| Sname | Project_name | Proj_name |
|---|---|---|
| smith | | |
| smith | | |
| Adamsky | | |
| Walton | | |
| Adamsky | | |

The dependency preserving 3NF is in which $A_1 \bowtie R_2 \bowtie R_3 = R$ supply and we don't have any non-trivial join dependency in $R_1$, $R_2$ and $R_3$

Dt:- 30/09/24

## MODULE – IV

### Query processing

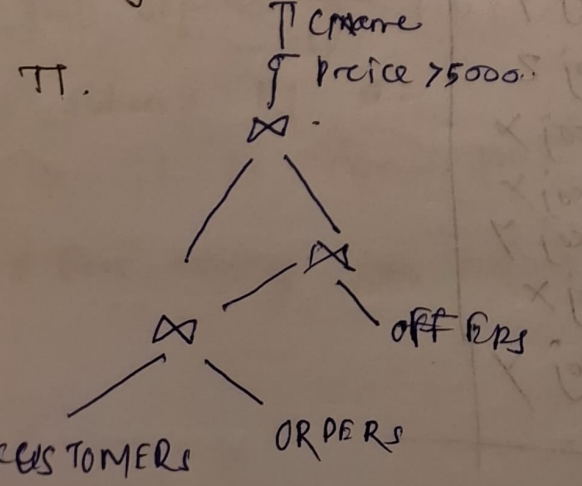
Query processing refers to the range of activities that are involved in a extracting data from a database

### Basic steps in query processing:-

The basic steps involved in processing query are:-

1) parsing and Translation.
2) Optimization
3) Evaluation

1) Parsing and Translation:-



## Information about relations and attributes:-

$N_R$ :- No. of tuple in relation R

$B_R$ :- No. of blocks that containing tuples of relation R. (where the data is stored)

$S_R$ :- size of tuple R

$F_R$ :- blocking factor; no. of tuple from R that fit into one block ($F_R = \lfloor \frac{N_R}{B_R} \rfloor$)

↳ scaling → big - O

$V(A, R)$ :- no. of distinct values for attribute A in relation R.

$SC(A, R)$ :- selectivity of attribut A.

= Avg no. of tuples of R that satisfy an equality condition on A

$$SC(A, R) = \frac{N_R}{V(A, R)}$$

Note:- The size of tuple means the amount of memory (in bytes) required for each tuple.

## Measurement of query cost :-

Number of disk seek, No. of blocks read, Number blocks written.

seek time - is amt the amount of time that a disk drive head take to move to a specific location of disk

Disk seek is the actual physical positioning of the re-drive head of the disk.

Disk transfer occurs from main memory to secondary memory

suppose

→ $t_T$ → time to transfer one block

$t_S$ → time for one disk seek operation

cost for b block transfers + S seek operations

$$= b \times t_T + S \times t_S$$

## Selection operation

$S1$ → linear search → for both sorted and unsorted list

$S_2$ → Binary search → only for sorted list

↓ ↓

It works on divide and conquer method.

It is much more faster than linear search.

No of blocks that needs to be examined to find a block containing the required record is $\lceil \log_2(B_R) \rceil$

$B_R$ → No. of blocks.

cost of Binary search, ie, the file ordered b

~~for N~~ If the selection condition is base on non key attribute the cost of $S_2$

$$cost(S_2) = \lceil \log_2(B_R) \rceil + \left\lceil \frac{SC(A,R)}{F_R} \right\rceil - 1$$

no of block required for storing the result of selection operation

then $SC(A,R) = 1$

for key attribute :-

$$\boxed{cost(S_2) = \log_2(B_R)}$$

if primary then $SC(A,R) = 1$

We consider a employee relation where it has 10,000 tuples. If blocking factor, $F_R = 10$ (employee)

$V(\text{deptno}, \text{employee}) = 50$

↳ The no. of distinct department existing on the employee database

Assume the following relⁿ operation.

$\sigma_{\text{deptno}=20}(\text{employee})$ and employee is sorted on such key deptno. Find the cost of selecting the required records that satisfy selection criteria using Linear search and binary search

Linear search

$\text{cost}(S_1) = \log_2 B_R + \log_2 1000 + \log_2 10$

$B_R = \frac{N_{emp}}{F_{emp}} = \frac{10000}{10} = 1000$

Linear search $= S_1 = 1000$

Binary search

$\frac{10000}{50} = 200$

$\text{cost}(S_2) = \log_2^{(B_R)} + \frac{SC(A,R)}{FR} - 1$

$= \log_2 1000 + \frac{200}{10} - 1 = 3 + 20 - 1$

$= 22$

$SC(A,R) = \frac{N_R}{V(A,R)} = \frac{10000}{50} = 200$

9) The emp rel^n is sorted on search key dept no.

There is an equal no. of employees present in each department.

Selection operation using Indices:- DT:- 22/10/24

Index scan:-

Notation

$HT_I$: numbers of levels in index I (B+- trees)

$LB_I$: number of blocks occupied by leaf node in index I (first-level blocks)

$Val_I$: numbers of distinct values for the search key.

B+ tree is balanced binary search tree. It follows a multi-level index format.

S3

S3 is a selection operation using index scan.

S3 - primary [index] for A, A primary key, with equality, comparison A=a (as the selection criteria)

Ex:- select * from student where std_id=101;

$$cost(S_3) = HT_I + 1$$ (only 1 tuple satisfies the condition)

$HT_I$ → no of levels in the B+ tree that is used to store the index I.

S4

S4 - primary index [~~for A, A~~ on non-key attribute A with equality comparison A=a] (as the selection criteria)

$$cost(S_4) = HT_I + \left\lceil \frac{SC(A,R)}{f_R} \right\rceil$$

S5

S5 - [Non primary/secondary index on non-key attribute A with equality comparison A=a

$cost(SS) = HT_I + SC(A,R)$

selection involving comparisions:

$S_6$ - selection involving a primary index on A

$S_7$ - selection involving a secondary index on A

$S_6$

Ex:- select * from student where std-id >= 105;

cost($S_6$) = seek time + B number of block before (<, ≤) are effect

(>, ≥) is added

$S_7$ (It is same as $S_5$) but here operation is comparison operation

Ex:- select * from student where age >= 18;

cost($S_7$) = $HT_I$ + SC(A,R) + N number of blocks searched

Q) Fate & employee

Ex:- Assume the query CUSTOMERS ⋈ ORDERS (with join attribute only being CName)

- $N_{CUSTOMERS}$ = 5000 tuples
- $F_{CUSTOMERS}$ = 20, i.e $B_{CUSTOMERS}$ = 5,000/20 = 250 blocks
- $N_{ORDERS}$ = 10,1000 tuples
- $F_{ORDERS}$ = 25 i.e $B_{ORDERS}$ = $\frac{10,000}{250}$ = 400 blocks
- V(CName, ORDERS) = 2500

The cartesian product R x S results in $N_R * N_S$ tuples.

$S_R$ = size of individual tuple in rel$^n$ R

each tuple requires $S_R + S_S$ bytes

$S_S$ = size of individual tuple in rel$^n$ S

Size of join using information about foreign key

If schema(R) ∩ schema(S) = foreign key in S referencing R, then the no. of tuples in R ⋈ S is exactly Ns.

→ In the example query CUSTOMER × ORDERS, the result has exactly Ns = 10,000 tuples.

Size of join without information about foreign key

If schema(R) ∩ schema(S) = {A} which is not a key for R or S

Then the no. of tuples:-

$$\min\left(\frac{N_R \times N_S}{V(A,S)}, \frac{N_R \times N_S}{V(A,R)}\right)$$

A = A is common in both R and S.

In previous ex:-

$N_R = 5000$, $N_S = 10000$

$V(A,S) = 5000$, $V(A,R) = 2500$

$$\min\left(\frac{5 \times 10^3 \times 10^4}{5000}, \frac{5 \times 10^3 \times 10^4}{2500}\right) = 20$$

$$\min(10^4, 2 \times 10^4) \quad \min(10000, 20000)$$

No. of tuples = 10000

Nested loop join:-

It is a join that contains a pair of nested for loops.

representation:-

$R \bowtie_C S$

C → condition

→ R is outer rel? and S is inner rel? of the join

Cost Analysis of nested loop join:-

Worst case:- The database buffer can hold only one block of each rel?

In this case the no. of block requires.

→ $B_R + N_R * B_S$ disk accesses.

Best case:- The database buffer.

There enough space for both rel? to fit in to the database buffer simultaneously, so in this case.

→ $B_R + B_S$ disk accesses.

Block Nested loop join:-

Here for each block no. of tuple are present.

Cost Analysis:- Worst case:-

Database buffer can hold only one block of each rel?

→ $B_R + B_R * B_S$ disk access.

Best case:-

both rel?s fit into database buffer

→ $B_R + B_S$ disk access.

Ex:- Nested loop join

Worst case:- $B_R + N_R * B_S$

$= 250 + 5000 \times 400$

$= 250 + 2000000$

$= 2000250$

best case:- $B_R + B_S$

$= 250 + 400$

$= 650$

Block nested loop join:-

worst case:- $B_R + B_R * B_S$

$= 250 + 250 \times 400$

$= 250 + 100000$

$= 100250$

best case:- $B_R + B_S$

$= 250 + 400$

$= 650$

Index Nested loop join:-

It is used to speed up the process of joining two tables.

COST ANALYSIS:-

Worst case:-

Database buffer has space only for one block of R and one block of index associated with S.

If c cost of a single selection on S using the join condition. Then cost of join = $B_R + N_R * c$

Q) CUSTOMER ⋈ ORDERS with CUSTOMER → orders
Let orderes have primary B+ tree index on the join attribute
CName Let the height of the B+ tree to accomodate
is 4'

$$sc(A,S) = \frac{N_S}{B(A,S)} = \frac{10000}{2500} = 4$$

what will be c To reach the root
cost c = height of the tree + one more access is need to get the a

$$= 4 + 1 = 5$$

$$\text{cost} = B_R + N_R * c$$
$$= 250 + 5000 * 5 = 250 + 25000$$
$$= 25250$$

## sorted merge join :-

It is the combination of two processes i.e sorting and merging

1) step 1 :- 1st both the reel's are sorted. base on the join attribute
step 2 :- join

## cost Analysis

if the reel's are sorted and tuples having the same value on the join attribute are placed consequitively then we need to reead each tuple only once and thus the disk blocks are also read once.

worst case :-
The no. of disk access = $B_R + \frac{B_R * B_S}{}$ + cost of sorting

best case :-
The no. of disk access = $B_R + B_S$ + cost of sorting

Q) CUSTOMER ⋈ ORDERS with CUSTOMER → outer rel$^n$
Let orders have primary B+ tree index on the join attribute
CName. Let the height of the B+ tree to accomodate
is 4'

$$SC(A,S) = \frac{N_S}{V_B(A,S)} = \frac{10000}{2500} = 4$$

what will be C to reach the
cost C = height of the tree + one more access is
need to get the a

= 4 + 1 = 5

cost = $B_R + N_R * C$

= 250 + 5000 * 5 = 250 + 25000

= 25250

Sort merge join:-

It is the combination of two processes i.e sorting and merging

1) step 1:- sort both the rel$^n$s are sorted.
base on the join attrib

step 2:- join

Cost Analysis

if the rel$^n$ are sorted and tuples having the same value on the join attribute are placed consequitively then we need to read each tuple only once and then the disk blocks are also read once.

worst case:-

The no. of disk access = $B_R + \frac{B_R * B_S}{B} +$ cost of sorting

best case:-

The no. of disk access = $B_R + B_S +$ cost of sorting

# Query optimization

Find the names of all customers who have an account at any branch located in "Brooklyn".

$\Pi_{customer\_name}(\sigma_{branch\_city = "Brooklyn"}(branch) \bowtie (account \bowtie depositor))$

$\Pi_{customer\_name}(\sigma_{branch\_city = "Brooklyn"}(branch \bowtie (account \bowtie depositor)))$

## Notations:-

$\theta_i \rightarrow$ represents predicate

$L_i \rightarrow$ used denote the lists of attributes

$E_i \rightarrow$ denote the relational-algebra expression.

$r \rightarrow$ relation name

## Rules

1) Conjuctive selection operations can be decomposed into a sequence of individual selections.

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

2) selection operations are commutative

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

3) only the final operation in a sequence of projection operations is required, the rest can be omitted

$$\Pi_{L_1}(\Pi_{L_2}(\ldots \Pi_{L_n}(E)\ldots)) = \Pi_{L_1}(E)$$

4) Selection operation can be combined with cartesian product and theta joins in the following manner

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

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

5) Theta-join operation are commutative

$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$

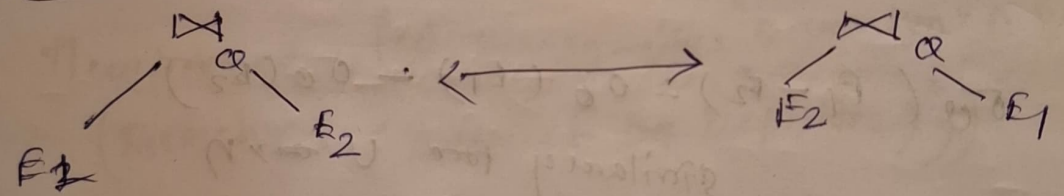
6) a) Natural join operations are associative

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

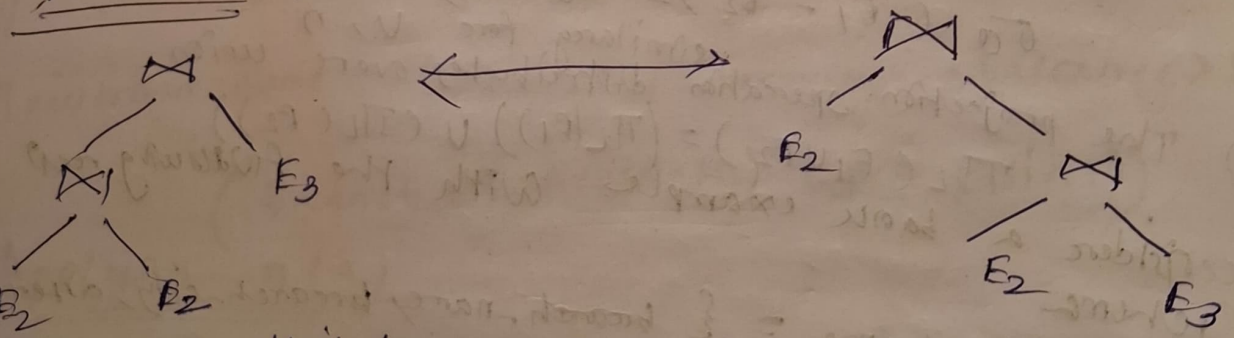
b) Theta join operations are associative

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

Rule - 5




Rule - 6 (a)




a) When all the attributes in $\theta$ involve only the attributes of one of the expression ($E_1$) being joined

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

When $\theta_1$ involves only the attributes of $E_1$ and $\theta_2$ involves only the attribute of $E_2$

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

8) (a) Let $L_1$ and $L_2$ be attributes of $E_1$ and $E_2$ respectively. If $\theta$ involves only attributes 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) From ppt

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

9) The set operation union and intersection are commutative

$$E_1 \cup E_2 = E_2 \cup E_1$$

$$E_1 \cap E_2 = E_2 \cap E_2$$

set difference is not commutative

10) set union and 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) The selection operation distributes over $\cup$, $\cap$ and $-$

$$\sigma_\theta(E_1 - E_2) = \sigma_\theta(E_1) - \sigma_\theta(E_2)$$

similarly for $\cup$ & $\cap$

$$\sigma_\theta(E_1 - E_2) = \sigma_\theta(E_1) - E_2$$

similary for $\cup$, $\cap$

12) The projection operation distributes over union

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

We consider a bank example with the following relational schema

Branch schema = { branch_name, branch_city, assets }

Account schema = { account_number, branch_name, balance }

Depositor schema = { customer_name, account_number }

Find the names of all customers who have an account at any branch located in "Brooklyn".

Branch (Account Depositor)

$\Pi_{customer\_name}(\sigma_{branch\_city = "Brooklyn"}(Branch \bowtie (Account \bowtie Depositor)))$

by applying rule 7(a) this query can be transformed to an equivalent query that generates smaller intermediate rel?

$\Pi_{customer\_name}(\sigma_{branch\_city = "Brooklyn"}(Branch)) \bowtie (Account \bowtie Depositor)$

suppose we modify the original query to restrict ~~ouere~~ ~~east~~ search on customers who have a balance greater than ₹1000 dollars

$\Pi_{customer\_name}(\sigma_{branch\_city = "Brooklyn" \wedge balance > 1000}(Branch \bowtie (Account \bowtie Depositor)))$

by applying rule 6(a)

$\Pi_{customer\_name}(\sigma_{branch\_city = "Brooklyn" \wedge balance > 1000}(Branch \bowtie Account) \bowtie (Depositor))$

Applying 7(a)

$\Pi_{customer\_name}(\sigma_{branch\_city = "Brooklyn" \wedge balance > 1000}(Branch \bowtie Account)) \bowtie (Depositor)$

Applying rule 1

$\Pi_{customer\_name}(\sigma_{branch\_city = "Brooklyn"}(\sigma_{balance > 1000}(Branch \bowtie Account)))$

Dt:- 04/11/24

## Schedule - 3

A = 1000

A = 1000 - 50
= 950

2000 × 1/100

950 × 1/100

B = 2000 A = 950

B = 1980 A = 1000 - 95 950 - 95

## Schedule - 4

1) A = A - 50
= 1000 - 50 = 950

A = 1000

A = 1000 - 100
= 900

B = 2000

B = 2050

B = 2150

A + B = 3050

Conflict serializability ⇒ given ex:- schedule - 1 & schedule - 3

| $T_3$ | $T_4$ |
|---|---|
| read(Q) | |
| | write(Q) |
| write(Q) | |

→ It is not conflict serializable

It is called as Blind write operation

## View serializability :-

| $T_3$ | $T_4$ | $T_6$ |
|---|---|---|
| read(Q) | | |
| | write(Q) | |
| write(Q) | | |
| | | write(Q) |

The corresponding serial schedule of this conference schedule

→ There is no read(Q) operation after write(Q) that's why condition 2 is no application.

schedule 9 is view equivalent to the serial schedule

It is view equivalent schedule but not conflict serializable schedule

The schedule containing blild writes in view serializable schedule that is not conflict serializable

schedule 9 and <T3, T4, T6> are not conflict serializable bcs every pair of instruction are not conflict each other

## testing fore serializability:-

### steps

1) input is concurrent schedule S
2) output is whether the schedule S is conflict serializable are not.

### TASK

Construct the precedence graph fore schedule 1, schedule 2, schedule 3 and schedule 4.

Suppose initially A = $100 and B = $200

## Recovery system

Failure classification

1) Transaction failure :-
   a) → logical Error
   b) → system Error
2) system crash
3) Disk Failure

| | |
|---|---|
| read (A) | read C |
| A = A - 50 | C = C - 100 |
| write (A) | write (C) |
| read (B) | |
| B = B + 50 | |
| write (B) | |

A = 1000, B = 2000, C = 700

A = 950 ← 1000, B = 2050 ← 2000, C = 600 ← 700

## Checkpoints

→ When a system failure occures we must consult the log to determine those transaction that needs to be redone and those that need to be undone.

→ We need to search the entire log to decide the recovery procedure.

There are 2 major difficulties with this approach

1) The search process is very time consuming
2) The transaction that needs to be redone have already written their updates into the database so redoing such transaction causes the recovery to take longer time. so to reduce this type of overhead check-points are introduced.

During execution the system maintains the log using deffered database modification or immediate database modification

In addition the system periodically performs check points

which requires the following sequence of steps :-

step 1 :- output onto stable storage all log records currently residing in main memory.

step 2 :- output to the disk all modified buffer blocks

step 3 :- output onto the stable storage a log record ~~<~~ checkpoint ~~>~~