.name, level, status, marks)

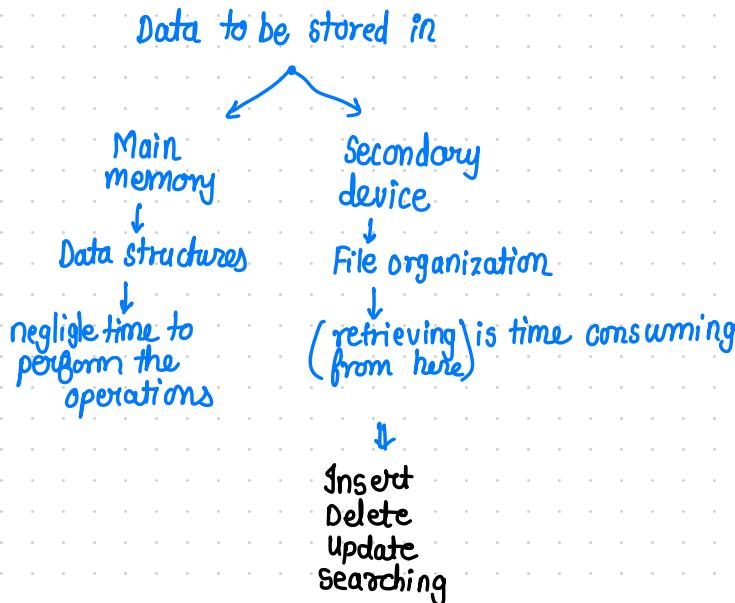
\Rightarrow range of t is study
range of u is student
retrieve (t.stitle, max(t.marks by t.stitle))
not asked but we are giving
where (t.sname = u.sname
and u.course = 'CSDP')

Q. find the name of students who secured equal marks in two different subjects.

\Rightarrow range of t is study
range of u is study
retrieve (t.sname)
where (t.sname = u.sname
and t.stitle \neq u.stitle
and t.marks = u.marks)

File Organization

→ How to store your data in secondary device s.t. retrieving time will be minimum.



For which type of operation, I should store my data in which fashion.



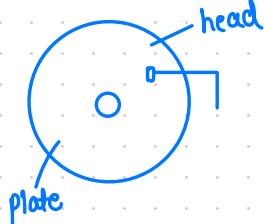
1 Block = 2400 bytes (varies from machine to machine)

Block

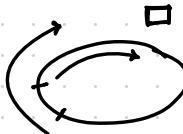
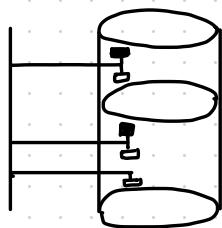
lowest unit by which we transfer data

or 4800 bytes

Gramophone = sequence of tracks give you a cylinder.



head has information, on which Block it has to perform operation.



Seek time = time it takes to move the arm to the correct cylinder.

Disk latency time (Rotation latency time) =

Disk latency is the time that it takes to complete a single I/O operation on a block device.

Average seek time (s) = time it takes to traverse $\frac{1}{3}$ rd of the cylinder.

Average rotational latency time (r) : avg time it takes to locate the correct block on a track once head is on correct track.

$$= \frac{1}{2} \times \text{revolution time}$$

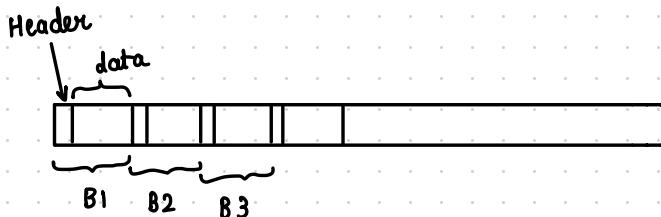
Block transfer time (btt) : the time it takes to transfer the information of one block from disk to memory once the head is correctly positioned at the beginning of a block.

$$= s + r + btt$$

(Size of block)
B bytes

(Data transferred at the rate of)
t bytes/ms

$$Btt = \frac{B}{t} \text{ ms}$$



Effective block transfer time (ebt) : Time it takes to transfer the information of one block along with header information.

$$ebt = \frac{B}{t_1}$$

ebt will be greater

* Parameters for IBM 3380

B 2400 bytes

btt 0.8 ms

C 600 (blocks per cylinder)

ebt 0.84 ms

N 885 (No. of cylinders)

τ 8.3 ms

δ 16 ms

t 3000 bytes/ms

t' 2857 bytes/ms

* Read 10 adjacent blocks (same for write)

$$\begin{aligned}\text{time} &= \$ + r + 10 \times ebt \rightarrow btt + 9 \times ebt \\ &= 32.7 \text{ ms}\end{aligned}$$

* Random access of 10 blocks

$$\begin{aligned}\text{time} &= 10 (\$ + r + btt) \\ &= 251 \text{ ms}\end{aligned}$$

* Sequential access time $\approx b \times ebt$ (when value of b is very large)

$$\text{random access time} = b \times (\$ + r + btt)$$

* Heap File Organization

T_F = time to fetch a record

T_N = time to fetch next record

T_I = time to insert new record

T_U = time to update a record

T_D = time to delete a record

T_x = time to exhaustive reading of a file

T_y = time to reorganize a file

b blocks

$$T_F = \frac{b}{2} \times ebt$$

* 100000 records of 400 bytes each

$$\Rightarrow b = \frac{100000 \times 400}{2400} = \frac{100000}{6} = \underline{\underline{16,667}} \text{ (no. of blocks)}$$

$\therefore T_F \approx 7 \text{ sec.}$ → avg. time taken to access any 1 block.
not considering time (s+3s)

* fetch 10000 records (in hospital)

time = $\approx \underline{\underline{19 \text{ hours}}}$ → Not suitable



05 does not read the header part of 1 block $\Rightarrow \underline{\underline{btt}}$

when you access continuous blocks, you are forced to

read header also = ebt

ebt > btt

T_N = time to fetch next record \rightarrow new address \rightarrow it can be anywhere
so I have to start from beginning.

$$= T_F$$

T_I = time to insert new record

\rightarrow Assuming I have enough space in last block to insert new record.

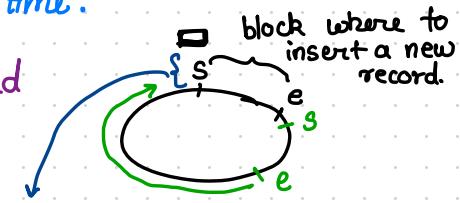
\rightarrow Assuming, I have address of last & first block.

\rightarrow Transfer to main memory, use data structures, bring back

after information is modified & write it.

→ Main memory takes negligible time.

s & e can be interchanged



$$(s + \sigma) + btt \text{ (read)}$$

$$+ (2\tau - btt) \quad (\text{one full revolution}) \\ + btt \quad (\text{write}) \qquad \qquad = 2\tau$$

$$T_I \Rightarrow s + \sigma + btt + 2\tau$$

T_D = time to delete a record

→ we don't delete physically, we just put a deletion beat.

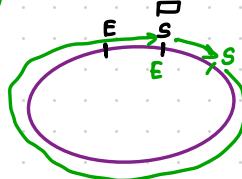
→ search the block

+ put deletion beat in main memory



$$T_D = T_F + 2\tau_{up}$$

includes writing time.



T_x = time to exhaustive reading of a file

$$T_x = b \times ebt$$

$$\underline{\text{e.g.}} \quad \frac{100000 \times 400}{2400} = b =$$

$$b \times ebt = \underline{14 \text{ sec.}}$$

To access 100000 records

→ This HFO is most suitable for sequential accessing of

data from beginning to end.

→ Not suitable for individual fetching.

T_x (independent fetch of all records) $\rightarrow n = \text{no. of records}$

$$= n \times T_F$$

$$= n \times \frac{b}{2} \times ebt = \underline{\underline{190}} \text{ hours} \rightarrow \underline{\underline{8}} \text{ days}$$

→ Random accessing is not possible with this organization.

T_y = time to reorganize a file

→ Rewrite entire file again by removing the deleted records.
write only active records.

→ we do it periodically (15 d or 1 month)

→ Access all the blocks, then rewrite active records only.

$$T_y = b \times ebt + \frac{n}{Bfr} \times ebt$$

No. of
records
per block

→ no. of blocks required
to store the active
record.

* Bucket :

→ logical unit of data structure in file organization.

→ Buffer = a portion of the memory kept aside for
doing some specific jobs. Physically it doesn't exist, only
logically you are visualizing.

1 Bucket = n blocks, $n = 1, 2, 3, \dots$

→ It will access data Bucket wise, not block wise.

Eg: 1 bucket = 2 blocks.

$$\text{time} = (\delta + \tau + btt) + (ebt)$$

$$\Rightarrow \text{Random time} = \underbrace{bk(\delta + \tau + dtt)}_{\substack{\text{no. of buckets} \\ \rightarrow \text{accessing randomly} \\ \text{required}}} + \overbrace{btt + ebt}^{(n-1)^*}$$

dtt = data transfer time

→ If Bucket size increases → \underline{bk} will decrease

$$\text{Random time} = \underbrace{bk(\delta + \tau)}_{\substack{\text{Reduce} \\ \downarrow \text{Reduce}}} + \underbrace{bk \times ddt}_{\text{constant}} \uparrow$$

For sequential case :

Eg: $bk = 1$ block = 2400 bytes

$$dtt = 0.8 \quad (=btt)$$

bucket size = 2 blocks

$$\begin{aligned} dtt &= btt + 1 ebt = 0.8 + 0.84 \\ &= \underline{\underline{1.64}} \end{aligned}$$

Bucket size = 1 $bk = 1250$

$$\Rightarrow 1250 \times 0.84 \approx \cancel{1050} \quad 10 \text{ } \textcircled{D} \quad +s+r$$

Bucket size = 2 $bk = 625$

$$\Rightarrow 625 \times 1.64 \approx 1025 \quad +s+r$$

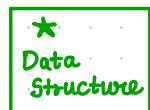
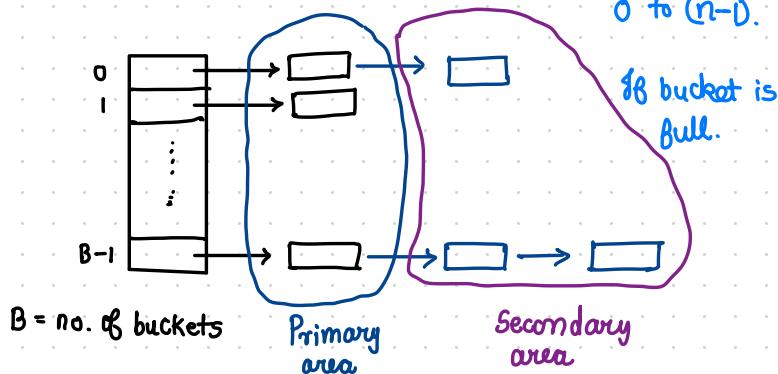
→ Not useful when we want to access sequentially. It is useful for Random access → it reduces the time.

Hash File Organization

→ I will store the records depending upon Key value.

no. of blocks required = n

create a hash function \rightarrow will give me integer value between 0 to $(n-1)$.



22, 35, 33, 42, 71, 60, 47, 19, 46, 17

$$h(x) = x \bmod 5 \rightarrow \text{no. of buckets}$$

No. of buckets = 5

$$\text{No. of records per bucket} = 3$$

Load factor

$$L_f = \frac{n}{\text{records}}$$

no. of buckets
in primary area

$$\frac{M \times Bkfr}{\text{No. of records}}$$

No. of records
per bucket

0	35	60					
1	71	46					
2	92	42	47	→	17		
3	33						
4	19						

$$\frac{10}{5 \times 3} = \frac{2}{3} = \underline{\underline{0.6}}$$

→ If it is less than 1 then we can say that overflow area will be less.

$$T_F(\text{no chain}) = \delta + \gamma + dtt$$

$$T_F(\text{successful}) = (\delta + \gamma + dtt) + \frac{x}{2}(\delta + \gamma + dtt)$$

To access first bucket in the primary area

I have x buckets in sec. area, on average I will access $\frac{x}{2}$ buckets.

$$T_F(\text{Unsuccessful}) = (\delta + \gamma + dtt) + x(\delta + \gamma + dtt)$$

$$T_I = \underline{T_F(\text{unsuccessful})} + 2\gamma \text{ (seen in last class)}$$

The information of that block/bucket is transferred to main memory

$$T_D = T_F(\text{successful}) + 2\gamma$$

$$(T_U) = T_D : \text{If key value is not changed.}$$

(Assuming key value is not changed)

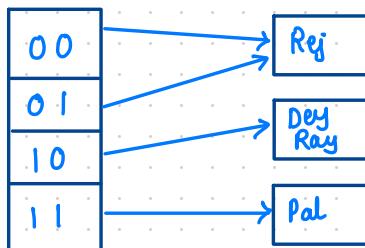
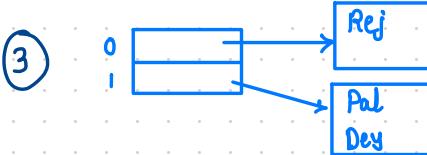
$$(T_U) = T_D + T_I : \text{If key value is changed.}$$

$$T_x = \text{time to exhaustive reading of a file} \\ = n \times \underline{\underline{T_F}} ?$$

Extendible hashing

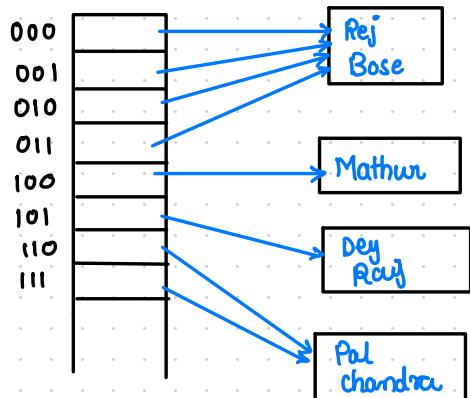
	Name	$h(\text{Name})$
⑤	Bose	0010 1101
⑥	Chandra	1101 0101
③	Dey	1010 0011
⑦	Mathur	1000 0111
①	Pal	1111 0001
④	Ray	1011 0101
②	Rej	0101 1000

create a bucket



Bucket size = 3

bits =



Next Monday
→ Pavan Kumar
B+

* B+ Tree :

Roll No., Grade } of a student

Stored at the leaf of a tree.

→ Assume 2 or 3 records in a leaf.

Record : Roll No. Grade

39 → C

etc.

→ 2 or 3 pointer in a non-leaf

→ Non-leaf store Key values

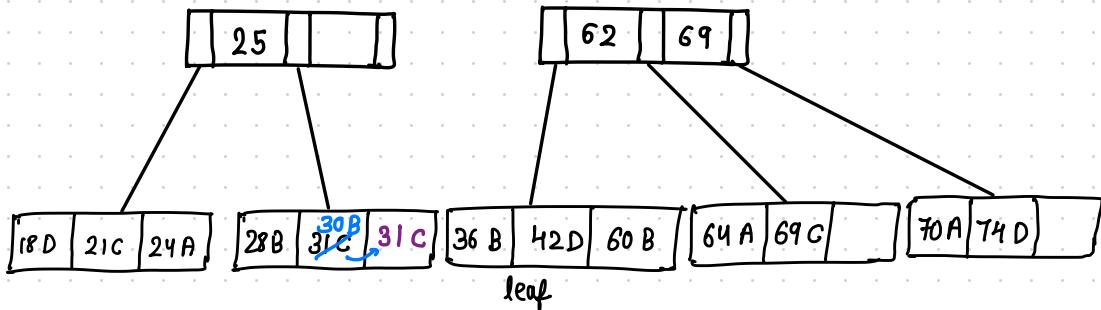
→ Policy can be strictly less than or less than equal to, anything.

B+ Tree

Roll No., Grade

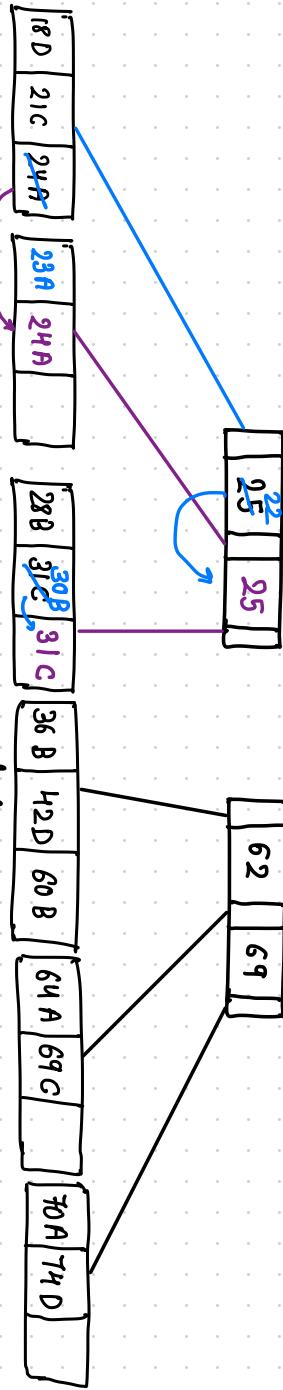
2/3 record in a leaf

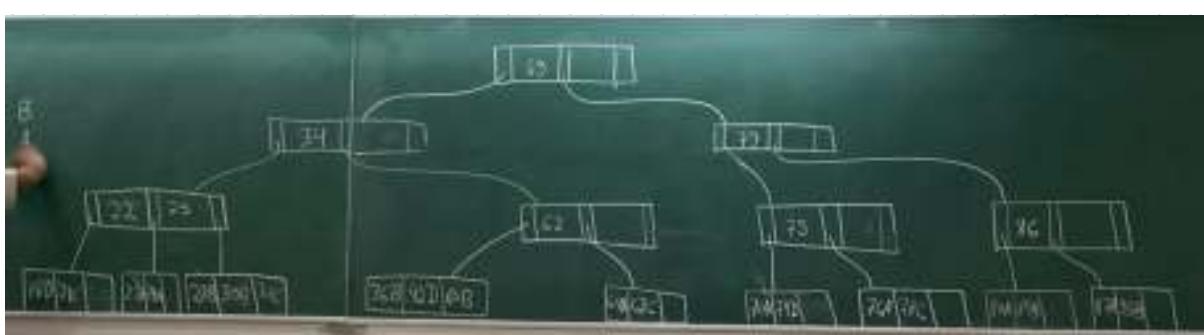
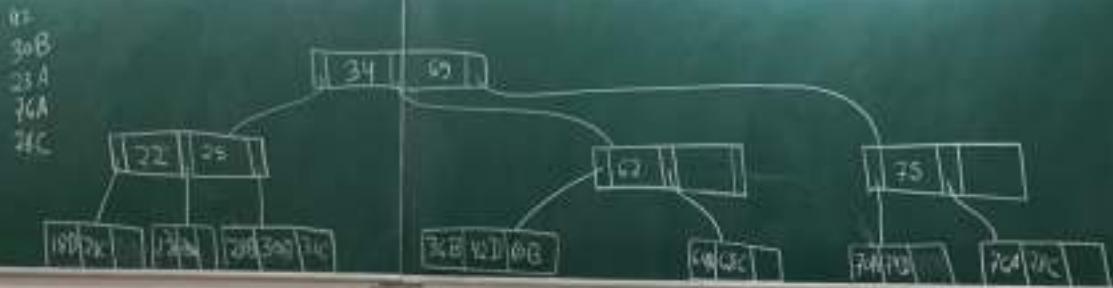
2/3 pointer in Non-leaf



Insert 30 B

insert-
30 B / 76 A
23 A





Eq:

14 A

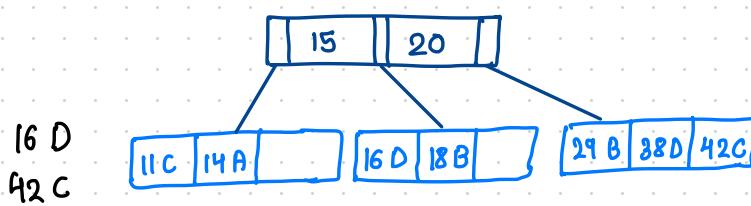
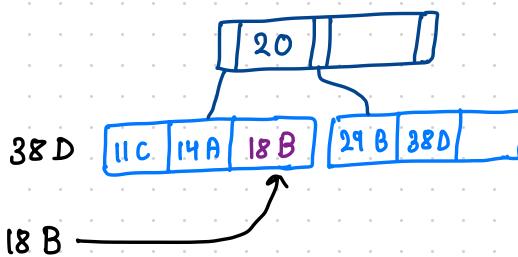
14 A		
------	--	--

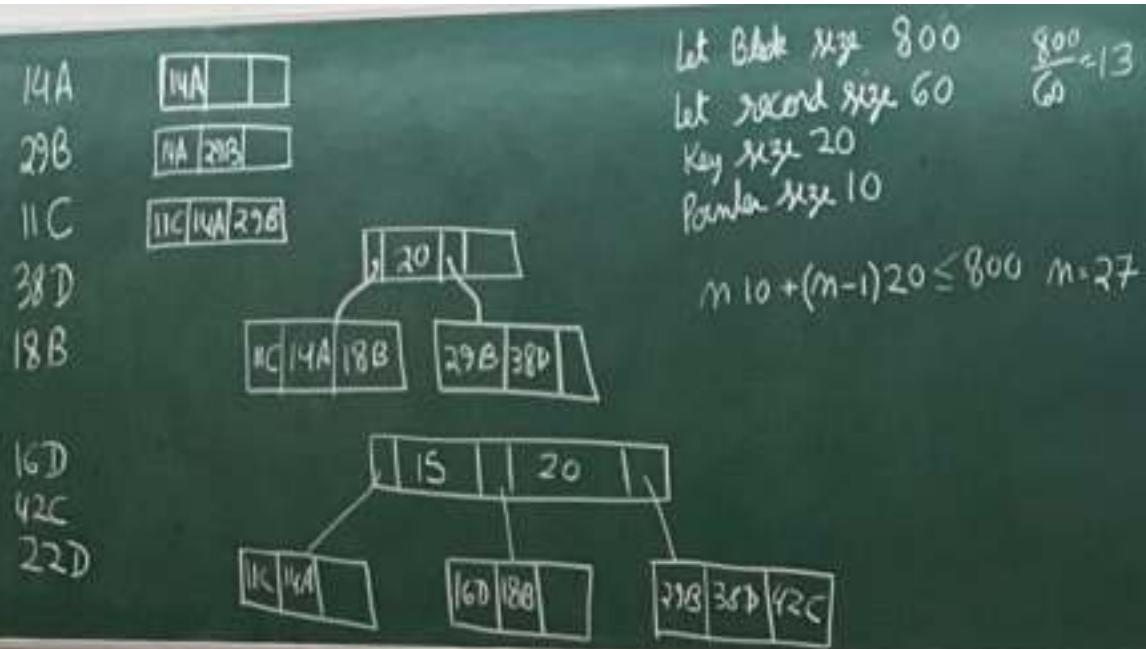
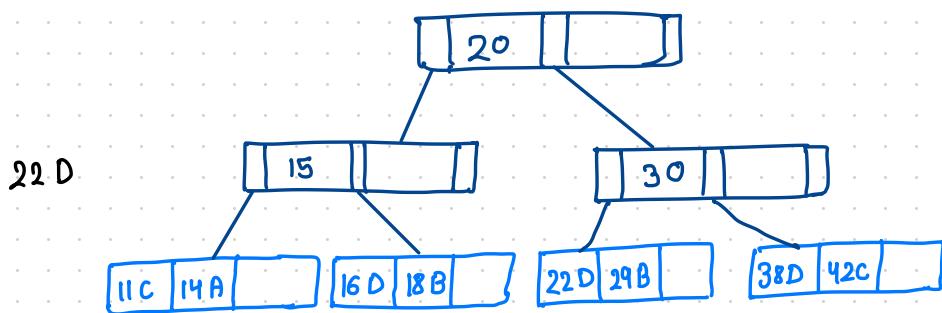
29 B

14 A	29 B
------	------

11 C

11 C	14 A	29 B
------	------	------





Let Block size 800

record size 60

$$\frac{800}{60} = 13$$

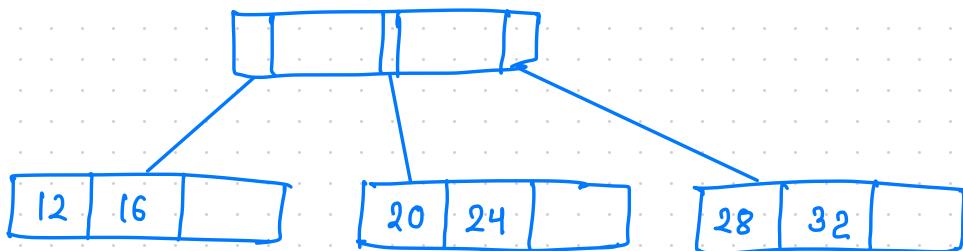
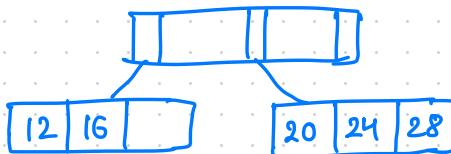
Key size 20

Pointer size 10

$$n \times 10 + (n-1)20 \leq 800 \quad (n=27)$$

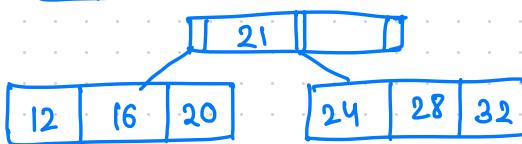


Eg: Let 12, 16, 20, 24, 28, 32, 36, 40 are inserted



Eg: Insert : 12, 16, 24, 28, 32, 20

Orders in which records are inserted, storage changes.



2/3 record in a leaf except root

3/4/5 pointer in non-leaf except root

14 March 2023

Index files :
 Key value in a sorted order
 corresponding address where you can find the desired record
 very large

→ Original file will not change.

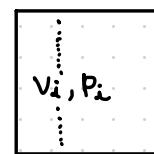
↑ will sort on Key values

→ It takes only 1 block access.

Dense Index



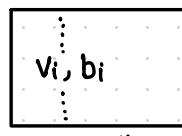
Original
unsorted
file
(Heap file)



Index file
For each record,
Key value & address
of bucket is written.



Original
Sorted
file



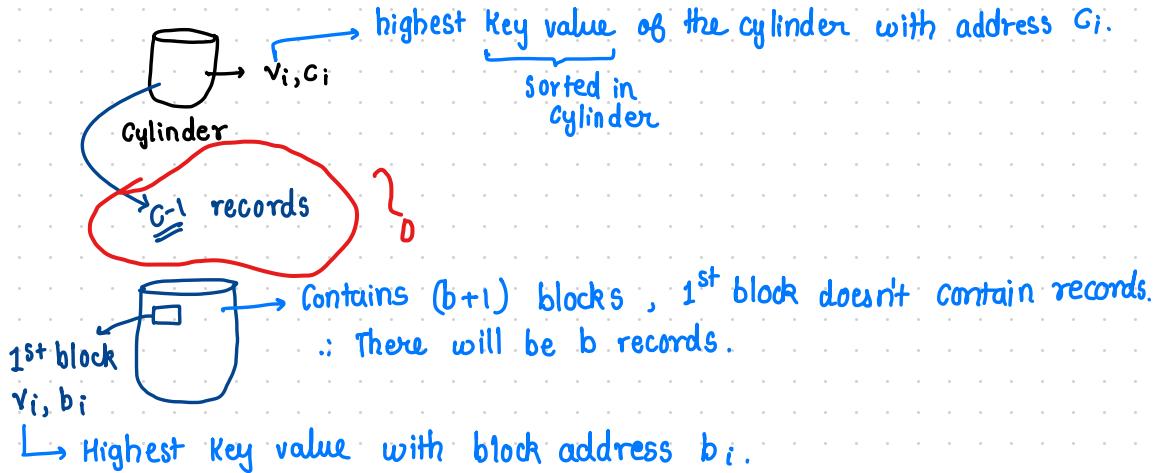
Sparse Index

Index file
First record in
the block with
address b_i and
Key value V_i

Indexed Sequential Access Method (ISAM)

→ File is stored in many cylinders.

C cylinders



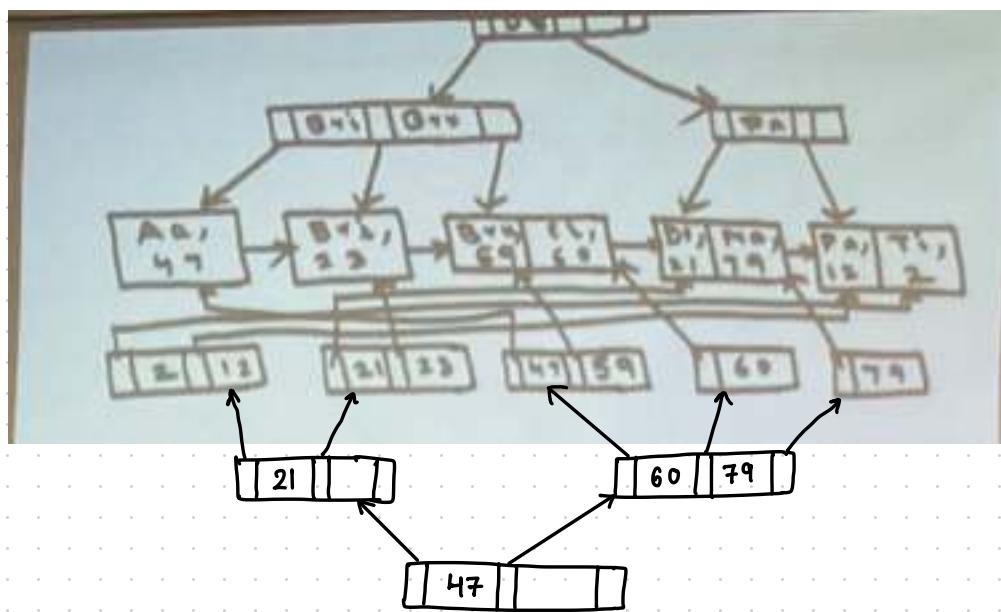
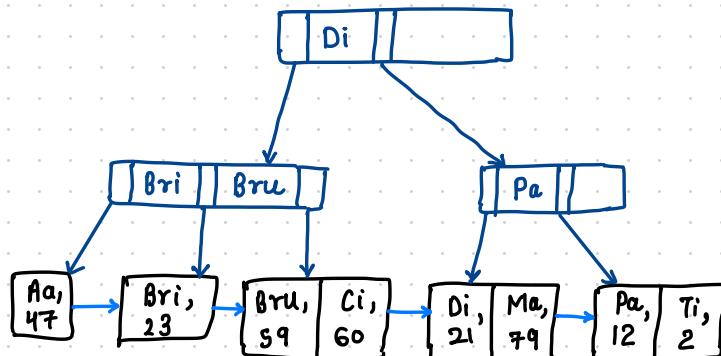
B + Tree of order v :

- Root has atleast 2 children unless it is a leaf.
- No internal node has more than $2v$ keys.
- No internal node except the root has less than v keys.

$$v \leq \leq 2v$$

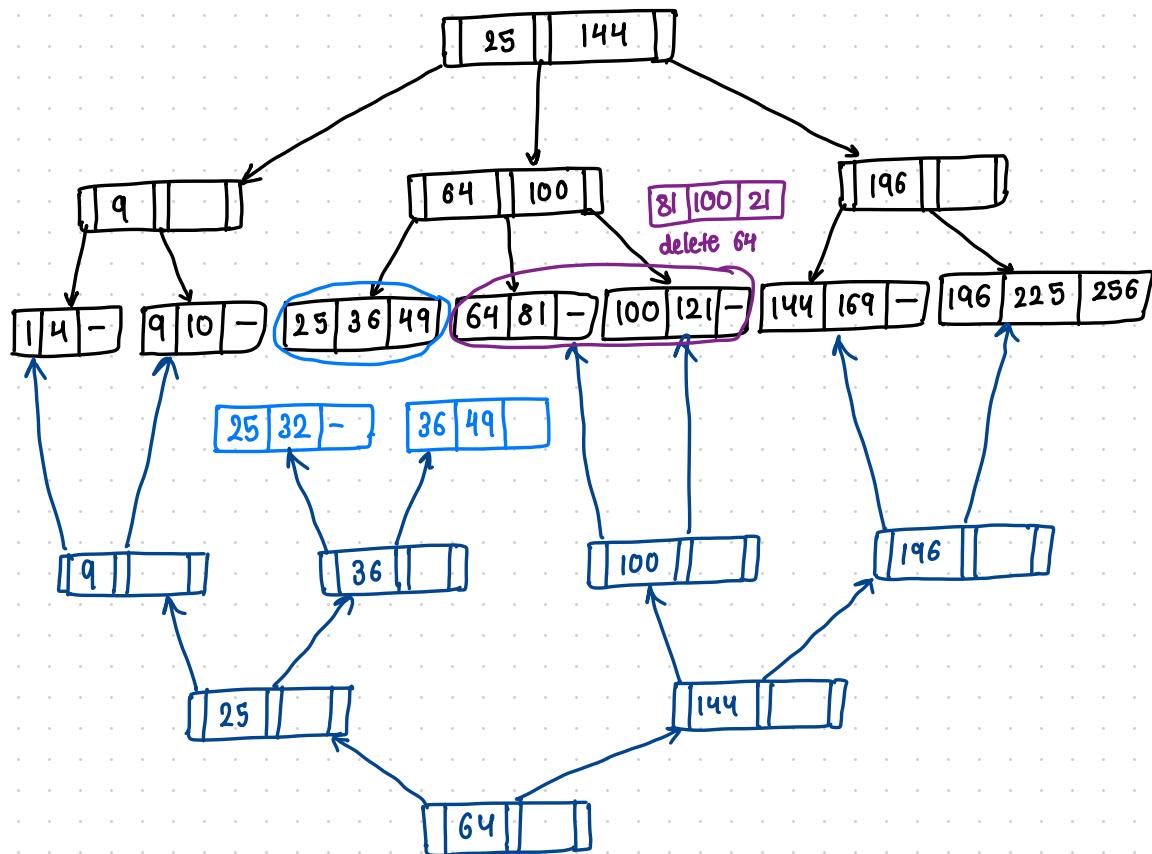
- Internal nodes contain only keys & addresses of nodes in the next level.
- An internal node with K keys has $(K+1)$ children.
- All leafs are at the same level.
- If B + Tree is used as primary index, the leaves of the tree contains data records.

- If B+ Tree is used as secondary indexing , leaves contains key value & record addresses.
- Leaves may also contain address of the next leaf for sequential processing.
- If an internal node is $\frac{2}{3}$ rd full then you will get optimum result.



$$2V = 2$$

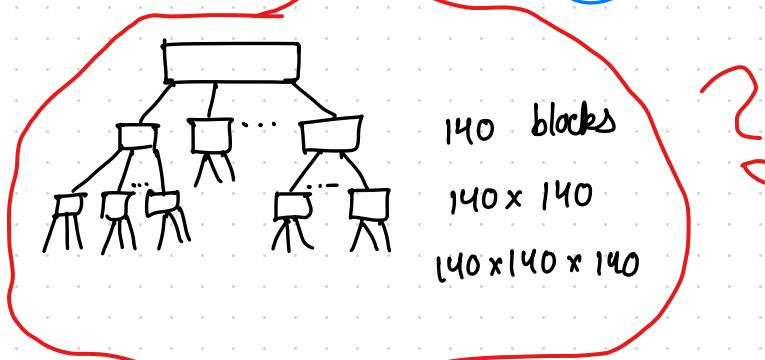
$$V = 1$$



1 block = 2400 bytes

Key value + address = 12 bytes

200



★ Variable length records :

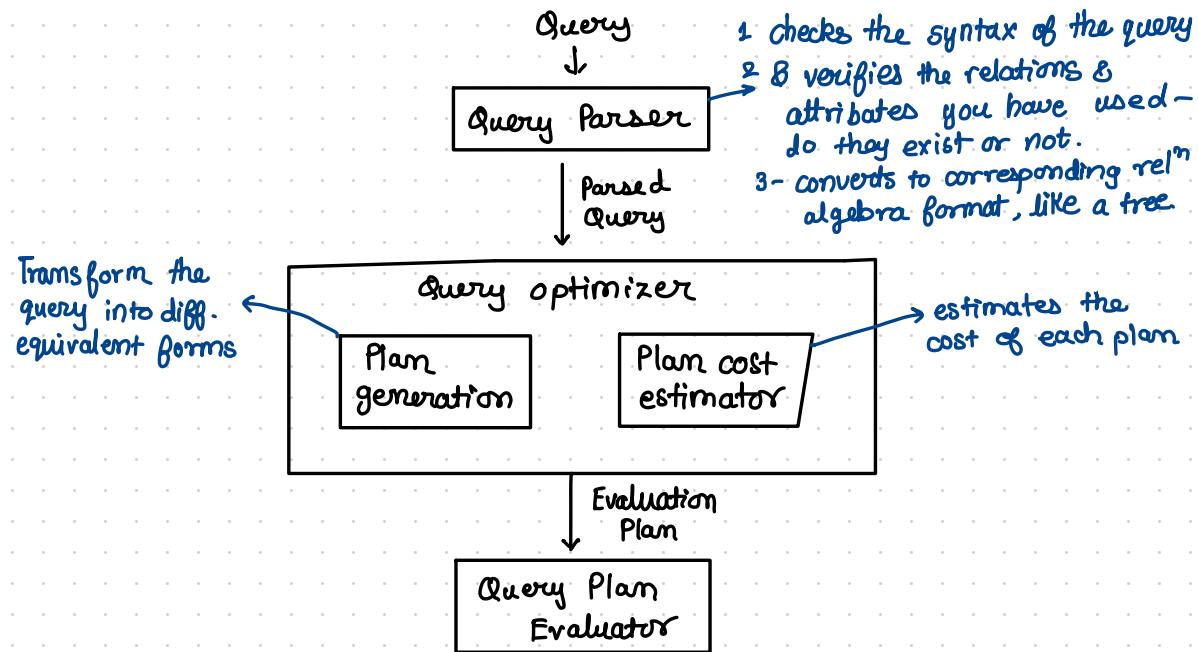
→ Convert it to a fixed length record. Take the maximum length & take it as fixed.

20th March, 2023

★ Query optimization

→ Is the SQL query unique ? No , we may write in various forms.

∴ Query is not executed in your format , it is optimized & then executed. calculate cost of various forms & see the minimum.



★ Equivalence Rules

$$1.) \quad \sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\underbrace{\sigma_{\theta_2}(E)}_{\substack{\text{if fast} \\ \text{then less time}}})$$

• Selection operator is commutative.

$$2.) \quad \sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

$$3.) \quad \pi_{L_1}(\pi_{L_2} \cdots (\pi_{L_n}(E)) \cdots) = \pi_{L_1}(E)$$

$$4.) \quad \sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$$

} whether Cartesian product takes less time or θ -join takes less time.

$$5.) \quad E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1 \quad \} \theta\text{-join is commutative}$$

$$6.) \quad (E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3) \quad \} \text{Natural join \& } \theta\text{-join are associative.}$$

Same for θ -join

$$7.) \quad \sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_{\theta_2} E_2 \quad \} \text{cardinality will be reducing.}$$

↳ If θ_1 is using attributes of E_1 only.

$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_{\theta} (\sigma_{\theta_2}(E_2)) \quad \} \text{when } \theta_1 \text{ involves only } E_1 \text{ \& } \theta_2 \text{ involves only } E_2.$$

$$8.) \quad E_1 \cup E_2 = E_2 \cup E_1 \quad \} \cup \text{ \& } \cap \text{ are interchangeable.}$$

\cap \cap } are associative.

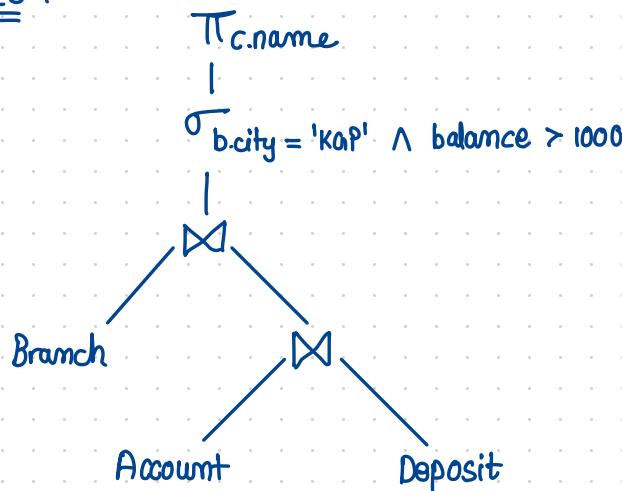
$$9.) \quad \sigma_{\theta}(E_1 - E_2) = \sigma_{\theta}(E_1) - \sigma_{\theta}(E_2)$$

$-$ $-$ } \cup \cup } \cap

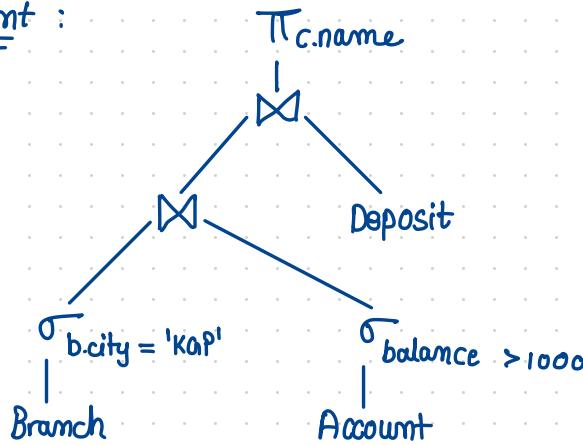
$$(10.) \quad \Pi_L(E_1 \cup E_2) = \Pi_L(E_1) \cup \Pi_L(E_2)$$

Eg: $\Pi_{c.name}(\sigma_{b.city = 'KOP'} \wedge \text{balance} > 1000) \quad (\text{Branch} \bowtie (\text{Account} \bowtie \text{Deposit}))$

Tree :



Equivalent :

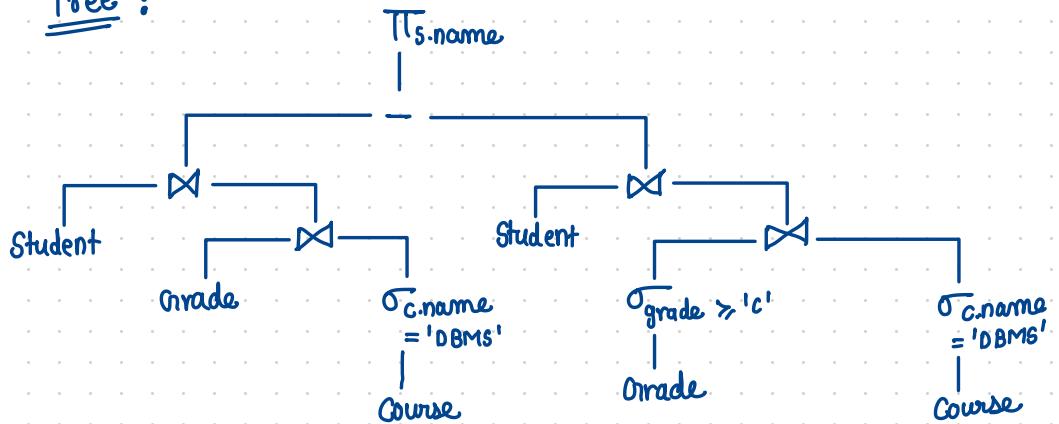


Eg: Find the list of students who have not obtained grade C or higher in DBMS course?

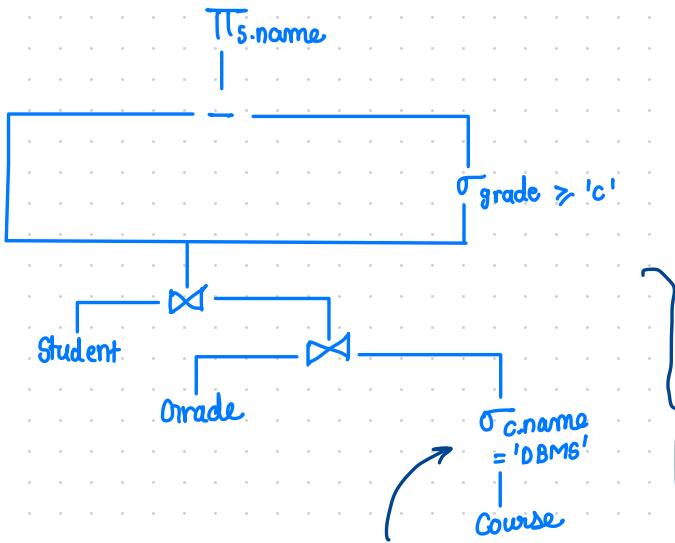
$\Rightarrow \Pi_{s.name} \{ \text{Student} \bowtie (\text{Grade} \bowtie \sigma_{c.name} = 'DBMS' (\text{Course}))$

- $(\text{Student} \bowtie (\sigma_{grade \geq 'C'} (\text{Grade}) \bowtie (\sigma_{c.name} = 'DBMS' (\text{Course}))))$

Tree :



Equivalent :

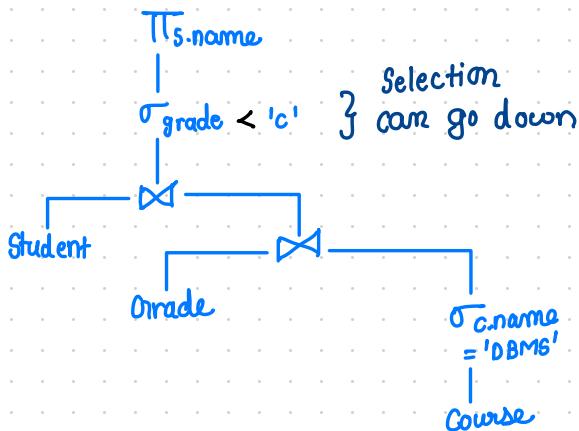


→ ultimately we have to perform selection at the earliest. It will reduce the tuples (hence time).

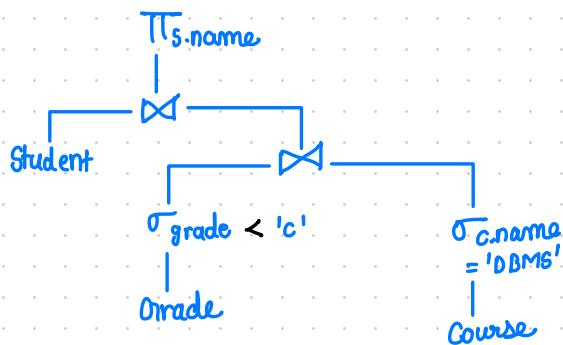
This operation we are doing only once. Earlier we were doing it twice.

$$R - \sigma_c(R) = \sigma_{Tc}(R)$$

Equivalemt :



Equivalent



Recovery from a failure:

→ Your system may fail when you are performing some operation.

- Periodically Backup can be done

- ↓
- Hardware failure
- Software failure
- Human error

* Recovery Schemes

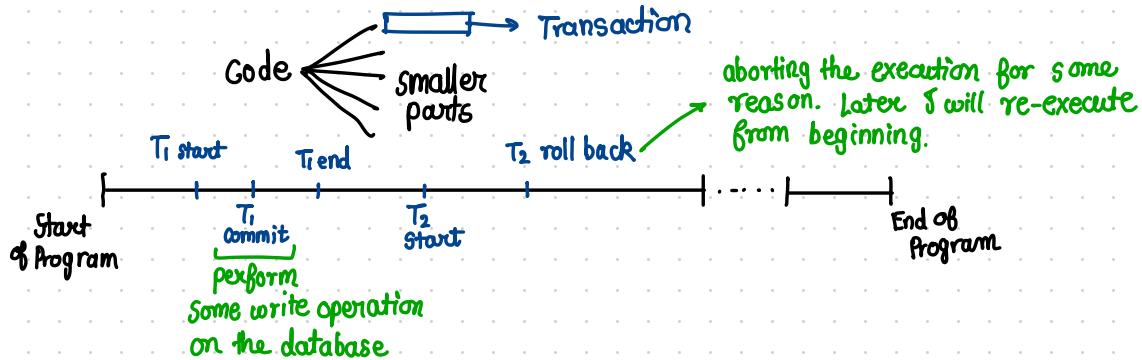
⇒ Forward error recovery : when a particular err in the system is detected, recovery system makes an accurate assessment of the state of the sys. then makes appropriate adjustment based on anticipated --- had the system been -- .

listed problem → corresponding solutions

⇒ Backward error recovery : the system is reset to some previous correct state which is free from errors.

* Transaction :

↳ A program unit whose execution may change the content of a database.



Successful Termination : It has reached the end. (after performing the operation).

Suicidal Termination : System detected some error. it forced the T to roll back

* Properties of a Transaction :

ACID Test

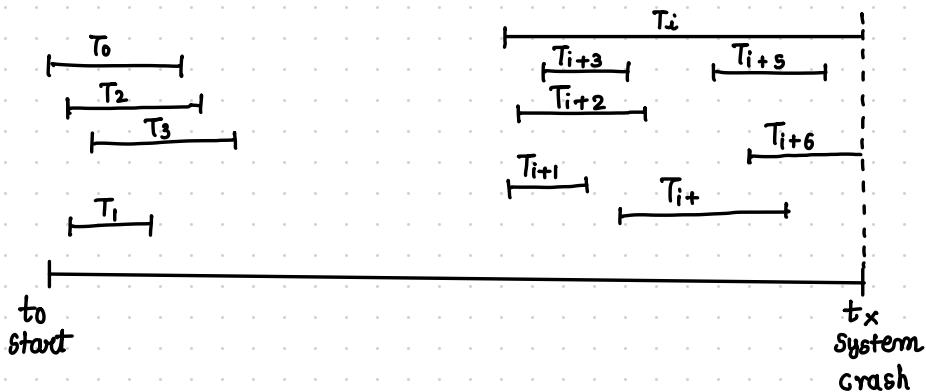
- ⇒ Atomicity Property : Implies that it will run to completion as an indivisible unit. At the end of which no changes have occurred to the database or the database has been changed in a consistent manner.
- ⇒ Consistency Property : This property implies that if the database was in a consistent state before the start of the transaction then on termination of the transaction , the database also will be in a consistent state.
- ⇒ Isolation Property : This property indicates that the actions performed by the T will be isolated or hidden from outside the transaction until the transaction terminates.
- ⇒ Durability Property : Ensures that the commit action of a Transaction on its termination will be reflected on the database.

* Recovery in Centralized DBMS :

Log : a file on the secondary device which stores the info about all transaction actions , operation performed. The prev. values of the modified data items.
→ storage can be a problem.

Write ahead log strategy :

Before writing on the database, write on the log file.

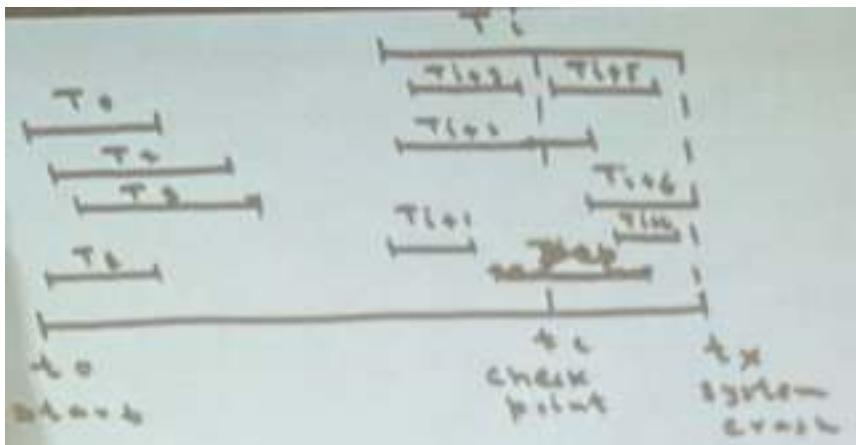


Checkpoint :

Transaction - consistent checkpoint : Here the T that are active when system timer signals a checkpoint are allowed to complete but no new transaction are allowed to be started until the checkpoint is complete.

Action consistent checkpoint : Active transactions are allowed to complete the current step & no new actions are allowed to be started on Database while the checkpoint complete.

T-oriented checkpoint : The idea is to take a check-point at the end of each transaction by forcing the log of the T on stable storage.



UNDO ($T_i, T_{i+2}, T_{i+4}, T_{i+5}, T_{i+6}$)



REDO ($T_{i+4}, T_{i+5}, T_{i+2}$)

UNDO (T_i, T_{i+6})

* Do, Undo, Redo :

- A transaction on the current database transforms it from current state to new state \Rightarrow Do operation
- Undo operation comes into picture when a transaction decides to terminate itself.
- T REDO involves performing the changes made by a T that committed before a system crash with the write ahead log strategy. A committed T implies that a log for the T could be written to secondary storage but the physical database may or may not have been modified before the system failure.

Concurrency Management

Schedule: The order in which the statements of transactions are executed is known as schedule.

T₁

1. Read (Avg. F. sal)

2. Avg. F. sal = $f_1(\text{Avg-F-sal})$

3. Write (Avg. -F-Sal)

T₂

4. Read (Avg. S. sal)

5. Avg. S. sal = $f_2(\text{Avg.S.sal})$

6. Write (Avg. S. Sal.)

S₁

1 2 3 4 5 6

S₂

4 5 6 1 2 3

S₃

1 4 2 5 3 6



AIM: Schedule should give you the correct result.
(same)

T₃

1. Read (A)

T₄

4. Read (A)

A = 200

2. A = A + 10

5. A = A * 1.1

3. Write (A)

6. write (A)

S₁
T₁ T₂

S₂
T₂ T₁

S₃

4 5 1 2 6 3

A=231

A=230

A $\frac{200}{210}$

Loss update

$\frac{200}{220}$

↓ write

A $\frac{210}{210}$

↓ write

From

{ T₃ }

→ 210

finally.

T₅

1 Read(A)

2 A = A - 100

3 write(A)

4 Read(B)

5 B = B + 100

6 write(B)

T₆

7 Sum = 0

8 Read(A)

9 Sum = Sum + A

10 Read(B)

11 Sum = Sum + B

12 write(Sum)

A = 500

B = 1100

S₁

1 7 8 2 3 9 4 5 6 10 11 12

A 500

Sum 0

A 400

A 500

B = 1100

Sum = 500

B = 1200

B = 1200

B = 1700

Sum = 1700

→ But sum should be 1600
bcz we are decreasing A
& increasing B
both by 100.

* Serializable Schedule :

→ Given a set of n-transactions, the schedule will be serializable if following conditions hold.

1. All transactions are correct in the sense that any one of the T is executed by itself on a consistent database, the resulting database will be consistent.
2. Any serial execution of the T is also correct & preserves the consistency of the database. The results obtained are correct.

* Precedence Graph :

$G_1(V, E)$
Set of vertices
The T of the schedule
↓
Directed edges

$T_i \rightarrow T_j$
Read → write
write → Read

→ A directed edge from Node T_i to Node T_j indicate one of the following condition :

1. T_j performs the operation read(A) to read the value written by T_i .
2. T_j performs write(A) after T_i performs read(A)

<u>Schedule</u>	<u>T_q</u>	<u>T₁₀</u>	<u>T₁₁</u>	{ J vertices }
Read (A)	✓			
$A = f_1(A)$	✓			
write (A)	✓			
Read (A)	-----	✓		
$A = f_2(A)$		✓		
write (A)		✓		
Read (B)		✓		
$B = f_3(B)$		✓		
write (B)		✓		
Read (B)	-----	-----	-----	✓
$B = f_4(B)$				✓
write (B)				✓



Schedule

Read(A)

T₁₂

✓

 $A = f_1(A)$ T₁₃

✓

Read(A)

✓

 $A = f_2(A)$

✓

write(A)

✓

Write(A)

✓

13 writes after 12 reads

→ Schedule is not serializable
(Loop is present)

12 writes after 13 reads

ST₁₄T₁₅T₁₆

Read(A)

✓

Read(B)

✓

 $A = f_1(A)$

✓

Read(C)

✓

 $B = f_2(B)$

✓

write(B)

✓

 $C = f_3(C)$

✓

write(C)

✓

write(A)

✓

Read(B)

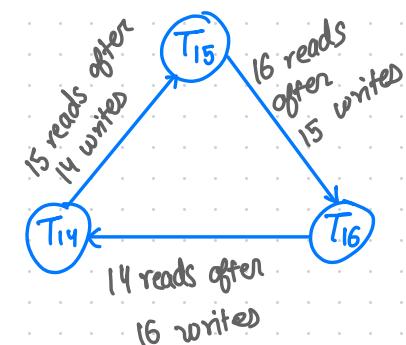
✓

Read(A)

✓

 $A = f_4(A)$

✓



Read (C)



write (A)



C = f_s (C)



write (C)



B = f_s (B)



write (B)



★ Locking System

Lock



(X) Exclusive Lock : T can update or write the dataitem & No

↓ other T can even read it until the T is unlocking it.

Exclusive use of dataitem for 1 transaction.

(S) Shared Lock (Read lock) : can read dataitem & work with it but can't modify the dataitem

T₅

T₆

Lock X (sum)

Sum = 0

Lock S (A)

Read (A)

Sum = Sum + A

unlock (A)

Lock X (A)

	<u>T₅</u>	<u>T₆</u>
Lock X (sum)		✓
Sum = 0		✓
Lock S (A)		✓
Read (A)		✓
Sum = Sum + A		✓
unlock (A)		✓
Lock x (A)	-----✓	
Read (A)		✓
A = A - 100		✓
write (A)		✓
Unlock (A)		✓
Lock x (B)		✓
Read (B)		✓
B = B + 100		✓
write (B)		✓
unlock (B)		✓
Lock x (B)	-----✓	
Read (B)		✓
Sum = Sum + B		✓
Write (Sum)		✓
Unlock (B)		✓
Unlock (Sum)		✓

★ Two Phase Locking : It has 2 phases

A growing phase where the no. of blocks increases from 0 to a Max value for the transaction
 &

A contracting phase where the no. of blocks held decreases from max value to zero.

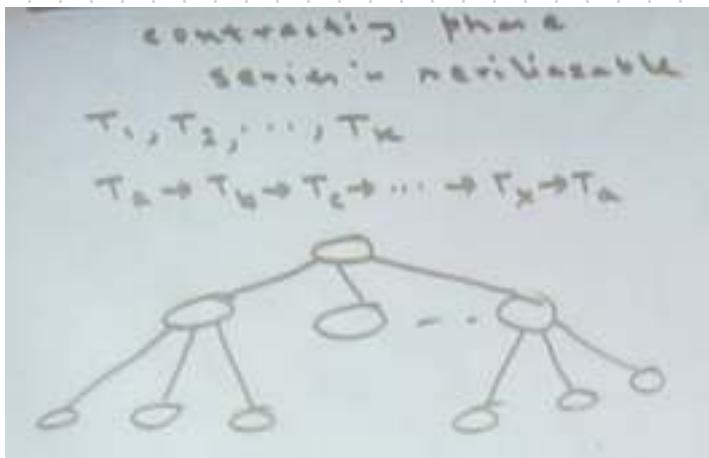
→ Series is serializable if a schedule follows 2 phase locking.

Proof by contradiction : Suppose the series is not serializable.

→ There is a cycle.

$T_a \rightarrow T_b \rightarrow T_c \rightarrow \dots \rightarrow T_x \rightarrow T_a$

→ Once you lock & unlock them you can't lock again.

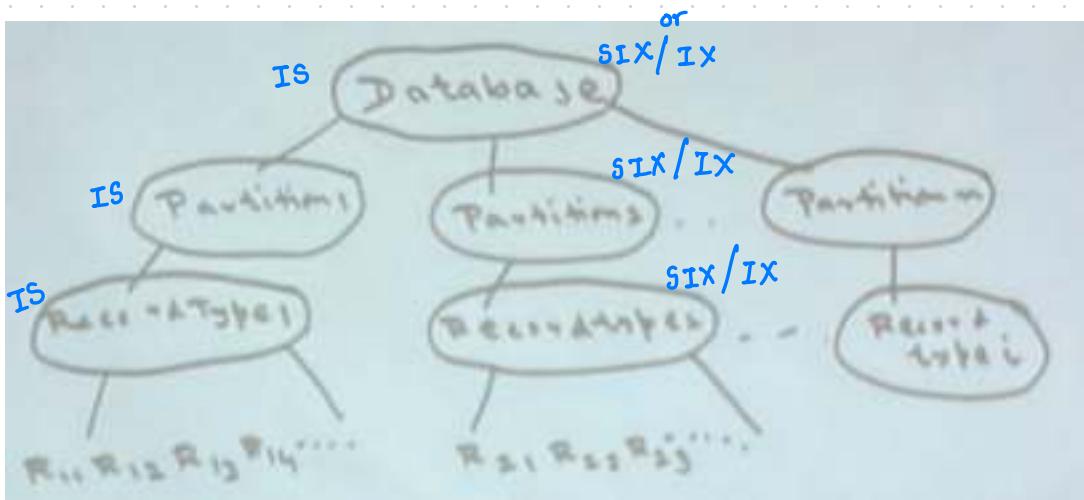
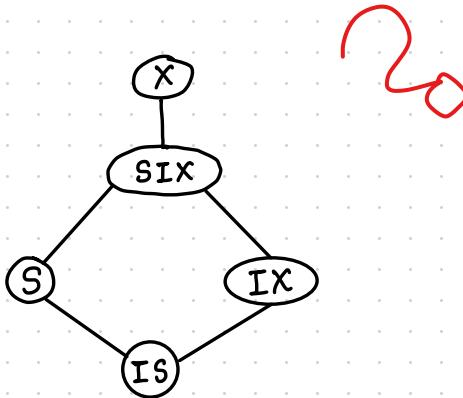


Intention share (IS): The node or its descendants cannot be exclusively locked. The descendant may be locked in S or IS mode.



Intention Exclusive (IX) : the node itself can't be exclusively locked. However the descendants can be locked in any of the locking modes.

Shared and Intention Exclusive (SIX) : Node locked in SIX mode & all the children are implicitly locked in S mode but any of the descendants can be locked in X or IX mode.



$R_{13} \rightarrow$ retrieve

$R_{13} \rightarrow S$

$R_{22} \rightarrow$ update

$R_{22} \rightarrow X$

★ Timestamp Based ordering Protocol :

$T_i \rightarrow$ time stamp value t_i

$T_i(t_i) \& T_j(t_j)$

s.t. $t_i < t_j$
older younger

$X : \{x, w_x, R_x\}$

x : value of X

w_x : the system clock value when the dataitem was written for the last time.

R_x : Read time stamp value

→ A transaction T_a with timestamp value t_a issues a read operation for the data item X with values $\{x, w_x, R_x\}$

- 1) This request will be allowed if $t_a \geq w_x$
- 2) This request will fail if $t_a < w_x$

→ A transaction T_a with timestamp value t_a issues a write operation for the data item X with values $\{x, w_x, R_x\}$

- 1) The write operation will be allowed if $t_a \geq w_x$ & $t_a \geq R_x$
- 2) The write operation will fail if $t_a < R_x$

A younger transaction is already using the value of x . So, t_a is not allowed to modify.

3) $R_x \leq t_a < W_x$, younger T has already updated the value of x . write operation is ignored \rightarrow not allowed.

<u>Schedule</u>	<u>$T_{22} (t_{22})$</u>	<u>$T_{23} (t_{23})$</u>
Sum = 0	✓	
Read(A)	✓	
Sum = Sum + A ✓	
Sum = 0		✓
Read(A)		✓
A = A - 100		✓
write(A) ✓	
Read(B)	✓	
Sum = Sum + B	✓	
Show(Sum)	✓	
Sum = Sum + A ✓	
Read(B)		✓
B = B + 100		✓
write(B)		✓
Sum = Sum + B		✓
Show(Sum)		✓



Initially $\Rightarrow A : 400, W_a, R_a$ $B : 500, W_b, R_b$

Step 2 : $A : 400, W_a, t_{22}$ "

Step 5 : $A : 400, W_a, t_{23}$ "

Read
write
operations

Step 7 : A : 300, t_{23}, t_{23}

11

Step 8 : " B : 500, w_b, t_{22}

Step 10 : Sum = 900

Step 12 : B : 500, w_b, t_{23}

Step 14 : B : 600, t_{23}, t_{23}

Step 16 : Sum = 900

Schedule

$T_{22}(t_{22})$

$T_{23}(t_{23})$

Sum = 0

Read (A)

Sum = Sum + A

Sum = 0

Read (A)

A = A - 100

write (A)

Read (B)

Sum = Sum + B

Show (sum)

Sum = Sum + A

Read (B)

B = B + 100

write (B)

Sum = Sum + B

Show (sum)

1. Sum = 0

2.	$sum = 0$	✓
3.	$RAND(A)$	✓
4.	$A = A + 100$	✓
5.	$WHITE(A)$	✓
6.	$RAND(A)$	✓ * Rollback
7.	$sum = sum + A$	✓
8.	$RAND(B)$	✓
9.	$B = B + 100$	✓
10.	$WHITE(B)$	✓
11.	$sum = sum + B$	✓
12.	$SHOW(SUM)$	✓
→ 13.	$sum = 0$	✓
14.	$RAND(A)$	✓
15.	$sum = sum + A$	✓
16.	$RAND(B)$	✓
17.	$sum = sum + B$	✓
18.	$SHOW(SUM)$	✓

$T_{22} \ t'_{22} (> t_{23})$

A : 400, Wa, Ra B : 500, Wb, Rb

Step 3 A : 400, Wa, t_{23}

"

Step 5 A : 300, t_{23}, t_{23}

"

Step 6 → Not allowed

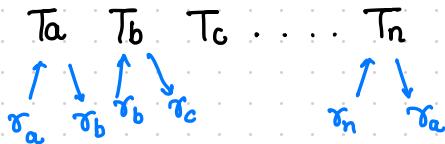
Deadlock

w/ Next class

who missed mid-sem
test on 7th April
from 10-12
mail to prof, your
Roll No.

No other test.

★ Deadlock and its Resolution:



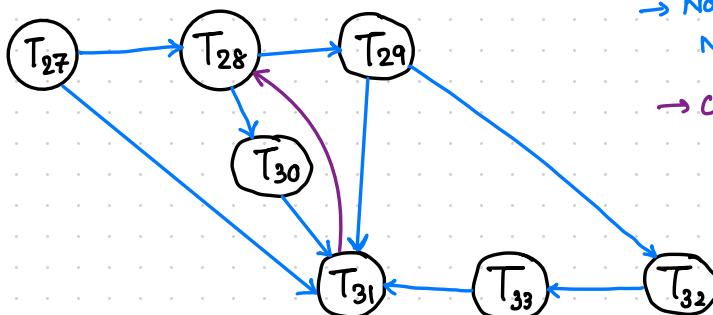
Wait for graph :

T-id	Items - locked	Items waiting for
------	----------------	-------------------

T ₂₇	B	C, A
T ₂₈	G, M	H, G
T ₂₉	H	D, E
T ₃₀	G	A
T ₃₁	A, E	C* ← added later
T ₃₂	D, I	F
T ₃₃	F	E



→ T_i is waiting for the data-item currently allocated to and held by T_j.



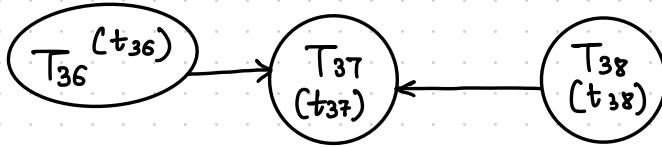
→ No cycle
No deadlock

→ Cycle present
Deadlock occurs.

Wait - Die :

Wait & Die scheme for Deadlock is as follows:

- 1) If the requesting transaction is older than the transaction that holds the lock on the requested data item. The requesting T is allowed to wait.
- 2.) If the requesting T is younger than the T that holds the lock on the requested data-item. The requesting T is aborted and rolled back.



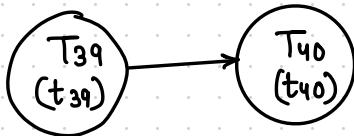
$$t_{36} < t_{37} < t_{38}$$

T₃₆ → allowed to wait

T₃₈ → Roll back

Wound - wait

1. If a younger T holds a data item requested by older one, younger T will be aborted and rolled back.
2. If a younger T requests a data item held by older T, the younger T is allowed to wait.



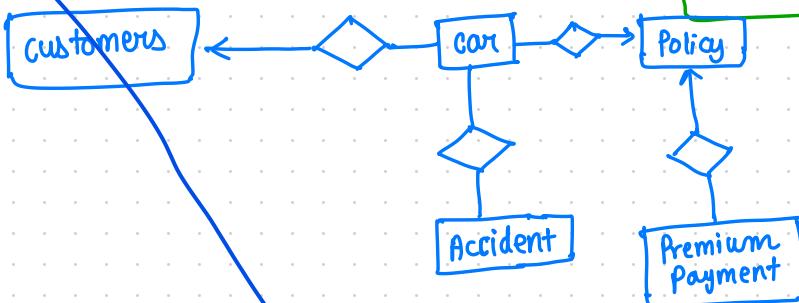
$t_{39} < t_{40}$

aborted & rolled back.

~~midsem solution~~

7th → 11 to 1
 MA seminar zoom
 check midsem paper.
 No other time

①



② $F = \{AB \rightarrow C, AB \rightarrow D, C \rightarrow A, D \rightarrow B\}$

AB, BC, CD, AD

3NF ✓ & Not BCNF

No BCNF decomposition preserving dependencies.

③

R & F

$$\{A \rightarrow B, C \rightarrow D, B \rightarrow C\}$$

$$R_1 = \{(A, B), (C, D), (B, C)\}$$

$$R_2 = \{(A, B), (C, D), (A, C)\}$$

$$R_3 = \{(B, C), (A, D), (A, B)\}$$

$$(ii) \quad \begin{array}{c} A \\ p, 2, 3, 4, 5 \\ B \\ 2, P, 3, 7, 5 \\ C \\ p, 2, 3, 4, 6 \\ D \end{array} \quad \left. \right\}$$

$BC \rightarrow D$ violated

$$\Rightarrow p = 2$$

$BC \rightarrow D$ violated

$$\Rightarrow p \neq 2$$

- (4)
- (i) YES - 1 M
proper justification for loss less join - 1 M } 2 M.
 - (ii) BCNF \rightarrow 3NF
implies
logic
 - (iii) $R \div S = \Pi_{A-B}(R) - \Pi_{A-B}[\Pi_{A-B}(R) \times S - R]$ 1M
example: 1 M

- (5)
- (i) user, Borrow
 - (ii) Borrow, Supplier, Supply
 - (iii) Supply , Book

- (6)
- (i) Loan
 - (ii) All
 - (iii) Loan, Borrower

Distributed Database

Coll' of multiple logically interconnected database on a network, and S/W used here is

DDBMS

Shared Memory Architecture

Share the same Secondary disk. 3 primary disk

Shared disk architecture

Multiple processor share the same sec disk but they have their own primary disk

Share nothing architecture

Multiple units have individual CPU, Disk, Database

Centralised Database Architecture



Truly Distribute



Advantages of Distributed Database

↳ Management of Dist. Data with almost level of transparency

~~where we're not disclosing where data is stored~~

~~we're not disclosing, where the data is stored.~~

↳ Dist. on net. transparency

This refers to filters for the user for the Operational detail.

① Location transparency → query doesn't need to "loc"

② Naming transparency → Once a name is specified, the named object can be accessed immediately availability without additional

↳ Replication transparency

Makes the user unaware of existence of multiple copies

+ Better availability

+ Performance

+ Reliability

↳ Fragmentation transparency

→ Horizontal fragmentation

→ Vertical "

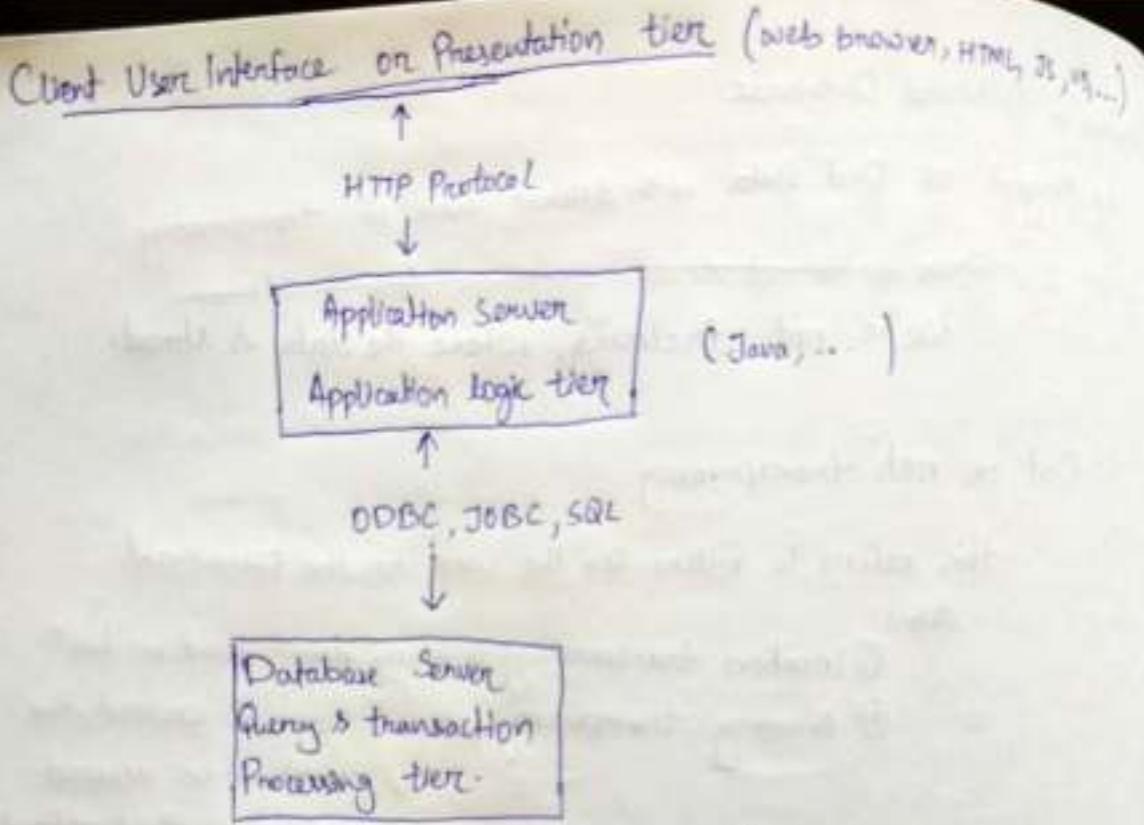
→ N-Grid

↳ Increased reliability and availability

↳ System is available on time

↳ Improved performance

↳ Easy expansion

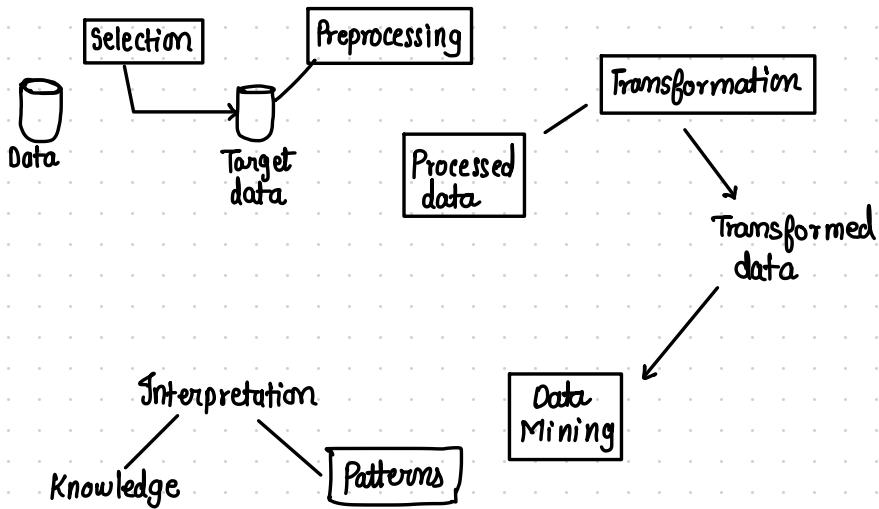


★ Data Mining

is a part of
Knowledge discovery in Databases (KDD)

→ Collect non-trivial, relevant data from a huge database.
or
discover

→ detect frauds in bank, etc.



- * Association Rule Mining → Mall : you purchase 1 thing, and another put aside it
- * Classification → A person of 20 years & having good credit points → he will buy a computer or not.
- * Clustering / Similarity Matching
 - ↳ predicting where should KFC open its new outlet in KGP.

<u>TID</u>	<u>Item</u>
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Itemset

Itemset is a collection of one or more items.

$$\hookrightarrow \{Bread\}, \{Milk, Bread\}$$

K- Itemset

An itemset which contain K-items.

Support Count (σ)

Frequency of occurring of an itemset in a set of Transactions.

$$\sigma \{Milk, Beer, Diaper\} = 2$$

Support : Fraction of Transactions that contain an itemset.

$$S \{Milk, Beer, Diaper\} = \frac{2}{5}$$

Frequent Item sets : An itemset whose support is \geq Minimum Support Threshold

Association Rule Discovery : $\{Diaper\} \rightarrow \{Beer\}$
 $\{Milk, Bread\} \rightarrow \{Coke\}$

d unique items,

$$\text{Total no. of itemsets} = 2^d$$

$$\text{Total no. of possible association rules} = 3^d - 2^{d+1} + 1$$

★ Apriori Algorithm (late 90's)

Frequent itemset

Any subset of frequent itemset must be frequent.

L_k : Set of candidate K -itemsets

L_{k-1}

Join Step : C_k is generated by joining L_{k-1} with itself.

↓
Candidate itemset
of size K

$L_k \Rightarrow$ frequent itemset
of size K .

↓
from G_k I will obtain L_k .
Possible itemsets
if you join K items
together.

Prune Step : Any $(K-1)$ itemset that is not frequent cannot be a subset of frequent K itemset.

$L_1 : \{ \text{frequent items} \}$;

for ($K=1$; $L_K \neq \emptyset$; $K++$) do begin

C_{K+1} = candidates generated from L_K ;

for each transaction t in database DO

increment the count of all
candidates in C_{K+1} that are
contained in t

B_{K+1} = Candidates in C_{K+1} with a min-support.

end

return $U_K L_K$;

TID	Items	min-support $= \frac{3}{9} = 33\%$	Minimum confidence $= 70\%$
T1	I1, I2, I5		
T2	I2, I4		
T3	I2, I3		
T4	I1, I2, I4		
T5	I1, I3		
T6	I2, I3		
T7	I1, I3		
T8	I1, I2, I3, I5		
T9	I1, I2, I3		

S1

IT	SC
$\{I_1\}$	6
$\{I_2\}$	7
$\{I_3\}$	6
$\{I_4\}$	2
$\{I_5\}$	2

IT	SC
$\{I_1\}$	6
$\{I_2\}$	7
$\{I_3\}$	6
$\{I_4\}$	2
$\{I_5\}$	2

C1

L1

S2

$L_1 \text{ join } L_1$

$\{I_1, I_2\}$	4
$\{I_1, I_3\}$	4
$\{I_1, I_4\}$	1
$\{I_1, I_5\}$	2
$\{I_2, I_3\}$	4
$\{I_2, I_4\}$	2
$\{I_2, I_5\}$	2
$\{I_3, I_4\}$	0
$\{I_3, I_5\}$	1
$\{I_4, I_5\}$	0

$\{I_1, I_2\}$	4
$\{I_1, I_3\}$	4
$\{I_1, I_4\}$	1
$\{I_1, I_5\}$	2
$\{I_2, I_3\}$	4
$\{I_2, I_4\}$	2
$\{I_2, I_5\}$	2
$\{I_3, I_4\}$	0
$\{I_3, I_5\}$	1
$\{I_4, I_5\}$	0

L₁ join L₁

{I ₁ , I ₂ }	4
{I ₁ , I ₃ }	4
{I ₁ , I ₄ }	1
{I ₁ , I ₅ }	2
{I ₂ , I ₃ }	4
{I ₂ , I ₄ }	2
{I ₂ , I ₅ }	2
{I ₃ , I ₄ }	0
{I ₃ , I ₅ }	1
{I ₄ , I ₅ }	0
	c ₂

{I ₁ , I ₂ }	4
{I ₁ , I ₃ }	4
{I ₁ , I ₅ }	2
{I ₂ , I ₃ }	4
{I ₂ , I ₄ }	2
{I ₃ , I ₅ }	2

L₂ join L₂

(S3)

{I ₁ , I ₂ , I ₃ }	2
{I ₁ , I ₂ , I ₅ }	2

G₃
=

{I ₁ , I ₂ , I ₃ }	2
{I ₁ , I ₂ , I ₅ }	2

L₃
=

(S4)

C₄: L₃ join L₃

I₁, I₂, I₃, I₅ ⚡ can't be frequent

C₄ = \emptyset

STOP

$$L = \{ \{I_1\}, \{I_2\}, \{I_3\}, \{I_4\}, \{I_5\},$$

*union
of all L's* $\{I_1, I_2\}, \{I_1, I_3\}, \{I_1, I_5\}, \{I_2, I_3\}, \{I_2, I_4\}, \{I_2, I_5\},$
 $\{I_1, I_2, I_3\}, \{I_1, I_2, I_5\} \}$

Association Rule

I S is subset of I

$S \rightarrow (I-S)$ if

$$\frac{\text{Support-count}(I)}{\text{Support-count}(S)} \geq \text{min. confidence}$$

$\begin{matrix} \{I_1, I_2, I_5\} \\ \diagdown \quad \diagup \quad \diagup \diagdown \\ \{I_1, I_2\}, \{I_1, I_5\}, \{I_2, I_5\}, \{I_1\}, \{I_2\}, \{I_5\} \end{matrix} \quad \left. \right\} \text{possible } S\text{-values}$

R1 : $I_1, I_2 \rightarrow I_5$ $\frac{SC\{I_1, I_2, I_5\}}{SC\{I_1, I_2\}} = \frac{2}{4} = \underline{\underline{50\%}}$

Reject

R2 : $I_1, I_5 \rightarrow I_2$ $\frac{SC\{I_1, I_5, I_2\}}{SC\{I_1, I_5\}} = \frac{2}{2} = \underline{\underline{100\%}}$

R₂ is selected

⋮