Advance DBMS

Field → Marks = [10, 20, 30, 40]

Record → Collection of fields

Admn	Name	Marks
1	x	140
2	y	150

Data files → Collection of Records

Database → Collection of Data files

Data → fields → Record → Data files → Database

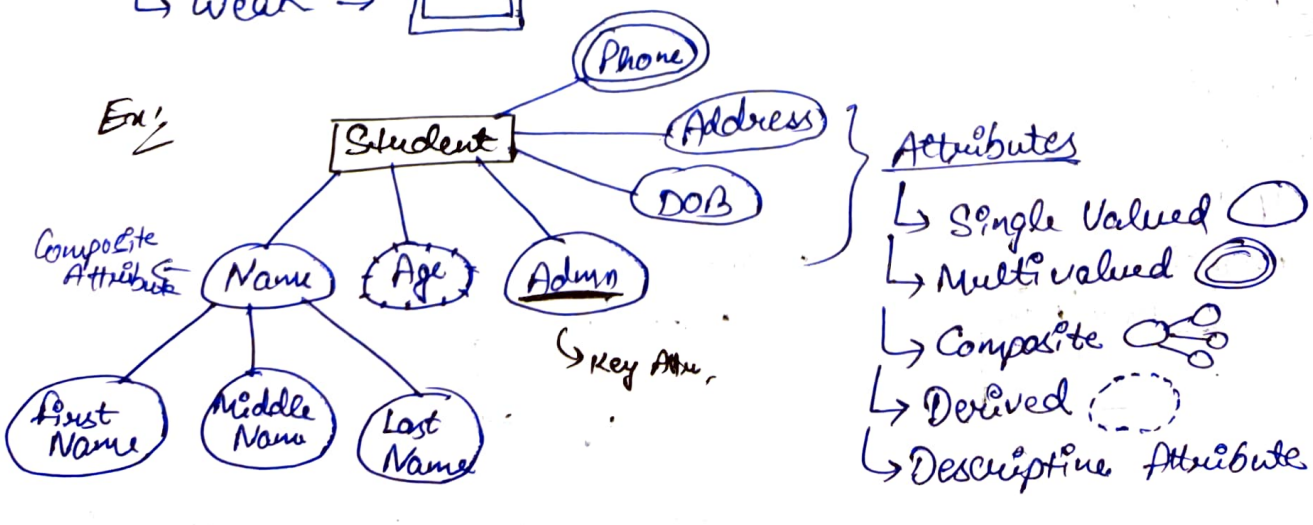
DBMS (Database Management System)

- ↳ Efficient retrieval, insertion of data.
- ↳ Main thing is time.

Entity Relationship Diagram

↳ Strong → ▭

↳ Weak → ▭

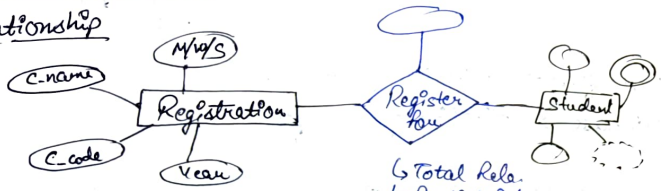


Keys:-

- → Primary key
- → Partial key
- → Candidate key
- → Super key
- → Foreign key

ER-Model
↓
Relational Model

Relationship



Conversion into Relations/Tables

Student

Ad-no	DOB	Name.First_Name	Name.MN	Name.LN	Age

Student_Phone

Ad-no.	Phone

Anamolies

- ↳ Insert
- ↳ Delete
- ↳ Update

Branch

B_Name	Addr.

Account

A_no.	B_NO

If the Branch name is there in Branch Relation, then only Account can be created.

Branch (BN, BC, Assets)
Customer (CN, CS, CC)
Account (AN, BN, BAL)
LOAN (LN, BN, AMT)
Depositor (CN, AN)
Borrower (CN, LN)

Basic Operations:-

- → Select
- → Project
- → Union
- → Set Difference
- → Cartesian Product
- → rename

Additional Operations

- → Set Intersection ($r \cap s$) = {t | t ∈ r and t ∈ s}
- → Natural Join [$r \bowtie s$]
- → Division [$r \div s$]
- → Assignment [$x \leftarrow \pi_{BN}(\sigma_{re}(Branch))$]

Q. Find names of the customer who have both loan & Account

Sol. $\Pi_{CN}(depositor) \cap \Pi_{CN}(Borrower)$

Natural Join:-

Let r, s be the relations on schema R(R) & S(S) respectively. Then, $r \bowtie s$ is a relation on schema RUS obtained as follows:-

1) Consider each pair of tuples t_r from r and t_s from s.
2) If t_r & t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result where,
   a) t has the same value as t_r on r
   b) t has the same value as t_s on s.

r

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

s

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

$r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

$$r \bowtie s = \Pi_{r.A, r.B, r.C, r.D, r.E}\left(\sigma_{r.B = s.B \wedge r.D = s.D}(r \times s)\right)$$

Q. Find the names of all branches with Customers who have an account at the bank and who live in Dhanbad city.

Sol:- Depositor (CN, AN) → for accounts

Customers (CN, CS, CC) → for city

Account (AN, BN, BAL) → for Branch Name

[Customer ⋈ Depositor] [CN, CS, CC, AN] ⋈ Account [CN, CS, CC, AN, BN, BAL]

$$\Pi_{BN}\left(\sigma_{CC = \text{"Dhanbad"}}\left((\text{Customer} \bowtie \text{Depositor}) \bowtie \text{Account}\right)\right)$$

Division

r

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ϵ	6
ϵ	1
β	2

s

B
1
2

$r \div s$

A
α
β

⇒ for all values of B which A values have tuple is the answer

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}\left((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r)\right)$$

r

A	B
a_1	b_1
a_1	b_2
a_2	b_1
a_3	b_1
a_4	b_2
a_5	b_1
a_5	b_2

$r \div s$

B		A
b_1	→	a_1
b_2		a_5

$r \div s$

B		A
b_1	→	a_1
		a_2
		a_3
		a_5

s

B		A
b_1	→	–
b_2		–
b_3		

B		A
–	→	a_1
–		a_2
		a_3
		a_4
		a_5

Q. Find names of all customers who have an account at all branches at Dhanbad city.

Sol Branch, Account, Depositor

⇒ $\Pi_{BN}\left(\sigma_{BC = \text{"Dhanbad"}}(\text{Branch})\right)$

(List of branches located in Dhanbad)

$\Rightarrow \Pi_{CN,BN}(\text{Depositor} \bowtie \text{Account})$

$\div$

$\Pi_{BN}(\sigma_{BC=\text{"Dhanbad"}}(\text{branch}))$

CN	BN
—	—
—	—

Extended Relational Algebra operation

→ Generalized Projection → $\Pi_{f_1, f_2, \ldots f_n}(emp)$

→ Outer Join

→ Aggregate function

r

A	B	C
1	2	3
4	5	6

$\Pi_{A,B,C,\ A+B+C,\ 2\times A,\ C/B}(r)$

A+B+C	2×A	C/B
—	—	—
—	—	—

Outer Join

r

A	B	C
1	2	3
4	5	6
7	8	9

s

C	D
3	10
9	12
10	11

$r \bowtie s$ (Inner Join or Natural Join)

A	B	C	D
1	2	3	10
7	8	9	12

$r \bowtie s$ (Left Outer Join)

A	B	C	D
1	2	3	10
7	8	9	12
4	5	6	Null

$r \bowtie s$ (Right Outer Join)

A	B	C	D
1	2	3	10
7	8	9	12
Null	Null	10	11

$r \bowtie s$ (full Outer Join)

A	B	C	D
1	2	3	10
7	8	9	12
4	5	6	Null
Null	Null	10	11

Aggregate Functions

↳ avg()
↳ max()
↳ min()
↳ count()
↳ Sum()

$G_1, G_2, \ldots G_n \mathcal{S}\ f_1(A_1), f_2(A_2) \ldots f_n(A_n)\ (E)$

- $G_1, G_2, \ldots G_n$ — Grouping Criteria (optional)
- $\mathcal{S}$ — Symbol for Aggregate function
- $f_1(A_1), f_2(A_2) \ldots f_n(A_n)$ — functions on which task is performed

$\mathcal{S}_{sum(c)}(r)$

$\Rightarrow 27$

r

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

Account

BN	AN	BAL
Dhan	A1	400
Dhan	A2	900
Ranchi	A3	750
Ranchi	A4	750
Jhr	A5	700

$BN\,\mathcal{S}\ Sum(BAL)\ (account)$

$\Rightarrow$

BN	Sum(BAL)
Dhan	1300
Ranchi	1500
Jhr	700

Modification of the Database

→ Deletion (−)
→ Updation (U) (−, U)
→ Insertion (∪)

Deletion

r

A	B
1	2
3	4
5	6

Want to remove this tuple ←

$\sigma_{A=3}(r)$

A	B
3	4

$r \leftarrow r - \sigma_{A=3}(r)$

A	B
1	2
5	6

Insertion

To add (8, 9) in the above Table

$r \leftarrow r \cup \{('8', '9')\}$

Q. Delete all account records of Dhanbad Branch.

Ex. Account (AN, BN, BAL)

$account \leftarrow account - \sigma_{BN="Dhanbad"}(account)$

Q. Delete all loan records of range (0 to 50).

Ex. Loan (LN, BN, AMT)

$loan \leftarrow loan - \sigma_{AMT \geq 0 \ \wedge \ AMT \leq 50}(loan)$

Q. Delete all account at branches located in Dhanbad city.

Updation

1 Case - If you want to select some tuple from r & to update them you can use the following

$$r \leftarrow \Pi_{f_1, f_2 \ldots f_n}(\sigma_p(r)) \cup (r - \sigma_p(r))$$

Conditions (Predicate)

Integrity Constraints

→ Domain Constraints on Columns (Check)

Student

Admn. (number)	Name (Starts with 'a')	DoB (date)	Branch_id
			CSE / ECE / DA

Check if value is in Branch table or not

Referencing Relation

→ Referential integrity Constraint

Branch

BN	dept_id	Prog_id	Est
CSE			
ECE			
EE			

If value is present here

Referenced Relation

Branch_id is foreign key

→ Functional Dependencies

$X \longrightarrow Y$

∴ Y is functionally dependent on x

[$X \longrightarrow Y$ doesn't mean that, $Y \longrightarrow X$ will also be true.]

r

X	...	Y	...
1		5	
3		1	
3		5	
1		8	

Using the FD notation, we say that, 'K' is Superkey of 'R' if $K \longrightarrow R$, that is K is a superkey if whenever $t_1[K] = t_2[K]$ it is also the case that $t_1[R] = t_2[R]$, it means $t_1 = t_2$.

r(R)

$R = \{A, B, C, D, E\}$ (K = BC)

$BC \longrightarrow A, B, C, D, E$

If none B or C can uniquely define A, B, C, D, E then BC is candidate key (minimal Super key)

Closure of FDs

A set of all FDs logically implied by F is the closure of F.

F = set of FDs

F^+ ⇒ Closure ⇒ { f }

⇒ If α is set of attributes & β ⊆ α, then β is FDependent on α (Reflexivity) $\alpha \to \beta$

$\alpha = \{A, B, C, D\}$, $\beta = \{B, C\}$

$\alpha \to \beta$

⇒ If $\alpha \to \beta$, then $\gamma\alpha \to \gamma\beta$ (Augmentation)

⇒ If $\alpha \to \beta$, $\beta \to \gamma$, then $\alpha \to \gamma$ (Transitivity)

⇒ If $\alpha \to \beta$ and $\alpha \to \gamma$, then $\alpha \to \beta\gamma$ (Union)

⇒ If $\alpha \to \beta\gamma$, the $\alpha \to \beta$ & $\alpha \to \gamma$ (Decomposition)

⇒ If $\alpha \to \underline{\beta}$ and $\gamma\underline{\beta} \to \delta$, the $\alpha\gamma \to \delta$ (Pseudotransitivity)

Example:

$R = \{A, B, C, G, H, I\}$

$F = \{A \to B, A \to C, CG \to H, CG \to I, B \to H\}$

$F^+ = \{?\}$

1) $A \to H$ $[A \to B, B \to H]$
2) $AG \to H$ $[A \to C, CG \to H]$
3) $AG \to I$ $[A \to C, CG \to I]$
4) $CG \to HI$ $[CG \to H, CG \to I]$

$AG^+ = \{A, B, C, G, H, I\} = R$ [Candidate key]

$A^+ = \{A, B, C, H\} \neq R$

$G^+ = \{G\} \neq R$

Closure of Attribute set

α $\alpha^+ = ?$

```
result = α
while (change to result) do
    for each FD β → γ in F do
    begin
        if β ⊆ result then result = result ∪ γ
    end
```

Usage of Attr. Closure

1) Testing of Super Key
2) Testing FDs.

$A \to H$

$A^+ = \{A, B, C, H\}$ ✓

3) Computing closure of F

Canonical Cover

→ Remove the Redundant Dependencies from F.

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

FD = { $A \to B$, $B \to C$, $A \to C$ } → Redundant

→ Remove Extra Attributes

$AG \to CH$

↳ may be redundant

F_c = Canonical

$F_c = F$

Repeat
- use the union rule to replace any dependencies in F_c of the form $\alpha_1 \to \beta_1$ and $\alpha_1 \to \beta_2$ with $\alpha_1 \to \beta_1\beta_2$
- find a FD $\alpha \to \beta$ in F_c with an extraneous Attr either in α or β, then delete it from $\alpha \to \beta$

until F_c does not change.

Extraneous Attribute

1) If $A \in \beta$, to check if A is extraneous, consider the set

$F' = (F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - A)\}$

and check if $\alpha \to A$

$F = \{A \to B, B \to H, C \to D\}$, $F' = \{A \to B, B \to H, C \to D\}$

$A^+ = \{A, B, H\}$ not C

⇒ To check, compute α^+ under F'

⇒ If α^+ includes A, then A is extraneous in β

2) If $A \in \alpha$, to check if A is extra.

$\gamma = \alpha - \{A\}$ and check $\gamma \to \beta$ can be inferred from F.

γ^+ under F.

Finding no. of Candidate Keys

$R = \{A, B, C, D, E, F, G, H\}$

$FD = \{AB \to C, A \to DE, B \to F, F \to GH\}$

Dependency / Edge Diagonal

Now, Attribute which do not have incoming edge :-

⇒ A and B

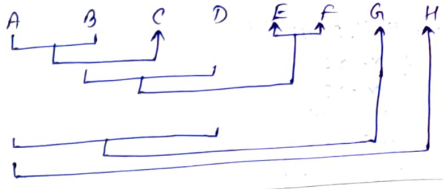
$(AB)^+ = \{A, B, C, D, E, F, G, H\} = R$ (Candidate Key)

$A^+ = \{A, D, E\} \neq R$

$B^+ = \{B, F, G, H\} \neq R$

Q. $R = \{A, B, C, D, E, F, G, H\}$

$F = \{AB \to C, BD \to EF, AD \to G, A \to H\}$



$(ABD)^+ = \{A, B, C, D, E, F, G, H\} = R$

Candidate Key.

Q. $R = \{A, B, C, D, E\}$

$F = \{BC \to AD, D \to E\}$

$C^+ = \{C\} \neq R$ (But an essential key and must be a part of CK)

$AC^+ = \{A, C\} \neq R$

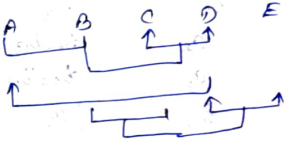
$BC^+ = \{A, B, C, D, E\} = R$ {CK}

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

$EC^+ = \{C, E\} \neq R$

Q. $R = \{A, B, C, D, E\}$

$F = \{AB \to CD, D \to A, BC \to DE\}$



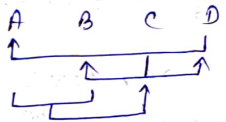
$B^+ = \{B\} \neq R$

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

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

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

Q. $R = \{A, B, C, D, E\}$, $F = \{D \to A, C \to BD, AB \to C\}$



⇒ No. essential Attr.

$A^+ \neq R$

$B^+ \neq R$

$C^+ = \{A, B, C, D\} = R$ ✓

$D^+ \neq R$

$AB^+ = \{A, B, C, D\} = R$ ✓

$BD^+ = \{B, D, A, C\} = R$ ✓

$AD^+ = \{A, D\}$

Lossless Join Algorithm

$R = \{X, Y, Z, W, Q\}$

$F = \{ X \to Z$
$Y \to Z$
$Z \to W$
$WQ \to Z$
$ZQ \to X \}$

Decomposition Set = $\{R_1(X,W), R_2(X,Y), R_3(Y,Q), R_4(Z,W,Q), R_5(X,Q)\}$

∴ whether this Decomp is lossless or not?

Step1:- Create a Table, as many ~~decomposed relations~~ Attributes (5) as many Columns.
Rows = no. of Decomposed Relations

	X (A_1)	Y (A_2)	Z (A_3)	W (A_4)	Q (A_5)
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{23} b_{13}	b_{24} a_4	b_{25}
R_3	b_{31} a_1	a_2	b_{33} b_{13} a_3	b_{34} a_4	a_5
R_4	b_{41} a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{53} b_{13} a_3	b_{54} a_4	a_5

for $X \to Z$, for same values of X we should have same Z. so for all a_1 in Row1 we should put same value of Z in R_3 as b_{13}.

for $Y \to Z$, Row 2 = Row 3 for same values of $a_2 \to b_{13}$

for $Z \to W$, for b_{13} & a_3 in Row 3 put same as in Row 4 (W).

for $WQ \to Z$,
R_3, a_4 a_5 → ~~b_{13}~~ a_3
R_4, a_4 a_5 → a_3
R_5, a_4 a_5 → ~~b_{13}~~ a_3

for $ZQ \to X$
R_3, a_3 a_5 → ~~b_{31}~~ a_1
R_4, a_3 a_5 → ~~b_{41}~~ a_1
R_5, a_3 a_5 → a_1

when we got a's & b's then replace b's with a's

After operations, if any Row has all a's value then Relation/Decomposition is Lossless.

Normalization

Indicators,
1) Redundant data
2) Insertion Anomalies
3) Updation Anomalies
4) Deletion Anomalies
5) NULL Handling

1) 1 NF → Domains are Atomic
2) 2 NF → 1NF + No Partial dependency
3) 3NF → 2NF + No Transitivity dependency
4) BCNF → 3NF + Determinant (LHS) should be candidate key
5) 4NF – BCNF + not more than 1 MVD
6) 5NF – 4NF + No Join Dependence (PJNF) (Project Join NF)

2NF

→ R is in 2NF ⇔ [(R should be in 1NF) ∧ (Not Contain any Partial dependency)]

⇒ $X \to Y$, is fully dependent, iff X is not a proper subset of candidate key or Y is a prime Attribute.
$[(X \not\subset CK) \vee (Y \text{ is PA})]$

⇒ $X \to Y$, is partially dependent ⇔ $[(Y \subset CK) \wedge (X \text{ is Not PA})]$

Q. $R = \{A, B, C, D\}$, $F = \{AB \to D, B \to C\}$

Soln $(AB)^+ = ABCD = R = CK$
$PA = \{A, B\}$, $NPA = \{C, D\}$

$AB \to D$, Fully dep. (CK)
$B \to C$, Partial dep.
↳ PA

3NF

R is in 3NF, iff $\forall$ fD_{NT} (for all fD which are non-Trivial)

$(X \to Y)\ [(X \text{ is Superkey}) \vee (Y \text{ is PA})]$

$\Rightarrow (X \to Y \text{ is Transitive Dep}) \Leftrightarrow (('X' \text{ is not a SK}) \wedge ('Y' \text{ is NPA})]$

Example:-

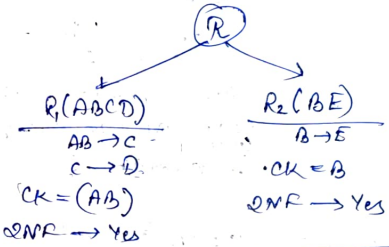
$R = \{A, B, C, D, E\}$, $F = \{AB \to C,\ C \to D,\ B \to E\}$

Sol: $AB^+ \to \{ABC, D, E\}$

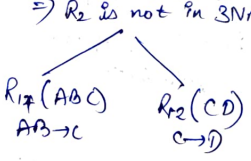
$AB \to C$, is in 3NF (CK)

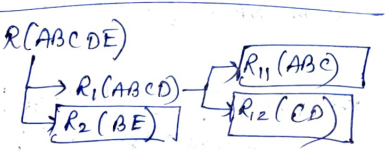
$B \to E$ ~~$B \to D$~~ → PA and D is not PA, Not in 2NF

∴ So relation is not in 2NF.



Because of $(C \to D)$,
⇒ R_2 is not in 3NF





⇒ R is in 3NF as well as in BCNF.

4NF → Remove all MVDependencies

Multivalued Dependency

$X_1 \to Y_1$, $X_1 \to Y_2$

$X \to\to Y$ (for same value of X we have multiple values of Y)

Three Necessary Conditions for MVD:—

1) $A \to\to B$, for a single value of A, more than 1 value of B exists.
2) Table/Relation should have atleast 3 columns
3) for a table with column A, B, C and there is MVD b/w A & B, then B and C should be independent of each other.

A	B	C
A_1	B_1	C_1
A_1	B_2	C_2

$A \to\to B$
$A \to\to C$

Ex:-

Admno.	Course	hobby
1	ADBMS	Chess
1	ADSA	Tennis
2	—	—

→

Adm	Course	Hobby
1	ADBMS	Tennis
1	ADSA	Chess

⇒ It will give two more rows in Table.
⇒ Its a bad design, because there is no relation b/w Course and hobby.

Decomposition → R_1(Adm, Course)
R_2(Adm, hobby)

⇒ MVD occures if two or more independent relation are kept in a single relation.

$\bar{A} \to\to \bar{B}$

$\forall\ t, u \in R$: $t[\bar{A}] = u[\bar{A}]$ then
$\exists\ v \in R$: $v[\bar{A}] = t[\bar{A}]$ and $v[\bar{B}] = t[\bar{B}]$ and $v[\text{rest}] = u[\text{rest}]$

	$\overline{A}$	$\overline{B}$	rest
t	$\overline{a}$	$\overline{b_1}$	$\overline{x_1}$
u	$\overline{a}$	$\overline{b_2}$	$\overline{x_2}$
v	$\overline{a}$	$\overline{b_1}$	$\overline{x_2}$
w	$\overline{a}$	$\overline{b_2}$	$\overline{x_1}$

MVD are also know as Tuple generating Dependencies.

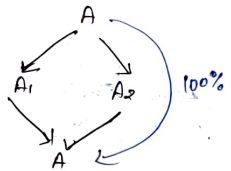
Admno	Phone	email
23	1	c1
~~23~~	1	c2
~~23~~	1	c3
~~23~~	2	c4
	2	

Adm	Phone
x	P1
x	P2
x	P3
x	P4
x	P5

Adm	email
x	c1
x	c2
x	c3
x	c4
x	c5

$A \twoheadrightarrow B$, $A \twoheadrightarrow C$

MVD



100%

Join Dependency

Eg:- R

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

R_1 (A,B)

A	B
1	2
2	2
3	3

R_2 (B,C)

B	C
2	1
2	2
3	2

Join

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

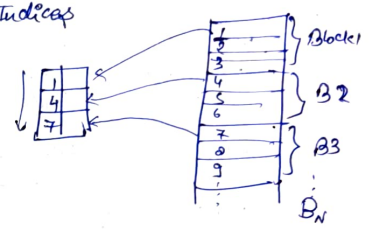
eg: PJNF/5NF says choose common attribute as key or part of key like A in this example.

→ Common Attn. should be Candidate or Super key of either relations (R, R_1, R_2), then Join will be Lossless.

Indexing

→ Ordered Indices
→ Hashed Indices

Ordered :-



1) Access Time :- Type specific value, Range of Values (2 to 200)
2) Access Time :- Individual access time taken.
3) Insertion Time :- Time to insert in DB (indexed). Making Index Table up to the mark.
4) Deletion Time :- After deleting record, updation of indexes.
5) Space Overhead :- Space/Memory to store indexes.

Ordered Indices

→ Dense Index
→ Sparse Index

All keys are Present

Key	Pointer
B	A1
D	A2
M	A4
P	A5
R	A8
U	A10

Dense

Key	Pointer
B	A1
M	A4
R	A8

Sparse

AN	BN	Bal
A1	B	–
A2	D	–
A3	D	–
A4	M	–
A5	P	–
A6	P	–
A7	P	–
A8	R	–
A9	R	–
A10	U	–

∴ Pointer will point to the 1st Occurrence of key.

⇒ Trade-off B/w both results into dividing the Relation into Blocks and store first value of each block in the Sparse Index table.

Types of Indices

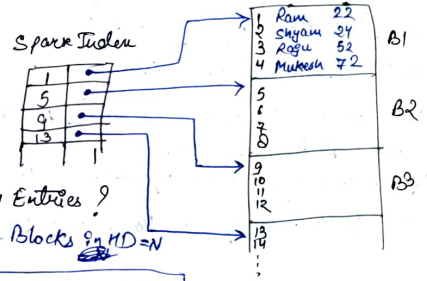
- → Primary
- → Clustered
- → Secondary

Ordered Type (Sorted)

Unordered Type

Primary Index	Clustered Index
1,2,3,4,5,6...	1,1,2,3,3,3,4...
Secondary Index	Secondary Index
4,8,1,2,6,9,3...	3,1,3,2,1,6,1,8...
Key (only Unique values)	Non-Key

Primary Index



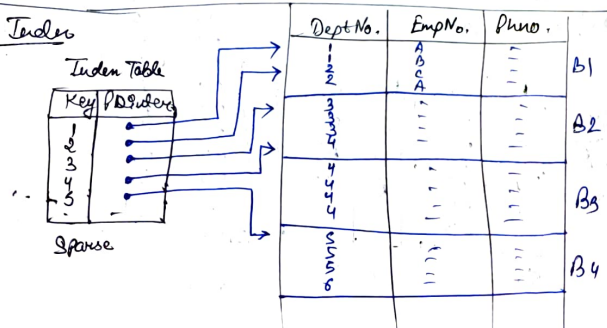
Q. How many Entries ?

Ans. Number of Blocks in HD = N

Search Time = $\log_2 N + 1$, where N = no. of Blocks in Index Table.

⇒ $O(\log_2 N)$

Clustered Index



⇒ When searching is req. for a value then, we require to access two Blocks (B2 & B3) of HD and this is called Block Hanker.

Search Time = $\log_2 N + 1 + 1$

- $\log_2 N$ → for Index Table (Index Tree)
- 1 → HD search
- 1 → for another Block (May search more than 1 Block due to Block Hanker)

Secondary Index

for Unordered + Non-Key (PAN)

SI

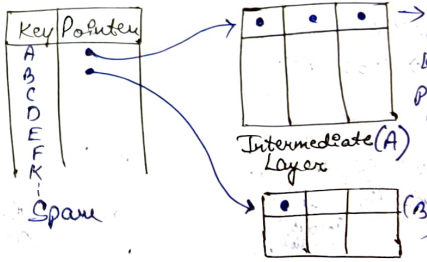
Key	Pointer
23	
33	
40	
51	
62	
71	
82	
91	
100	

Dense Index Table

Empid	Name	PAN
1	A	40
2	B	51
3	C	62
4	A	33
5	A	71
6	D	82
7	E	91
8	F	23
9	K	100
10	L	120
11	M	130
12	C	195
Ordered (Key)	Unordered (Non-Key)	Unor- (Key)

Search Time = $\log_2 N + 1$

for Unordered + Non-Key (Name)



→ For multiple value, we will create a Intermediate layer and each column will point to all locations of value in the HD.

Overall Index = Dense

Search Time ⇒ $\log_2 N + 1 + 1$

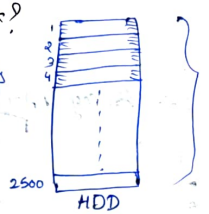
- $\log_2 N$ → Main IT
- 1 → Intermediate Layer
- 1 → HD Access

Ex:- Find out the avg Time Comp. to search a record from the HD. Given the following:-

Total Records = 10,000
Record Size = 250 Bytes / Record
Block Size = 1000 Bytes / Block
Data entered in HDD without any order (Unordered)

1) What is the no. of Records/Block?

Ans. $\frac{\text{Block Size}}{\text{Record Size}} = \frac{1000}{250} = 4$ Records



No. of Blocks = $\frac{10000}{4} \Rightarrow 2500$

2) Sorted Data

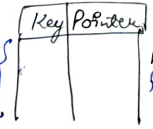
Sol- Binary Search (Indexing)
Best Case = O(1)
Worst Case = $O(\log_2 N) \Rightarrow \log(2500)$
$\Rightarrow$ 12 Blocks

3) Unsorted Data

Sol- Linear Search
Best Case = O(1)
Worst Case = O(N)
= 2500 Blocks
Average Case = $O(N/2) \Rightarrow$ 1250 Blocks

4) Sorted Data + Sparse Indexing.

Sol- How many no. of Entries in SI?
$\Rightarrow$ 2500 (same as no. of Blocks)



Key = 10 Bytes
Pointer = 10 Bytes
Record = 10 + 10 = 20 Bytes

One Block of Index Table $\Rightarrow$ 1000 Bytes
One entry Size of Index T. $\Rightarrow$ 20 Bytes
No. of entries in 1 Block of IT $\Rightarrow \frac{1000}{20} \Rightarrow 50$

No. of Blocks in IT $\Rightarrow \frac{2500}{50} \Rightarrow$ 50 Blocks

Total Search Time $\leq \log_2 n + 1$
$\Rightarrow \lceil \log_2(50) \rceil + 1$
$\Rightarrow 7$

5) UnSorted + Dense Indexing

No. of Blocks in IT $\Rightarrow \frac{10000}{50} \Rightarrow$ 200 Blocks

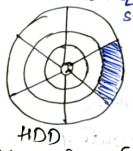
TST $\Rightarrow \log_2(200) + 1$
$\Rightarrow 9$

Roll list (10) (30) (10) (8) (50) $\Rightarrow$ 128 Bytes

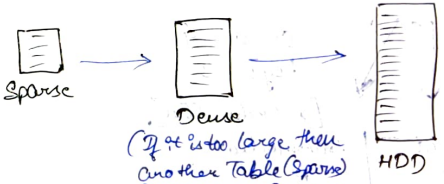
Admno.	name	Prog.	Branch	Dept
...				

100 Records $\rightarrow$ 25 Blocks of HDD
& req to store

1 Record Size = 128 Bytes



Block Size = 512 Bytes



Index Updation

↳ Insertion

$\Rightarrow$ Insert a Record to HD
$\Rightarrow$ If data is already present then go to the pointer & to the last } Dense
repetition and insert the record.
$\Rightarrow$ If not then insert and add entry to IT.

⇒ Sparse

→ Entry is present then add to HDD
→ Not present, then Add to HDD & then in Index Table
→ If HDD full, then Create Block.

↳ Deletion

⇒ First go to the pointer, then delete all repetitions & then Delete entry from Index Table. } Dense

For Sparse

⇒ Search in the HDD & delete the Repetitions.
⇒ If the entry of is present in Sparse IT then, after deleting increment the Sparse entry to the next Element.

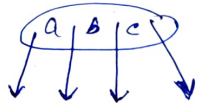
Multilevel Searching

B Tree | B+ Tree

→ Balanced
→ Multiway Search
→ Random

B+ Tree

→ Balanced
→ Multiway Search
→ Random & Sequential Search



Order = 4
Keys = 3

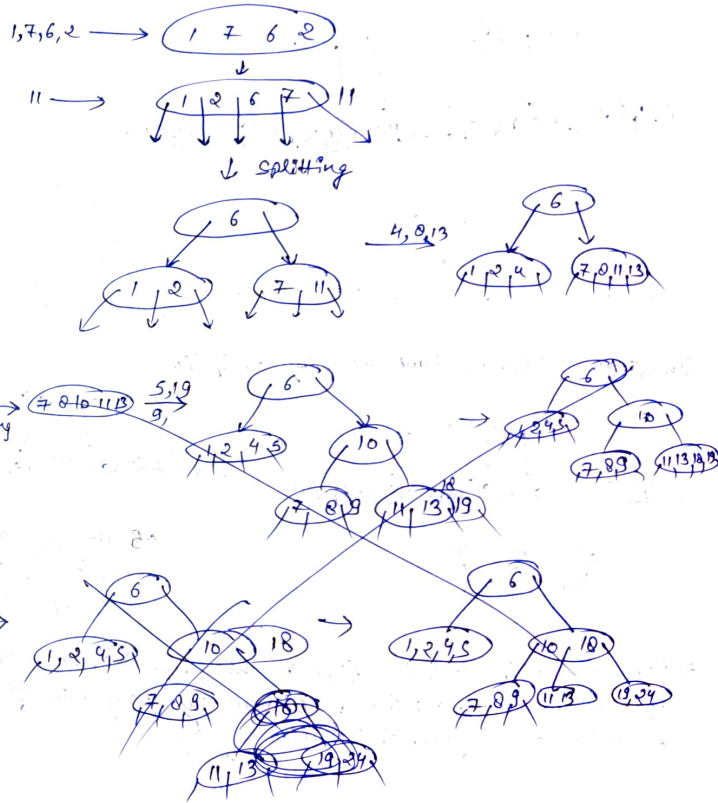
P-Order B Tree

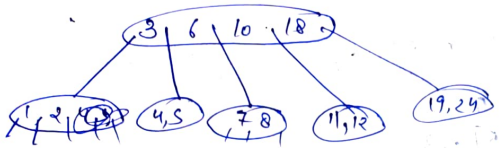
Children	Root	Intermediate
max	p	p
min	2	$\lceil \frac{p}{2} \rceil$

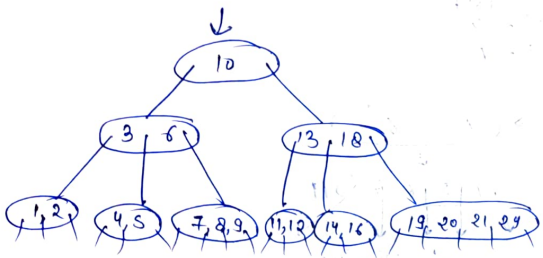
Order = 5

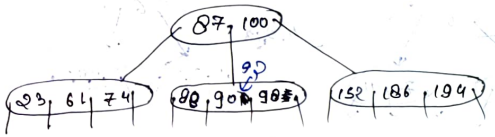
1, 7, 6, 2, 11, 4, 8, 13, 10, 5, 19, 9, 18, 24, 3, 12, 14, 20, 21, 16

max Children = 5 | 5
Min Children = 2 | $\lceil \frac{5}{2} \rceil \Rightarrow 3$
max keys = 5-1 = 4
min keys = 2-1 = 1





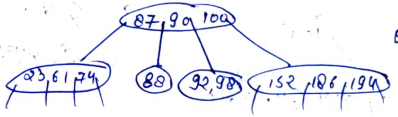




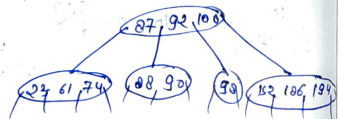
Order = 4

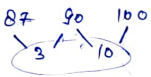
	Root	Int. Node
Max Children	4	4
Min	2	$\lceil \frac{4}{2} \rceil = 2$

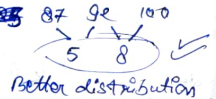
Insert 92 → Then we have two ways to Split



or







Better distribution

⇒ Thus we would select left bias in this case. Since, it more nearly equalizes the probability of a new key going into left & right, Assuming a uniform distribution of keys.

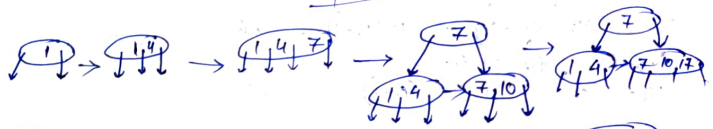
B⁺ Tree :-

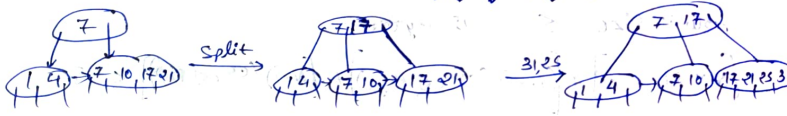
→ Repitition of some keys.
→ All queries are resolved after going to the leaf and all leaf are connected. (Sequential Search).

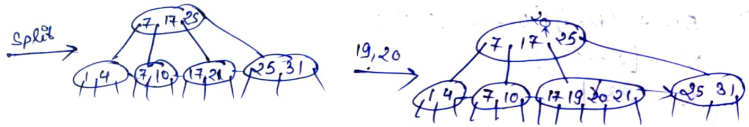
Insertion in B⁺ Tree

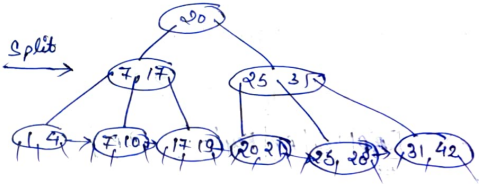
1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42 (Order = 4)

			key
Max	4	4	3
Min	2	2	1









Q Consider a B Tree of Key Size = 10 Bytes, Block Size = 512 Bytes, Data Pointer Size = 8 bytes, Block Pointer Size = 5 bytes. find the Order of BTree.

Sol:

Let no. of Block Pointer = N

Total Size of Block Pointers = $5N$ → (1)

Number of Keys ⇒ $N-1$

Total size of Keys ⇒ $10(N-1)$ → (2)

Number of DP ⇒ $N-1$

Total size of DP ⇒ $8(N-1)$ → (3)

Block Size ≤ 512 Bytes

$$[N \times \text{Size of BP}] + [(N-1)(\text{Size of Keys})] + [N-1)(\text{Size of DP})] \leq 512$$

$$\Rightarrow 5N + (N-1)(10+8) \leq 512$$

$$\Rightarrow N = \lfloor 23.04 \rfloor$$

$$N = 23$$

Q Consider B^+ Tree,

Key Size = 10 Bytes, Block Size = 512 B, Data Psize = 8 bytes, BP Size = 5 Bytes.

What is the Order of Leaf & non-Leaf nodes?

Sol: for non-leaf Node ⇒

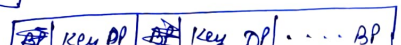
$5N + (N-1)10 \leq 512$

$15N \leq 522$

$N = 34.8$

$N = 34$

for leaf ⇒

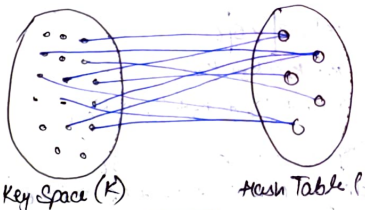


$5 + (N-1)(10+8) \leq 512$

$N = 28$

Hashing

↳ Hashed Indices



$h(K) : K \rightarrow I$

(Hash Function)

Hash function

10, 19, 35, 43, 62, 59, 31, 49, 77, 33

$h(K)$ = sum of digits.

$10 \rightarrow (1+0) \% 10 = 1$

$19 \rightarrow (1+9) \% 10 = 10$

$h(K) = \left(\sum_{n=0}^{n} d_i\right) \% 10$

0	19
1	10
2	
3	49
4	59, 31, 77
5	
6	33
7	43
8	35, 62
9	

Collisions

Most Used Hash function

1) Division Method :-

$H(K) = K \bmod h$

Index 0 → $H(K) = K \% h$

Index 1 → $H(K) = K \% h + 1$

h = no. of spaces in Hash table

2) Mid Square Method :-

$K = 1234$

$K^2 = 1522756$

Store at (525)

3. Folding Method

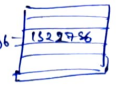
K = 15 | 22 | 75 | 6

K_1 K_2 K_3 K_4

$\Rightarrow K_1 + K_2 + K_3 + K_4 \Rightarrow R_{sum}$

Chopping $\Rightarrow$ 01 52 27 56

Pure folding $\Rightarrow$ 01 + 52 + 27 + 56 $\Rightarrow$ 136



fold shifting $\Rightarrow$ 10 + 52 + 72 + 56 $\Rightarrow$ 190

fold Boundary $\Rightarrow$ 10 + 52 + 27 + 65 $\Rightarrow$ 154
(first & last no. folded)

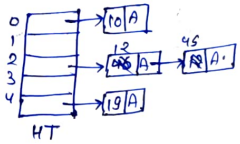
Collision Resolution Technique

Open Hashing (Chaining)

42 | 19 | 10 | 12

$h(K) = K \% 5$

42 % 5 = 2
19 % 5 = 4
10 % 5 = 0
12 % 5 = 2



$\Rightarrow$ Insert at front of linked list
$\Rightarrow$ Disadvantages, if all keys mapped to same location (Extra space).

Disadvantage $\Rightarrow$ Primary Clustering
$\Rightarrow$ Secondary Clustering
$\Rightarrow$ Linear time (O(n)), if most spaces are filled
$\Rightarrow$ Deletion Difficulty

Advantages
$\Rightarrow$ No Extra Space

Closed Hashing (Open Addressing)

Linear Probing

$h(K) = K \bmod n$

$h'(K, i) = (h(K) + i) \bmod n$

(in case of Collision)

i = collision/probe number.

43 | 135 | 72 | 23 | 99 | 19 | 82

n = 10

$h(K) = K \bmod 10$

$h(K, i) = (h(K) + i) \bmod 10$

for, 23

$h(K, i) = (h(23) + i) \% 10$
$\Rightarrow 4$

for 19,

$h(19, i) = (h(19) + i) \% 10$
$\Rightarrow 20 \% 10$
$\Rightarrow 0$

Same for 82,

Disadvantages

$\Rightarrow$ If two or more key are following same probing sequence, thats called Secondary Clustering.
$\Rightarrow$ Primary Clustering
$\Rightarrow$ In worst case searching time is O(n)
$\Rightarrow$ Deletion is difficult.

Advantages

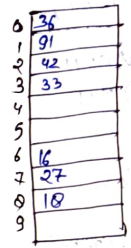
$\Rightarrow$ No Extra Space.

Quadratic Probing

$h(K) = K \bmod n$

$h(K, i) = (h(K) + i^2) \bmod n.$

42 | 16 | 91 | 33 | 18 | 27 | 36 | 62



for 36,

1) $(h(36) + 1^2) \% 10 \Rightarrow 7$, Collision
2) $(h(36) + 2^2) \% 10 = 0$

for 62,

1) $(h(62) + 1^2) \% 10 \Rightarrow 3$, Collision
2) $(h(62) + 2^2) \% 10 \Rightarrow 6$, Collision
3) $(h(62) + 4^2) \% 10 \Rightarrow 2$, Collision

!

So on

Not getting space for some keys, even if space are there, This is biggest disadvantage.

Disadvantages:

$\Rightarrow$ Search O(n)
$\Rightarrow$ Secondary Clustering
$\Rightarrow$ No ~~guarantee~~ guaranteed of space in HT

Advantages

1) No extra Space
2) Primary clustering is not here.

Double Hashing (Rehashing)

$h_1(K) = K \bmod 11$

$h_2(K) = 8 - (K \bmod 8)$

$[h_1(K) + i\,h_2(K)] \bmod 11$

20	34	45	70	56

0	
1	34
2	
3	56
4	45
5	
6	70
7	
8	
9	20
10	

for 45, (collision)

$h_1(K) = 1$

$h_2(K) = 8 - (45 \% 8)$

$\Rightarrow 3$

$\Rightarrow (h_1(K) + i\,h_2(K)) \% 11$

$\Rightarrow (1 + (1)(3)) \% 11$

$\Rightarrow 4$

for 70, Collision

$h_1(K) = 4$

$h_2(K) = 8 - (70 \% 8) \Rightarrow 2$

$(4 + 1(2)) \% 11 \Rightarrow 6$

for 56, Collision

$h_1(K) = 1$

$h_2(K) = 8 - (56 \% 8) \Rightarrow 8$

$(1 + 1\cdot8) \% 11 \Rightarrow 9$, Collision

$(1 + 2\times8) \% 11 \Rightarrow 6$, Collision

$(1 + 3\times8) \% 11 \Rightarrow 3$,

Advantages

→ No Primary & Secondary Clustering.

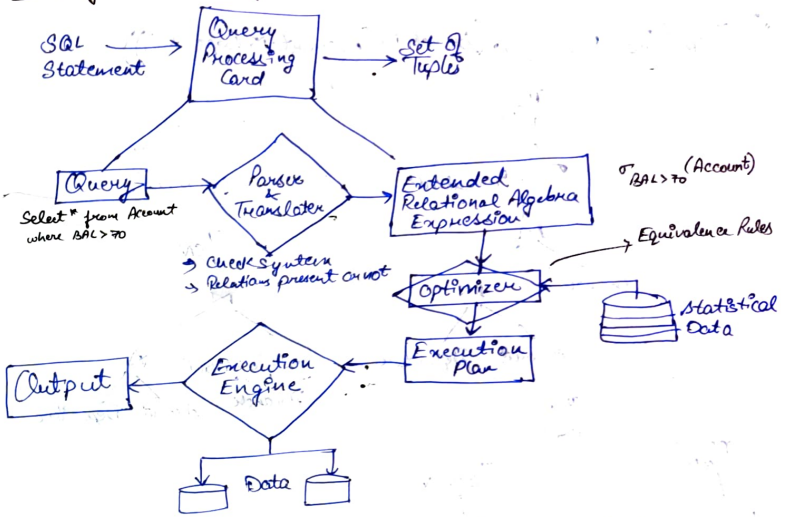
→ Always get space, if present

→ No extra Space

Disadvantages

→ ~~No~~ Linear time $O(n)$ Searching.

Query Processing & Optimization



Account

AN	BN	BAL
A1	A	2000
A2	B	3000
A3	A	6000
A4	B	7000
A5	C	1000

π_{BAL} | $\sigma_{BAL>2000}$ | Account

AN	BN	BAL
A2	B	3000
A3	A	6000
A4	B	7000

$\sigma_{BAL>2000}$ | π_{BAL} | Account

BAL
3000
6000
7000

More Optimized

Account

AN	BN	BAL

→ Bytes/Record

→ How many Records

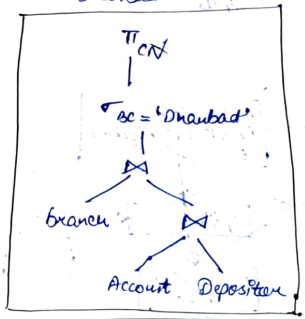
→ How many Blocks?

→ Records/Block

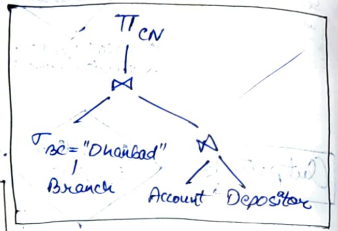
→ (AN, Account) = Change Value Count

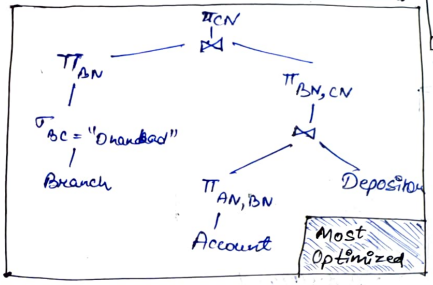
Q- find the names of all Customers who have an account at any branch located in Dhanbad.

Sol: $\Pi_{CN}(\sigma_{BC = \text{'Dhanbad'}}(\text{branch} \bowtie (\text{Account} \bowtie \text{Depositor})))$



Branch(BN, BC, Assets)
Account(AN, BN, BAL)
Depositor(CN, AN)





Selection Operation (σ)

→ file scanning $[\sigma_p(x)]$

Basic Algorithms

↳ Linear Search (A1)
↳ Binary Search (A2)

Selection Using Indices

↳ Primary Index, Equality on Key
↳ Primary Index, Equality on Non-Key
↳ Secondary Index, Equality on Key
↳ Secondary Index, Equality on Non-Key.

Selection Involving Comparision

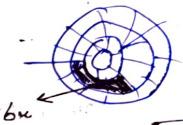
→ Primary Index Comparision.
→ Secondary Index Comparision

More Complex Selection Predicate

→ Conjunction (Series of AND (∧))
→ Disjunction (Series of OR (∨))
→ Negation (NOT)

Cost Involvment

→ Disk Scanning Time (BST) → Seek Time
→ Block Transfer Time (TB) →



Let, n blocks to be transferred,

$Cost_{LS} = TS + b_x.TB$ (Linear Search) → A1

$Cost_{Avg} = TS + \frac{b_x * TB}{2}$ (Equality on Key)

> If the file is ordered on an Attribute and selection Condition is Equality Comparison on the Attribute.

$Cost_{BS} = \lceil \log(b_x) \rceil * (TS + TB)$ (Binary Search) → A2

Selection Operation

A1 : Linear Search [Cost = $TS + b_x TB$]
A2 : Binary Search [Cost = $\lceil \log_2(b_x) \rceil * (TS + TB)$]
A3 : Primary Index with Equality on Key
~~Student record~~ (B^+ Tree) $Cost_{BS} = [TS + b_x T_B + T_S + T_B]$
$Cost_{BS} = [\lceil \log_2(b_x) \rceil + T_S + T_B]$
[Cost = $(h_i + 1) * (T_S + T_B)$]
A4 : Primary Index (Non-key) [Cost = $h_i(T_S + T_B) + T_S + n * T_B$]
A5 : Secondary I (Key) [Cost = $(T_S + T_B)(h_i + 1)$] (Same as A3)
A6 : SI with Non-Key [Cost = $h_i(T_S + T_B) + n(T_S + T_B)$] [Key is on n multiple Loc]

Index Table

AdmNo.

Key	Address
1	a_1
2	a_2
3	a_3
4	a_4
5	a_5
6	a_6
7	a_7
8	a_8
9	a_9
10	a_{10}

Dense Index

Student Record

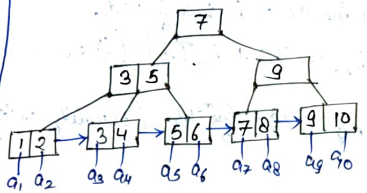
Address	Admno.	Name	Marks	
a_1	1	Amit	20	b_1
a_2	2	Shyam	40	
a_3	3	Ram	31	
a_4	4	Robin	69	b_2
a_5	5	Risabh	82	
a_6	6	Anita	73	
a_7	7	Prakash	43	b_3
a_8	8	Meera	52	
a_9	9	Rowan	39	
a_{10}	10	Manoj	28	b_4

$\sigma_{Admno.=7}$ (Student Record)

$Cost = h_i(T_S + T_B) + (T_S + T_B)$

$Cost = (h_i + 1)(T_S + T_B)$

B+ Tree



$Cost_{BS} = \log_2(b_x)(t_S + T_B) + (t_S + T_B)$

$Cost_{IS} = (h_i)(T_S + b_x T_B) + (T_S + T_B)$

Selection Using Comparision

Algo A4 (Ordered)

The SI provides pointers to the Records but to get actual Record we have to fetch the records by using Pointers.

I. $A > V$ or $A \geq V$

$\sigma_{Admno. > 5}$ (Student Record)

$\sigma_{Admno. \geq 5}$ (SR)

$Cost = h_i(T_S + T_B) + (T_S + T_B)$

No. of Blocks to be moved

II. $A < V$ or $A \leq V$

$Cost = T_S + n T_B$

for secondary Index, B+ Tree, Comparision Unordered

$Cost = h_i(T_S + T_B) + n(T_S + T_B)$

⇒ The lowest level index are scanned either from the smallest value upto V or from v up to max. value

A6 and A7

More Complex Selection Predicate

A8 : Conjuctive Selection using one Index

$\sigma_{\theta_1 \wedge \theta_2 \wedge \theta_3 \wedge \ldots \wedge \theta_n}(r)$

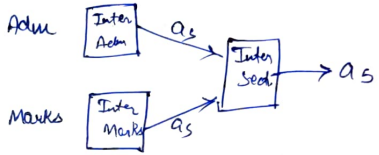
Select * from student_Record where Adm > 5 and Adm < 8

⇒ After Applying Adm > 5, then second Cond. will only be applied on the resulting data (Attribute must be same).

A9 : Conjunctive Selection using Composite Index

Select * from SR where Adm = 5 and marks = 82

A10 : CS by Intersection of Identifier



Marks	Address
20	a_1
28	a_{10}
31	a_3
39	a_9
40	a_2
43	a_7
52	a_8
69	a_4
73	a_6
82	a_5

A11 : ~~Conj~~ Disjunctive Selection by union of Identifier

$\sigma_{\theta_1 \vee \theta_2 \vee \theta_3 \vee \ldots \vee \theta_n}(r)$

A12 : Negation ($\neg$)

Join Operation

$r \bowtie s$

↑ Outer Relation ↑ Inner Relation

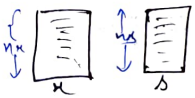
$r \bowtie_{r.A=s.B} s$ → Equi Join

Depositor ⋈ Customer
(AN, CN) (CN, CS, CC)

Number of Records in depositor = 5000 & Customers = 10,000
" " Blocks " " = 100 & " = 400

Nested Loop Join

```
for each tuple tr in 'r' do begin
    for each tuple ts in 's' do begin
        test pair (tr, ts) to see if they satisfy the Join Condition
            - If they do, add (tr, ts) to the result
    end
end
```



Comparision = $n_r \times n_s$

Worst Condition

⇒ 1 Block of each is allowed at a time
⇒ Total no. of Block Transfer = $b_r + n_r b_s$ = 400100
⇒ Total seeks = $b_r + n_r$ = 5100

Best Case

⇒ Block Transfer = $b_r + b_s$ [Bring all Blocks of r & s into memory
⇒ Seek Time = 1+1 ⇒ 2

Block Nested Loop Join :-

```
for each Block Br of 'r' do Begin
    for each Block Bs of 's' do Begin
        for each tuple tr in 'r'
            for each tuple ts in 's'
                test pair (tr, ts) to see if they satisfy the Join Condition
                    If they do, add (tr, ts) to the result
        end
    end
end
```


Worst Case

Total BT = $b_r + b_r * b_s$ = 40100
Total ST = $b_r + b_r$ = 200

Best Case

BT = $b_r + b_s$
ST = 1+1 = 2

Improvements

If the Join Attribute form key on the inner relation, then for each outer relation tuple, the inner loop can terminate as soon as the first match is found.

Instead of using disk blocks as the blocking unit for outer Relati we can use the biggest size that can fit in memory, while leavin enough space for the buffers of the inner relation & outer.

Memory has m blocks.
Read (m-2) blocks of outer relation at a time.
Each block of inner relation will Join with (m-2) blocks of outer relation.
Reduces the no. of scan of inner relation from $b_r \rightarrow \lceil \frac{b_r}{m-2} \rceil$

$$\text{Total Block Transfer} = b_r + \left\lceil \frac{b_r}{m-2} \right\rceil * b_s$$

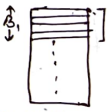
$$\text{Total Seek Time} = 2 \left\lceil \frac{b_r}{m-2} \right\rceil$$

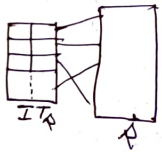
Inner loop can be scanned alternatively forward & Backward. This scanning method alters the request for disk blocks so that the data remaining in the buffer from the previous scan can be reused. Thus, the reducing the no. of disk accessing.

Indexed Nested-Loop Joins

Total Cost = $b_r (t_s + t_p) + n_r * c$

where, c = The Cost of single selection on S using join cond^n r





Merge Join

a	3	A
b	1	G
d	8	N
d	13	N
m	5	B

a_1	a_2
a	3
b	1
d	8
d	13
f	7
m	5
z	6

P_R →

a_1	a_2
a	A
b	G
c	L
d	N
m	B

← P_S

1) It associates 1 pointer with each relation.
2) Pointer points to the initially tuple of resp. relation.
3) Pointer value will be incremented only if there is an unequal occurences.

Cost = $b_R + b_S$ Seek time = 2

4) If either of the input relation is unsorted on a Join Attribute, they can be sorted first before applying the merge Join Algorithm.

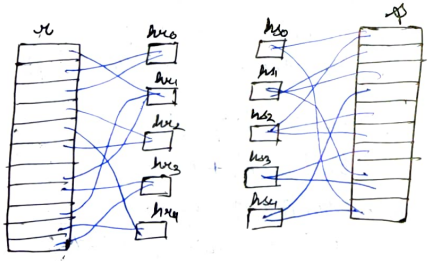
Seek time(worst case) = $\left\lceil \frac{b_R}{b_b} \right\rceil + \left\lceil \frac{b_S}{b_b} \right\rceil$ (Total Block Cap = b_b in m/m)

if $b_R = b_S = b_b$,

ST = 2

Hash Join

1) Partition the record.
2) One by one take partition
3) Build the Hashed index & build input.
4) Probe the tuples from other relation.



Cost = $(b_R + b_S)$ + $(b_R + b_S)$ + $(b_R + b_S)$

[Loading all blocks of R & S in m/m for Hashing/Partitioning] [Back to HD after Partitioning] [for Build and Probe]

Seek time = $2(b_R + b_S) + 2n_h$ [n_h = no. of Hash partition of the relation]

Ex:

R

E_id	Name	Salary	D_id
10	P	1000	1
11	Q	2000	1
12	R	3000	1
13	S	4000	2
14	T	5000	2
15	U	6000	3
16	V	7000	3
17	W	8000	3
18	X	9000	4

S

D_id	Name
1	A
2	B
3	C
4	D
5	E

We will apply Hash Join,

$H(x) = (D\_id \% 3)$

h_{R_0}

E_id	Name	Sal	D_id
15	U	6000	3
16	V	7000	3
17	W	8000	3

h_{R_1}

E_id	Name	Sal	D_id
10	P	1000	1
11	Q	2000	1
12	R	3000	1
18	X	9000	4

h_{R_2}

E_id	Name	Sal	D_id
13	S	4000	2
14	T	5000	2

Probe Relation

h_{S_0}

D_id	Name
3	C

→ Bucket 0

→ Bucket 1

h_{S_1}

D_id	Name
1	A
4	D

→ Bucket 0

→ Bucket 1

h_{S_2}

D_id	Name
2	B
5	E

→ Bucket 0

→ Bucket 1

Build Relation (Keep in memory)

~~Partitioning~~

Sorting Operation

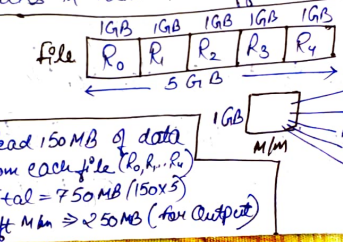
→ Internal Sorting: If entire relation can be accomodate into m/m and sorting is applied. (Heap, Quick, merge, etc.)

→ External Sorting: If the relation is large and can't fit into m/m.

External sort Merge

Let M denote the no. of blocks in main m/m Buffer available for sorting.

M = 1 GB
file size = 5GB



Read 150 MB of data from each file ($R_0, R_1 .. R_4$)
Total = 750 MB (150 x 5)
Left M/m ⇒ 250 MB (for Output)

Ex:-

S_id	Name
1	Ravi
2	Amit
3	Pawan
4	Magan
5	Neha
6	Rahul
7	Abhishek
8	Rajesh
9	Santosh
10	Jaya
11	Monu

M=3

R_0

2	Amit
3	Pawan
1	Ravi

R_1

6	Rahul
4	Magan
5	Neha

R_2

7	Abhishek
8	Rajesh
9	Santosh

R_3

10	Jaya
11	Monu

2	Amit
6	Rahul
4	Magan
5	Neha
3	Pawan
1	Ravi

→ Back to disk

7	Abhishek
10	Jaya
11	Monu
8	Rajesh
9	Santosh

7	Abhishek
2	Amit
10	Jaya
11	Monu
6	Rahul
8	Rajesh
4	Magan
5	Neha
3	Pawan
1	Ravi
9	Santosh

Total Cost $= b_R(2\lceil \log_{M-1}(b_R/M)+1 \rceil$
where, M = no. of runs

Total Seek Time $= 2\lceil b_R/M \rceil + \lceil b_R/b_b \rceil (2\lceil \log_{M-1}(b_R/M) \rceil - 1$

Query Processing & Optimization

SQL → Ext. Rel. Algebra → [Various Execution Plan → Best Execution Plans] (Catalog Information)

Equivalence Rule

1) Conjunctive selection Op^n:- can be deconstructive into a sequence of individual selection.

$$\sigma_{\theta_1 \wedge \theta_2}(E) \equiv \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2) Selection Op^n are Commutative:-

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3) $\pi_{L_1}(\pi_{L_2}(\pi \ldots \pi_{L_n}(E))\ldots) = \pi_{L_1}(E)$

4) Cartesian Combination:- $\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$

5) $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$

Theta Join Op^n are Commutative:-

$$E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$$

Associative (⊗ Join):-

$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$ → Natural Join

$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$ → Theta Join

(Possible iff θ_2 involves attr. only from E_2 & E_3)

7) Selection Op^n distributes over θ Join Op^n under two conditions:-

a) when all Attr in θ_0 involve only the Attr. of one of the Exp(E_i) being Join.

$$\sigma_{\theta_0}(E_1 \bowtie_\theta E_2) = \sigma_{\theta_0}(E_1) \bowtie_\theta E_2$$

b) when θ_1 involve only Attr of E_1 & θ_2 of E_2 and if you have this,

$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie E_2) = \sigma_{\theta_1}(E_1) \bowtie \sigma_{\theta_2}(E_2)$$

8) Projection Op^n distributes over theta Join:-

a) If θ involves only L_1 & L_2 Attr. (θ from $L_1 \cup L_2$)

$$\pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \pi_{L_1}(E_1) \bowtie_\theta \pi_{L_2}(E_2)$$

b) $E_1 \bowtie_\theta E_2$

L_1 = Set of Attr. of E_1
L_2 = " " " E_2
L_3 = " " " of E_3 that involves in θ but are not in $L_1 \cup L_2$.

$E_1 = (A, B, C, D, E)$, $L_1 = (ABC)$
$E_2 = (B, C, D, F, Q)$, $L_2 = (BCD)$
$L_3 = (EF)$

L_4 = " " " E_1 that involves in θ but not in $L_1 \cup L_2$

$$\pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \pi_{L_1 \cup L_2}(\pi_{L_1 \cup L_3}(E_1) \bowtie_\theta \pi_{L_2 \cup L_4}(E_2))$$

9- Union & Intersection are commutative:- (44)

$E_1 \cup E_2 = E_2 \cup E_1$

$E_1 \cap E_2 = E_2 \cap E_1$

10) Union & Intersection are Associative:-

$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$

$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$

11) Selection opⁿ distributes over Union, Intersection & Set difference

σ ∪ ∩ &

$\sigma_\theta (E_1 - E_2) = \sigma_\theta (E_1) - \sigma_\theta (E_2)$ ← ∪ & ∩

$\sigma_\theta (E_1 - E_2) = \sigma_\theta (E_1) - E_2$ ← ~~X~~ & ∉ ∩

12) Project opⁿ distributes over Union:-

$\pi_{L_1} (E_1 \cup E_2) = \pi_2 (E_1) \cup \pi_2 (E_2)$

13) Join Ordering:-

$(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$

Which order to prefer?

If $r_2 \bowtie r_3$ is quite large & $r_1 \bowtie r_2$ is small, then LHS to choose

Statistical information for Cost Estimation

Catalog Information ↓↓↓

I- for each relation

n_r — number of tuples in 'r'

b_r — # of blocks containing tuples of 'r'

l_r — size of a tuple of 'r'

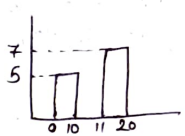
f_r — blocking factor - No. of tuples of 'r' that fit into 1 block.

$f_r = \left\lceil \frac{n_r}{b_r} \right\rceil$ $\quad$ $b_r = \frac{n_r}{f_r}$

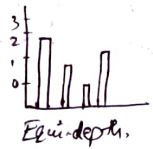
II. for each Attn of a relation:- (45) $r(A,B,C,D)$

$V(A,r)$ = # of distinct values that appear in relation r for A (Attn.)
($\Pi_A(r)$)

min (A, r)
max (A, r)
Histogram → Equi-width
Histogram → Equi-depth



Equi-width



Equi-depth.

⇒ Equi-width divides the range of values into equal size range

⇒ Equi-depth adjust the boundaries of the range i.e. each range has same # of value

III. Index (B^+ Tree):-

↳ Height

↳ # of Leaf Tree Nodes

↳ Order of Tree

Statistical Information for Cost Estimation

n_r — no. of tuples in 'r'

b_r — no. of blocks containing tuples of 'r'

l_r — size of a tuple of 'r'

f_r — blocking factor

$V(A,r)$ — No. of distinct values that appears in 'r' for Attn. A

$b_r = \left\lceil \frac{n_r}{f_r} \right\rceil$

min (A, r) — min. value of A in 'r'

max (A, r) — max " " " "

Selection Size Estimation

1) $\sigma_{A=V}(r)$, $\frac{n_r}{V(A,r)}$

⇒ If A is key, then result will be 1/0 tuple.

⇒ Assume uniform distribution of values, $\boxed{\frac{n_r}{V(A,r)}}$, V(A,r) = values of occurrence of A in r

2) $\sigma_{A \le V}(r)$

AdmNo	Marks
1	90
2	80
3	40
4	35
5	22

a) If A is less than equal to v as 0, if $v < \min(A,r)$

b) If A is ≥ max(A,r) then n_r tuples (All tuples).

c) Otherwise, (b/w 22 to 90),

$$\frac{n_0 * (V - \max(A,r))}{\max(A,r) - \min(A,r)}$$

d) $\sigma_{A<V}(r)$, $\boxed{\sigma_{A \le V}(r) - \sigma_{A=V}(r)}$

e) $\sigma_{A>V}(r)$, $\boxed{n_r - \sigma_{A \le V}(r)}$

f) $\sigma_{A \ne V}(r)$, $\boxed{n_r - \sigma_{A=V}(r)}$

g) Conjunction ($\sigma_{\theta_1 \wedge \theta_2 \ldots \theta_n}(r)$)

⇒ Selectivity of condition θ_i is the probality that a tuple in relation r satisfies θ_i. = $\frac{s_i}{n_r}$

$$\Rightarrow \left[\frac{s_1}{n_r} \times \frac{s_2}{n_r} \times \ldots \times \frac{s_n}{n_r}\right] n_r$$

$$\Rightarrow n_r * \frac{s_1 * s_2 * s_3 * \ldots * s_n}{(n_r)^n}$$

Adm	Prog	Gender
1	BT	M
2	BT	F
3	MT	M
4	MT	M
5	BT	F
6	DD	M

$s[BT] = 3/6 \Rightarrow 1/2$

$s[MT] = 2/6 = 1/3$

$s[DD] = 1/6$

$s[M] = 4/6 = 2/3$

$s[F] = 2/6 = 1/3$

Q. Find $\sigma_{Prog=BT \wedge Gender=M}(r) = ?$

$\Rightarrow 6 * \frac{3}{7} * \frac{4}{6} \Rightarrow 2$

$\sigma_{Prog=MT \wedge Gender=F}(r) = 6 * \frac{2}{6} * \frac{2}{6} \Rightarrow \frac{2}{3}$

h) Disjunction :- $\sigma_{\theta_1 \vee \theta_2 \vee \theta_3 \ldots \theta_n}(r)$:-

$$n_r - n_r\left[\left(1 - \frac{s_1}{n_r}\right)\left(1 - \frac{s_2}{n_r}\right) \cdots \left(1 - \frac{s_n}{n_r}\right)\right]$$

i) Negation :- $\sigma_{\neg\theta}(r) = n_r - \sigma_{\theta}(r)$

Size of Joins

$r \times s = n_r \times n_s$

1) IF $(R \cap S) = \phi$, then $r \bowtie s$

2) IF $(R \cap S)$ is a key for 'R'

# of tuples in $r \bowtie s < n_s$

3) RNS is a foreign key in S referencing R

No. of tuples in $r \bowtie s = n_s$

4) $R \cap S = \{A\}$ is not a key for R and S

⇒ If we assume that every tuple T in R produces tuples in $r \bowtie s$, then # of tuples in $r \bowtie s$ = $\frac{n_r * n_s}{V(A,s)}$

⇒ If we assume every tuple T in s produces in $R \bowtie S$, then

no. of tuples = $\frac{n_r * n_s}{V(A,r)}$

r

Adm	Name
1	A
2	B
3	C
4	D

S

Adm	Marks	Sid
1	10	S_1
2	20	S_2
1	30	S_3
2	40	S_4
1	50	S_5
2	60	S_6

$R \cap S$ = Adm (Key of R)

r ⋈ s
1
1
1
2
2
2

result will never be more than n_s as RNS is key of r.

r

Adm	Sid
1	S_1
3	S_2
4	S_3
	S_4

$n_r = 4$

$V(Sid, r) = 4$

S

Tid	Sid
T_1	S_1
T_2	S_2
T_3	S_3

$n_s = 3$

$V(Sid, s) = 3$

$r \bowtie s = 4 * \frac{3}{3} \Rightarrow 4$

Estimation of Size of Projection

$\pi_A(r) = V(A, r)$

1) $A\,\mathcal{S}_F(r) = V(A, r)$

2) Set operation

$\sigma_{\theta_1}(r) \cup \sigma_{\theta_2}(r) = \sigma_{\theta_1 \vee \theta_2}(r)$ [Disjunction]

$r \cup s = \lceil n_r + n_s \rceil$

$r \cap s = \lceil \min(n_r, n_s) \rceil$

$r - s = \lceil n_r \rceil$

$r \bowtie s = \lceil n_r \rceil$ + Size of $r \bowtie s$

$r \bowtie s$ = Size of $r \bowtie s$ + $\lceil n_s \rceil$

$r \bowtie s$ = Size of $r \bowtie s$ + $\lceil n_s + n_r \rceil$

Choice of Evaluation Plan

→ Cost based - Approach

→ Heuristic

$r_1 \bowtie r_2 \bowtie r_3 \cdots r_n$

ways = $\frac{\lceil 2(n-1) \rceil !}{(n-1)!}$

↳ Perform Selection early (not always true)

Eg: $\sigma_\theta(r \bowtie s)$

and s is very large compared to r.

$\sigma(r \bowtie \sigma_\theta(s))$

⇒ $r < s$

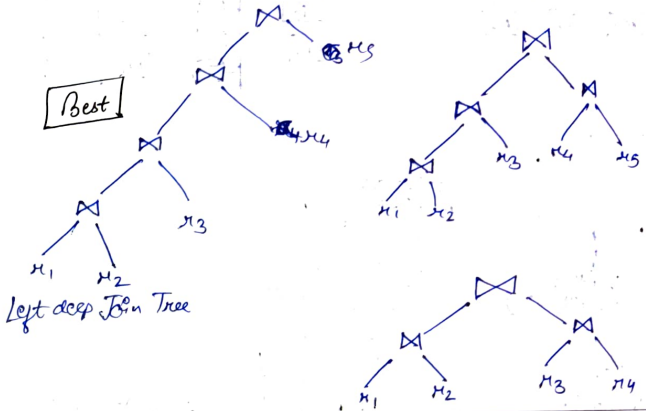
⇒ Index on the join attr. of s exists. No index on the attribute by θ of 's'

→ So join will take place by non key (B,C) & takes time

$S(A, B, C, D)$ — Index on B, θ on C

→ Perform Projection early

→ Perform most restrictive selection & Join Op^n (i.e. with smallest result size) before other similar Op^n.

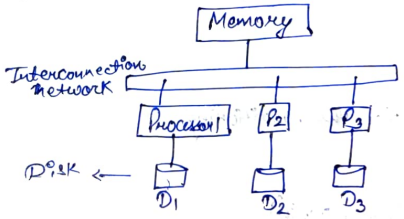


Parallel Databases

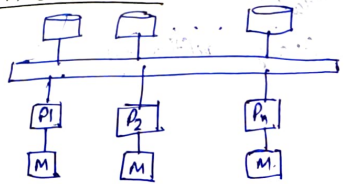
→ Objective = To provide Parallelism

→ Query can be processed from multiple places.

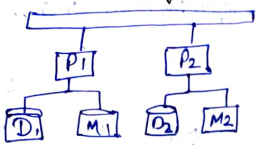
1) Shared Memory Architecture



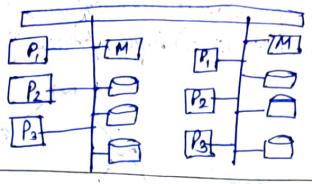
2) Disk Architecture

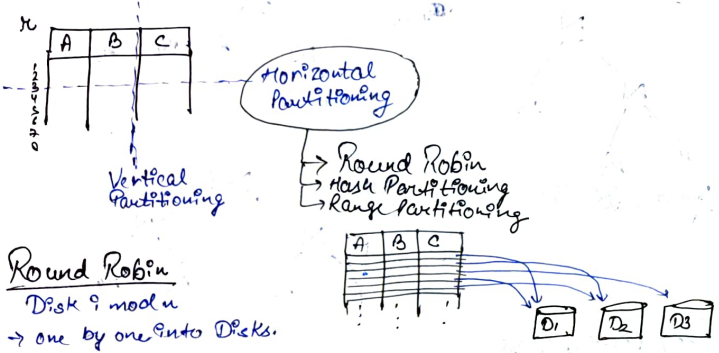


3) Shared Nothing Architecture



4) Hierarchical Architecture





1) Round Robin

Disk i mod n

→ one by one into Disks.

2) Hash Partitioning

h(i) → it will return disk no.

3) Range Partitioning

Partition Vector = [5, 11]

⇒ Blocks 1 to 5 → D_1

⇒ Blocks 6 to 11 → D_2

⇒ Blocks 12 - ∞ → D_3

⇒ Better the Partitioning Vector better the Partitioning.

Types of Query

1) Scanning the entire relation (Round Robin)

2) Point Query A=5 (Hash)

3) Range Query 5<A<11 (Range Partitioning)

Advantage of Round Robin

→ Best suited for sequential scan of Entire Relation.

→ Complex for Range queries.

Advantage of Hash Partitioning

→ Sequential Scan

→ If Point Query & search key is from same Attr. on which Hash Join has been performed.

→ In worst case of Range Query, scanning the whole Disks.

Advantage of Range Partitioning

→ Is Good for sequential scan & Range Query.

→ for Point Query also it is easy to search.

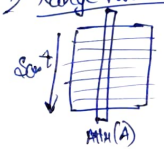
Handling Skew

→ Attribute-value skew

→ Partition Skew.

→ Some values appear in the Partitioning Attr. of many tuples. So all tuples will fall into the same Partitioning.

1) Range Partitioning



Sort relation based on skewed Attr. Then it will divide m blocks into n partitions (m/n) to avoid skewness

Histogram

Virtual Processes Partitioning

⇒ n real Processes

⇒ m >> n (VP)

2) Hash Partitioning

→ To avoid skewness, make good Hash function.

Advantages of Partitioning

1) Different queries can run in ll on data at different sites.

2) Time required for retrieval of relation (data) from disk is reduced.

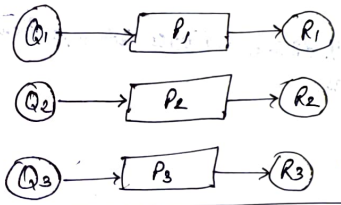
3) Increase in throughput of transaction processing.

Parallelism

Interquery Parallelism
1) Multiple query running

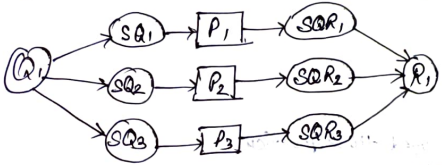
Intraquery Parallelism
1) Single Query involved

- Intraoperation Parallelism
  - → Multiple operations (σ, π)
  - Parallel Sort
    - → Range Partitioning Sort
    - → Parallel External Merge Sort
  - Parallel Join
    - → Partition Join
    - → Fragment & Replicate Join
    - → Partition Parallel Hash Join
    - → Parallel Nested Join
- Interoperation Parallelism
  - Pipeline Parallelism
  - Independent Parallelism

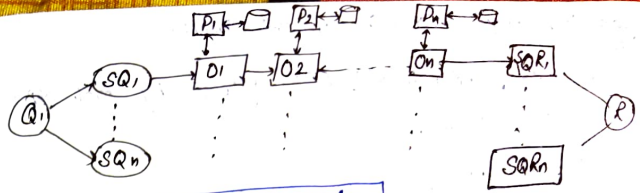


where, Q = Query
P = Processor
R = Result

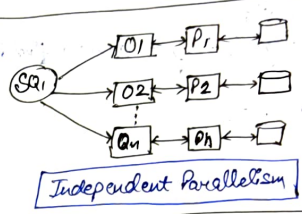
→ Interquery Parallelism



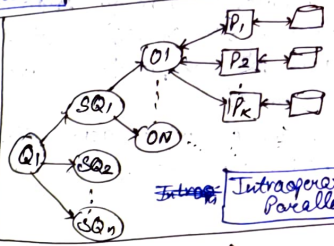
Intraquery Parallelism



Pipeline Parallelism



Independent Parallelism



~~Interop~~ Intraoperation Parallelism

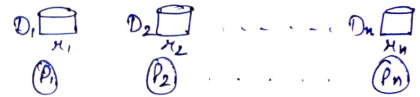
Interquery Parallelism (Parallism among Query)
- → Easiest form of ||sm to support particularly in Shared m/m || db.
- → More complicated to implement on shared disk or Sharing Nothing Arch.
- → Some of the examples of cache coherence protocols of shared disk systems are Locking of data (Shared/Exclusive).
- → Cache Coherence protocols for shared Nothing System uses Home processor (All independent M/c)

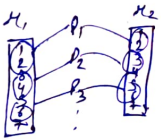
Intraquery Parallelism (Parallelism within Query)

Parallel Sort
→ Sort a Relation that resides on n disks.

$R \rightarrow D_1, D_2, \ldots D_n$

1) Range Partitioning Sort





1) → Identify Range Partitioning vector
2) Redistribute the relation using RP vector.
   a) All tuples that lie in the i^{th} range are sent to the processor P_i
   b) P_i stores the result temporarily if receives on D_i.

c) ~~Disk~~ These steps require I/O & Communication Overheads

* We get the Sorted data in all Disks followed by each dist

4) final merge op^n is trivial.

2) Parallel External Merge Sort

1) Assume relation is already Partitioned.
2) Each processor will locally sort the data.
3) Sorted runs ~~for~~ on each processors are merged to get sorted final data.

A Sorted Partition

⇒ P_i are range Partitioned across the processor.
⇒ Each P_i has own sorted part.

Distributed Databases

→ DS – Loosely Coupled System → Each Process is having its own m/m & disk.
→ OS – NOS/DOS
↓ Network OS ↗ Distributed OS

NOS – no. of systems are connected together

DOS – Responsibility of System to hide the identity of multiple m/c.

→ System image
→ Autonomy
→ Fault Tolerance Capability
↳ low for NOS (Data is present on one node only)
↳ High for DOS (Because of replicas of data)

1) **Homogeneous DDB** → Same type of DBMS are present on every node
→ all sites have identical softwares.
→ Each site surrenders parts of its autonomy in terms of right to change Schemas/software
→ Query processing is easy.

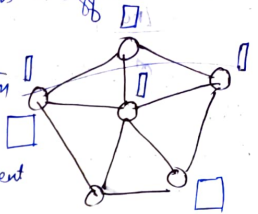
2) **Heterogeneous DDB** :- Different types of DBMS.
→ Different Schemas have different Softwares
→ Query Processing is typical so that Transaction management

Distributed Data Storage (R)

1) **Replication** :- Multiple copies of Relations are made and distributed to every nodes. Challenges are write op^n to ensure consistency.

2) **Fragmentation** :- Store fragments of relations in different nodes.

3) **Combination of Replication & fragmentation** :-



4) **Transparency** :- User is provided experience of single coherent System. (Everything is present on the user's System).

Fully Redundant DBs

→ Every site contains a copy of entire database.

Advantages of Replication
1) Data Availability ↑
2) Parallelism ↑
3) Reduced data Transfer

Disadvantages of Replication
1) Write/Update (In all places of Copies)

Horizontal fragmentation

account(AN, BN, BAL)

$\sigma_{BN='A'}$ (account)
$\sigma_{BN='B'}$ (account)

Advantages
1) Parallel Processing on fragments of relations.

Vertical Fragmentation

$\Pi_{AN,BN}$ (account) → fragment 1
$\Pi_{AN,BAL}$ (account) → fragment 2

→ Common Attr as key to Join them Again

Advantages:
1) Parallel Processing

Fragmentation Transparency → User is unaware of location of the fragmented data storage.

Replication Transparency = Replication data location

Location Transparency =

Naming Of Data Items

1) Criteria

1) Every data item must have system-wide unique name.
2) It should be possible to find the location of data items efficient
3) It should be possible to change the location of Data item transparency.
4) Each site should be able to create new data item autonomously.

Centralized Scheme (Name Server)

→ Its a name server responsible for assigning naming.
→ Each site maintains a record of local items of data

Advantages:-

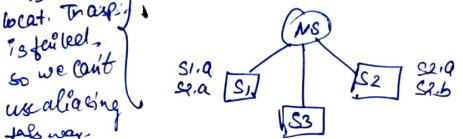
1) System wide unique name.
2) Efficient to find location of Data items.
3) Possible to change the location of data items.

Disadvantages

1) Bottleneck
2) Single point of failure.
3) Each site can't create new data item autonomously.

↳ Solution
use of aliasing like S1.a, S2.a

But locat. Transp. is failed, so we can't use aliasing this way.



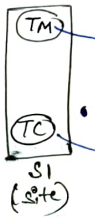
Solution

Create a set of aliases for data items; Store the mapping of aliases to the real names at each site.

Distributed Transactions → Local Trans.
↳ Global Trans.

Atomicity ✓
Consistency ✓
Isolation ✓
Durability ✓

→ Each site has local Transaction Manager & Transaction Coordinator.

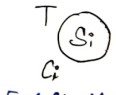


System Structure

Responsibilities of TM
1) Ensuring ACID Properties to local Transactions. (maintains log file)

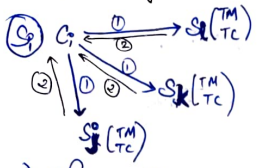
Responsibilities of TC :-
1) Starting the execution of Transaction that originates at the site.
2) Distributing sub Transactions at appropriate sites for execution.
3) Coordinating the Termination of each Transaction that originates at the site which may result in the Trans. being comitted at all sites or aborted at all sites

Commit Protocols
→ 2PC (2 Phase CP)
→ 3PC (3 Phase CP)



Fail Stop Model
↳ when any sort of failure occurs the site immediately stops executing.

Phase 1
(obtaining Decision)



1) <Prepare T>

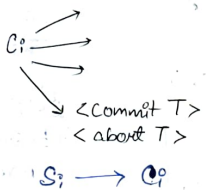
[TM at sites determines if it can commit the Trans.]

2) If No → <no T> (to log) → <Abort T> to Ci
   If Yes → <ready T> (to log) → <ready T> to Ci

Phase 2
(Recording the Decision)

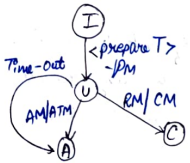
⇒ T can be committed?
<ready T> from all sites

⇒ T should be aborted?
Any one of site has aborted or any site has not responded in stipulated time.



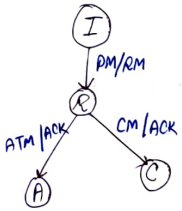
State Diagram

Coordinator



I = initial state
U = undecided state
RM = Ready message
CM = Commit message
PM = Prepare T message
AM = Abort message

Participant



R = Ready state
ACK = acknowledgement

Handling of failure

1) Site Failure

a) If the Coordinator Ci detects that site has failed
Actions: 1) Site is at initial state (It means site has a failure) ⇒ Abort
Abort ⇒
2) Site failure at (R), after ready message has been sent,
↳ failure is IGNORED

On Site Recovery

⇒ Check LOG
↳ Log Contains <commit T> record : redo(T)
" " <abort T> record : undo(T)
" " <ready T> record : Site must Consults Ci to determine whether it's Committed or Aborted.

UP
→ Check status, then send Commit(redo T) or abort(undo T)

DOWN
→ If Ci is down, Si must must try to find out the fate of T from other sites.
<query status T> → to all sites of the System.
Yes → If any site has the information about T
No → In periodic Time it again send message.

2) Coordinate Failure

IPC: A new coordinator to be elected, Reexecute Transaction Protocol.
If some acting Participating sites does contain <Ready T> Block in its Log, then the failed Coordinator Ci can not have decided to Commit. ⇒ ABORT
<commit T> ✓
<abort T> ✓
<ready T> ✗

If none of the above cases hold then all active sites must have <ready T> records in their log but no additional control records such as <abort T> or <commit T>

Blocking Problem:- when all sites have sended <ready T> msg to coordinator, then coordinator fails & all sites have to wait to get current state of Transaction(abort/commit) after it recover, And all sites will wait and block the resources they are using.

Scenario1: One of the Live participant received the decision
⇒ Non-Blocking Problem.

Scenario2: None of the Live Participant know the decision of Coordinator.

Case1 ⇒ All participants are Live (No failure)

Case2 ⇒ Participants also failed along with Coordinator. (Have to wait)

3) Network Partition

1) If coordinator & its Participants are in one Partition then no failure
2) If they belongs to several Partition, they may be blocking Problem
   a) Sites that are not in the Partition where coordinator is present, think that Coord. has failed and executes the protocol to deal with failure of coord.
   b) Coord. & sites are in same partition think that the sites in other partition have failed.

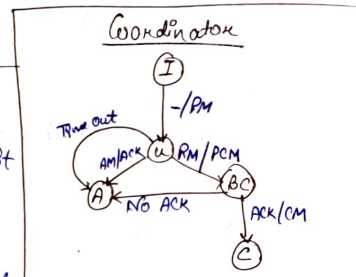
Recovery & Concurrency Control

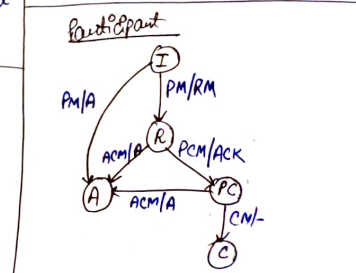
→ In-doubt Transaction is that we don't know its Abort or Commited.
→ The Recovering site must determine the Commit/Abort status of such Transactions by contacting other sites. This can slow & potentially Block recovery.
→ Recovery Algo. can ~~lock~~ ~~know~~ note Lock info. in the log.
<ready T, L>, where L = List of all write locks held by T where the Log is written.
→ For every in-doubt Transaction T, all the Locks ~~X~~ noted in the <ready T, L> log record are reacquired.

3 Phase Commit Protocol

⇒ To avoid the Blocking Problem of 2PC.
→ ~~No~~ Assume, no N/w Partitioning.
, At any point at least one site is up.
, At most K sites (Participants as well as Coordinator) can fail.

where, PCM = Prepare to Commit msg
BC = Before Commit
PM = Prep. msg
RM = Ready msg
AM = Abort msg
ACM = Abort Commit msg





Coordinate failure

1PC → Same as 2PC

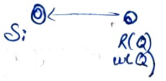
2PC → Also have a Participant failure
Participant will be in PC (Prepare to commit State) & Coordinator will be in BC (Before Commit State). Then ~~we can~~ Rest of the participants elect new coordinator

3PC → Coord. passed to 3PC & failed. (Commit State).
Taken decision to commit & convey to Participants.

Concurrency Control

1) Locking Protocol

a→ **Single Lock Manager Approach**: One Sigle is SLManager & other willing to sites to take lock will ask from manager.
If op" is Read, then data can be read from any Replica.
If op" is write, then data/change should be written back to all replica's

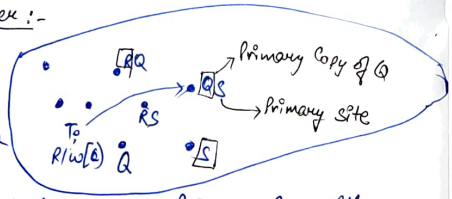


Advantages of SLMA:-
1) Implementation is easy.
2) If there is deadlock, the manager can easily resolve.

Disadvantages of SLMA:-
1) If manager is down, then System will shut (Single Point of Failure)
2) There will be bottleneck (Traffic).

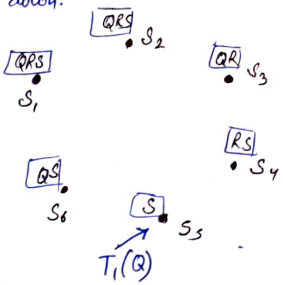
b) **Distributed Lock Manager:-**

1) **Primary Copy:-**
Primary Copy of a Sigle data item work as Single Lock Manager.
Once a write Op" is started then then Primary Copy will send msg to invalidate all copy of data (Replicas).



Drawback of Primary Copy —
1) If Primary Copy site is down.

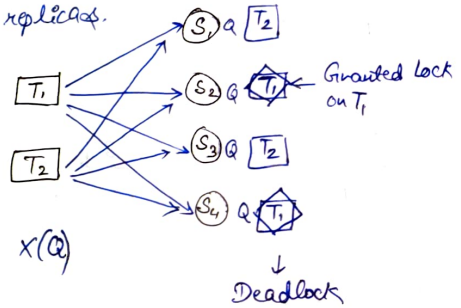
2) **Majority Protocol**
a) Q is Replicates at n = 4 sites, then a lock requests msg must be sent to more that half of the n sites on which Q is stored.
Msg Exchange = $2\left(\frac{n}{2}+1\right)$
Lock Released = $\left(\frac{n}{2}+1\right)$



b) The Transaction doesn't operate on Q until it has obtained ~~an~~ lock on majority of Replicas of Q.

c) When writing the data item, Transactions performs writes on all replicas.

3) Deadlock occurs.



4) Both Read & write op" involves same Level of Complexity.

4) Implementation is complex.

3) Biased Protocol:-

→ Biased towards Read/Shared Lock.

Advantages
1) Read op" can be handled faster.
Disadvantage:- Possibly of Deadlock, Complexity (Are Disv of Majority Protocol)

4. Quorum Consensus Protocol :

→ The Protocol assigns each site that have replica with a weight

→ For any data item, the Protocol assign read Quorum (Q_R) & write Quorum (Q_W) → integers. (Sum of weights of some sites) and these 2 integers are chosen according to the following conditions put together:

$$Q_R + Q_W > S$$
$$2Q_W > S$$

Q to All (S_1 to S_4)
weight = 1
S = 4

S_1	S_2	S_3	S_4
0	0	0	0

S_1 S_1 ← Read
① ①

S_2 S_3 S_4 ← write
① ① ①

Ex: $Q_R = 2$, $Q_W = 3$, $S = 4$
$2 + 3 > 4$
$2 \times 3 > 4$

Ex: $Q_R = 1$, $Q_W = 4$, $S = 4$
$\Rightarrow 1 + 4 > 4$
$\Rightarrow 2 * 4 > 4$

S_1 → Read
S_1 S_2 S_3 S_4 → Write

2) Timestamping -

T_i issue read (Q)

a) If Timestamp (T_i) < W_TS(Q)
then T_i needs to read value of Q that is already over written
⇒ read (Q) is Rejected
⇒ T_i is rollback.

b) If TS(T_i) ≥ W_TS(Q)
then, read (Q) is executed
R_TS(Q) = max(R_TS(Q), TS(T_i))

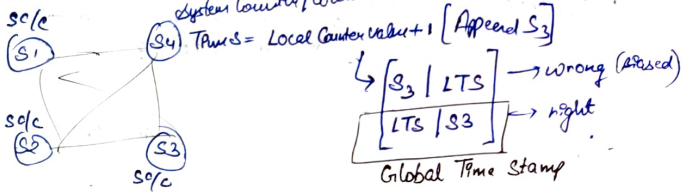
T_i → Q

T_i issued write(Q)

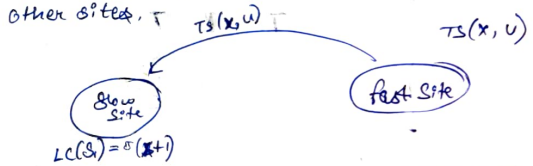
a) If TS(T_i) < R_TS(Q), then value of Q that T_i is producing was needed previously & system assume that value would never be produced.
write (Q) is REJECTED., T_i is Rollback.

b) If TS(T_i) < W_TS(Q)
write (Q) is REJECTED, T_i is Rollback

c) Rest all conditions, write(Q) is EXECUTED.
W_TS(Q) = TS(T_i)



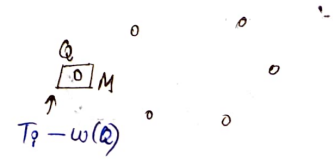
Note:- A site with slow clock will assign smaller Timestamp. The fast sites logical counter will be larger than that of other sites.



3) Replication with weak Consistency -

↳ DB is updated slowly

Master - Slave Replication



⇒ Replica should see a transaction consistent Snapshot

Multi Master replication: In this updates are permitted at any replica & are automatically propagated to all replicas.

Lazy Propagation:- Many systems supports lazy propagation where updates are transmitted after transaction commits but at a cost of consistency.

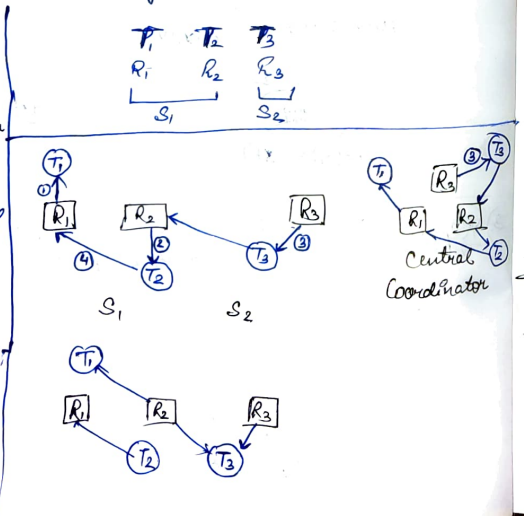
It also allows updates to occur even if some sites are disconnected from the nw but at the cost of consistency

- Update at Replica [Go to primary site immediately and then propagated lazily to all Replicas. To ensure ordering]
- Update at any Replica → Propagate to all replica

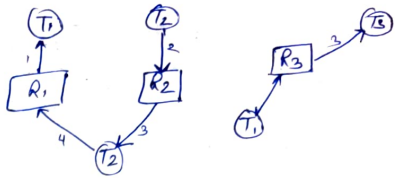
4. Deadlock Handling :-

⇒ Easy in case of centralized Systems.

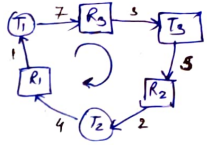
- Centralized
- Hierarchical
- Distributed
  - wait for Graph
  - probe based



1) Transaction T_1 requested for R_1
2) T_2 requests R_2
3) T_3 requests R_3
4) $T_2 \leftarrow R_1$ (waiting)
5) $T_3 \leftarrow R_2$ (waiting
6) T_1 releases R_1, then R_1 will go to T_2
7) $T_1 \leftarrow R_3$ and is waiting

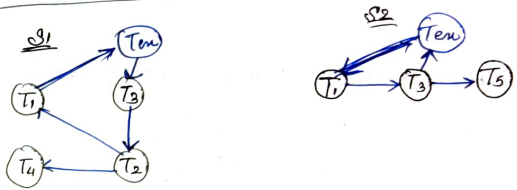


⇒ Resolve:- By appending a unique Global Time Stamp with each msg



[Such situation is known as false Cycle.]

Wait For Graph (WFG)



⇒ An extra node Tex is added to local ~~graph~~ wait for graph of each site & this node is connected to wFG of the corresponding site in the following manner.

a) an edge T_i to T_{ex} is added if a T_i is waiting for Resource in another site being held by any Transaction.

b) An edge T_{ex} to T_j is added if T_j is a Transaction of another site that is waiting for a resource currently being held by a Transaction of this site

At site1

a) An edge T_1 to Tex is added because a Trans (T_1) is waiting for resource at another site 2 that is held by T_3 and an edge T_{ex} to T_3 is added because a Transaction T_3 is a process of site 2 that is waiting to acquire a res currently held by a Transaction T_2 of site1.

At Site2

An edge T_3 to T_{ex} because a Transaction T_3 is waiting for resource inside S1 that is held by T_2 and an edge T_x to T_1 because a Trans. T_1 of site S1 that is waiting to acquire a resource currently held by a Transaction T_3 of site 2.

To confirm a distributed deadlock a DistDDetection Algo. is invoked by a site whose WFG contains a cycle involving T_{ex}.

$(T_{ex}, T_9, T_j, \ldots T_k, T_{ex}) \rightarrow S_j$

S_i = initiating the Algo (S_i has detected formation of cycle)

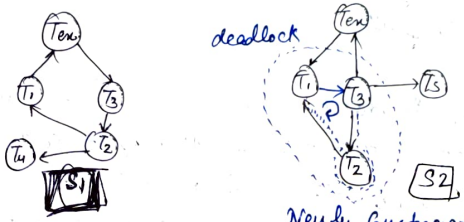
a) S_i will send deadlock detection Message to S_j.

b) The DDM doesn't contains complete WFGs of site S_i but only the part of WFG that forms the cycle.

Ex:- In above Graph msg send will be:-

at site 1 $\Rightarrow (T_{ex}, T_3, T_2, T_1, T_{ex})$ sended to S2

c) On receiving the msg, site S_j updates its local WFG by adding those edges of the cycle that don't involve T_{ex} to its WFG (wait for Graph).



Newly Constructed Graph

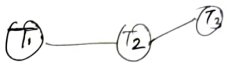
$\Rightarrow$ If a site S_j contains a cycle that doesn't involve T_{ex} then deadlock exists and appropriate recovery process must be initiated.

Probe Based

Probe message = ~~<Sender>~~

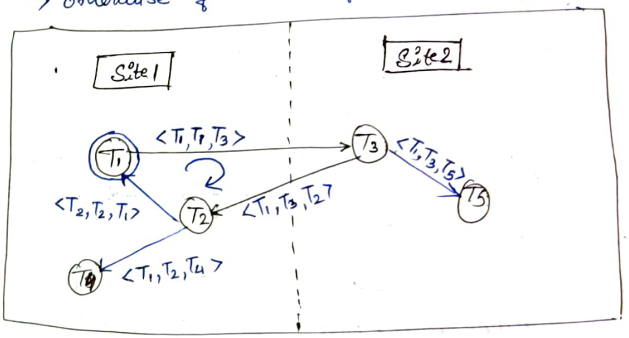
<originater, sender, receiver>

$< T_1, T_1, T_2 >$



↳ if T_1 grants a resource which it wants then it will proceed

↳ otherwise it will check for deadlock by sending Probe msg.



1) T_1 requests for T_3, then wait for sometime & then send probe msg = $< T_1, T, T_3 >$

2) Then T_3 is also waiting for some resource at T_5, then it will send $< T_1, T_3, T_5 >$

3) Then T_5 is doing nothing & will ignore the msg.

4) After that T_1 will finally received the msg it has sended at the starting that means there is deadlock.

Distributed Query Processing

→ In centralized based on Disk access we are choosing optimized Query.
→ But in distributed we need to calculate Communication Cost to transfer b/w different sites.
→ It allows Parallelism, due to replicas.

Q- Find all records of accounts relation.
Ans- Select * from account (in centralized System)

But in Distributed → Replication ($\frac{R}{F}$)
→ fragmentation (R / F) ... So on
→ R & F ...

⇒ So Basically portion of account relation needed to find that where account relation is situated. ~~and~~ we need to construct back account.

a) If only Replication is used, then choose any Replica of account
b) If Replica is fragmented, then we need to use many joins and unions to reconstruct the relation.
c) If no Replicas are fragmented, then whichever replica is closer to site.
d) Relation is Fragmented

Simple Join Strategy

account ⋈ depositor ⋈ branch
S_1 S_2 S_3

Query for Join → S_2

Ship ✓
Shift ✗

Different ways to Handle Joins:-
1) Shift Copies of all 3 Relations to S_2.
2) Shift a Copy of Account Relation to S2
   Then at S2, find [temp1 = account ⋈ depositor]
   After that, Shift temp1 to S3,
   Then, temp2 = temp1 ⋈ branch [At S3]
   Shift temp2 to S_2 (Result).

Numerical

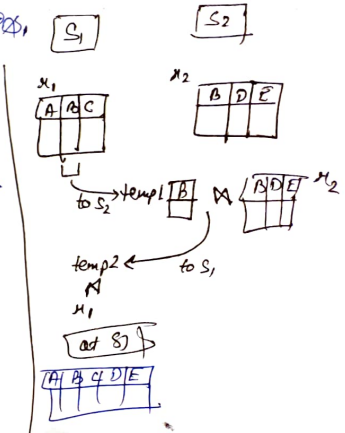
Must Considered following factors to choose (1) or (2):
1) Amount of data being Shipped.
2) Cost of transmitting data Blocks b/w Sites.
3) Relative processing Speeds at each Sites

Semi-Join Strategy

$r_1(R_1)$ → Site1
$r_2(R_2)$ → Site2

Q $r_1 \bowtie r_2$ and obtain the result at S_1

Steps:-
1) Compute temp1 ← $\Pi_{R_1 \cap R_2}(r_1)$ at S_1.
2) Ship temp1 from S_1 to S_2.
3) Compute temp2 ← $r_2 \bowtie$ temp1 at S_2
4) Ship temp2 from S_2 to S_1.
5) Compute $r_1 \bowtie$ temp2 at S_1.



3rd Step: $x_2 \bowtie \pi_{R_1 \cap R_2}(x_1)$

5th Step: $x_1 \bowtie x_2 \bowtie \pi_{R_1 \cap R_2}(x_1)$

$\Rightarrow x_1 \bowtie \pi_{R_1 \cap R_2}(x_1) \bowtie x_2$

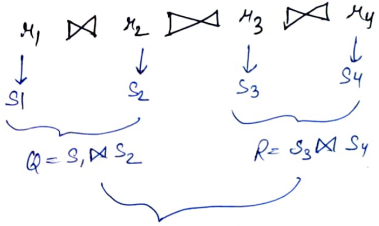
$x_1 \bowtie x_2$

Semijoin at x_1 with $x_2 \Rightarrow x_1 \ltimes x_2$

↳ selects those tuple at x_1 that contributed to $x_1 \bowtie x_2$

$\pi_{R_1}(x_1 \bowtie x_2)$

Using Parallelism



Heterogeneous Distributed Databases (Multi-Databases System)

Wrapper: A Software which performs schema translation.

Advanced Transaction Processing

Transaction Processing ~~Manager~~ Monitor:-

→ It provides infrastructure for building & administering complex transaction processing system with Large no of clients & multiple servers.

→ It provides functionalities such as :- Managing, Deploying and ~~Devel~~ Developing transactional Distributed Information Systems.

→ TP monitors acts as middleware, its main task is to Support and handle interaction b/w applications on a variety of computer platforms.

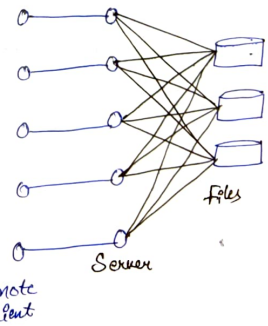
→ It controls programs that monitor or manager, a transaction of data as it passes from one stage in a process to another in an ~~org~~ organised or, transaction oriented manner.

→ ~~It~~ Transaction Processing is Critical in Multitier Architecture with Processes running on different platforms, a given Transaction may be forwarded to any one of the several servers.

→ TP monitor handles all Load balancer.

TP Monitor Architecture

a) Process Per-client Model :-



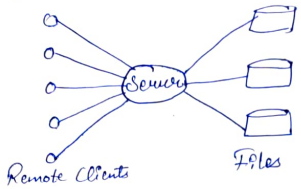
→ Instead of individual Logging session per terminal, server process communicates with the terminal.

Problems

→ High CPU overhead for context switching

→ Memory like

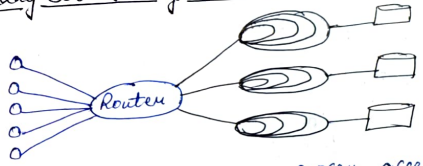
b) Single Server Model:-



→ Multithreading Concept is used
→ The Server will create multiple threads or port to multiple clients

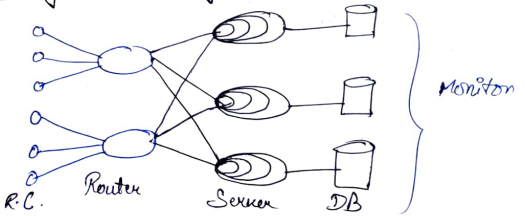
Problem:-
1) Because of sharing Address space each thread will share address space there concern of Protection among Application.
2) Moving System to another, then there will be change in address space

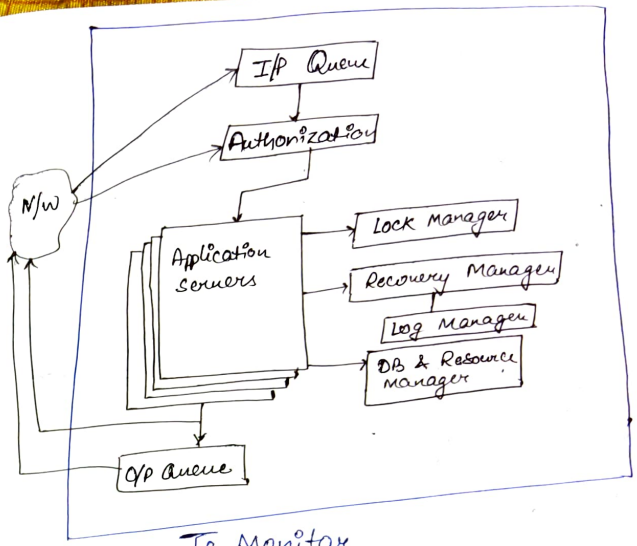
c) Many Server Single Server Model



→ Multiple application server process access a common database
→ Clients communicates with the application through a single communication process that routes request.
→ It supports Independent server processor for multiple applications
→ Each appl can have a ~~pot~~ pool of server processors.
→ Multithread server processors are allowed.
→ Communication process handles communication among process.

d) Many Server, Many Router Model:-





Tp Monitor

Persistent & Durable Messaging Queue

→ I/P Queue
→ O/P Queue
→ Queues are safe even after System will fail

Remote Procedure Call (RPC)

→ Widely used in Distributed Systems.

Workflow System

→ Workflow Specifications – Predefined things
→ WF Execution – whether in a particular Fashion

A → B → C → D
(t_1) (t_2) (t_3) (t_4)

[t_1 must be committed] – Necessary condition

[WF Specifications = $\{t_1, t_2, t_3, t_4\}$]

Cases:-

1) Previous activity must have committed.
2) Previous activity has committed plus the output must be ~~as~~ that is required.
3) Based on time any activity is Triggered.

→ **Workflow Specifications**
  ↳ static spec.
  ↳ Dynamic spec.

a) **Static Specifications**

→ Tasks & Dependencies among them are defined before the execution of workflow starts.

Pre-Execution
1) Execution state of other tasks.

$t_i \rightarrow t_j$

2) Output value of other tasks

$t_i \xrightarrow{[x>25]} t_j$

3) External variables that are modified by some external parameters.
Ex:- Clock

$t_i \xrightarrow{9\,hrs} t_j$

b) **Dynamic Specifications:-**

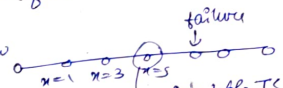
→ Not decided by the specifier. in prior.
Ex:- Entering a URL in Browser, Sending mail (Different path from the Routers)

Failure Atomicity Requirements

1) **Acceptable Termination States:-** Every execution of workflow will terminate in a state that's satisfies the failure Atomicity Requirements defined by the designer.

→ **Committed:-** Objectives of a workflow have been achieved

→ **Aborted:-** valid termination state in which a WF has failed to achieved its objective.



2) A workflow must reach an AT State even in the presence of System failure.

3) **Compensating Transaction:-** To undo the result of already committed Transaction.

Execution of Workflows

Workflow management Systems:- It includes
1) Scheduler → Everything is planned
2) Task Agents
3) Query Mechanism

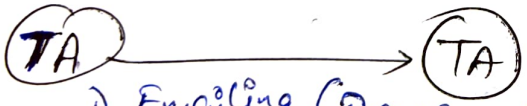
Job of Scheduler:-
Program that process workflows by submitting various tasks for execution, monitoring various events & evaluating conditions related to intertask dependencies.

Task Agents:- Controls the execution of a task by processing entities.

Query Mechanism:- Different workflow running.

There are Architectural Approaches to the development of Workflow Management Systems.

1) Centralized (Single Scheduler)
2) Partially Distributed (for each ~~Scheduler~~ workflow there is one Scheduler instantiated)
3) Fully Distributed (No scheduler, but task agents start communicating with ~~itself~~ each other & coordinate the execution to satisfy task dependencies & other workflow execution requirements.



(Communicating method)

1) Emailing (Dropping of messages very high)
2) Persistent Messaging (Guarantee of message delivery)