Date No

Query processing

(3) steps,

(01) Pharsing and translation
- output relational algebric expression

(02) optimization
- output query tree (execution plan)

(03) Evaluation.
- output of the query.

(01) Pharsing and translation.

ex :
```
Select e.name.
from employee e , works w
where e.eid = w.eid and e.age > 50 ;
```

(σ) → Select symbol with condition. table name. (condition)

⋈ → join tables. with join condition. (join)

I → Select command. (show the output)

(⋈ or *) can used.

$$\Pi_{e.ename}\left(works \bowtie_{e.eid = w.eid} \left(\sigma_{age>50}(emp)\right)\right)$$

↑
R.A.E (1)

~~$\Pi_{e.ename}\left(\sigma_{age>50}(emp)\right) works \bowtie e.eid = w.eid$~~

↑
R.A.E (2)

$$\Pi_{e.ename}\left(\sigma_{age>50 \text{ and } e.eid = w.eid}\left(works * emp\right)\right)$$

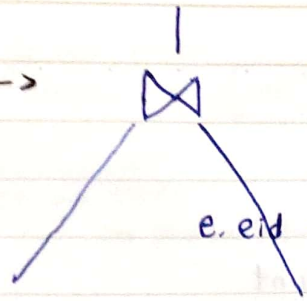
↑
R.A.E (3)

(02) Optimization.

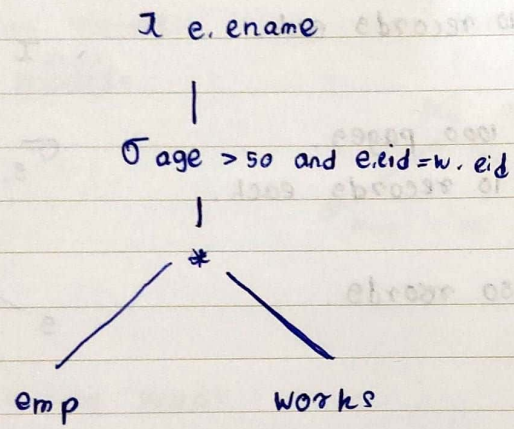
- R.A.E to query tree
- draw bottom to top
- ~~left to right~~ right to left.

Print output. → π e.ename.

Join table →



→ R, A, E (1)



→ R, A, E (2)

- best query tree selected based on cost.

(03) Evaluation.

- result and sql query.

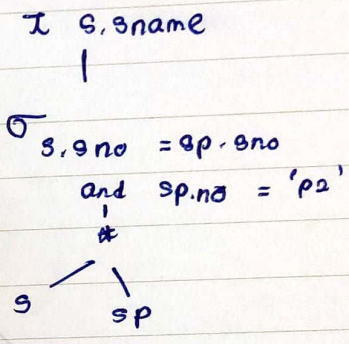
Calculating cost

need to know
- page count
- no of record in each page.
- no of records that match the condition.

ex: S table → 100 pages
10 records each.

Sp table → 1000 pages.
10 records each.

condition match → 50 records.



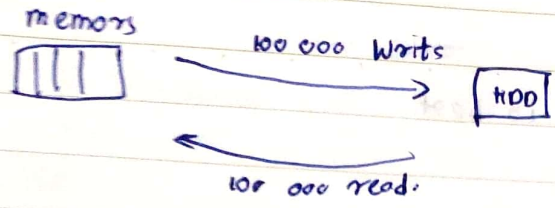
S table → 1st + 1000 Sp table
2nd + 1000 Sp table
⋮ ⋮
100 + 1000 Sp table.

} Join cost

Join cost = table 1 page count x table 2 page count

= 100 x 1000

= 100 000

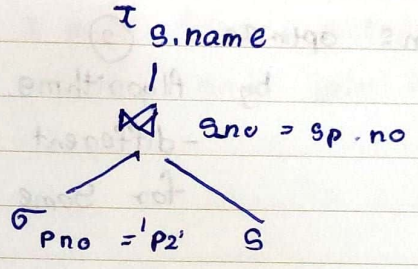


total will be 300000 reads and writes. (3m disk I/o)

According to R.A.E (2)

Sp = 10 000 ~~records~~ Pages

S = 100 ~~records~~ Pages



Cost = 10 000 reads

Sp table = 10 000 x 10 = 100 000

So records matched condition.

- only those 50 records will matched with S table

S ~~table~~ = 100 x 10
= 1000

Cost = 100 reads.

Cost = 10 000 + 100

= 10 1000 reads (output)

10 100 (I/o)

2 nd R, A, E is best

Heuristic optimization

- ~~2 ways~~ ①

  (01) remove cartisian join

  (02) Perform Selection befor join

Algorithms optimize ②

by Algorithms. (⋈ , I)

- different algorithms give different values for some symbol

Different algorithms

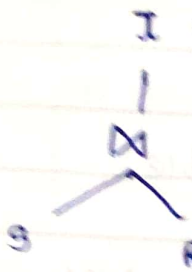
③ algorithms

1. SNLJ
2. PONLJ
3. BNLJ
4. INLJ
5. SMJ

- Will get different output for some SQL

(01) SNLJ (simple nested Loop join)

```
Select *
from Reserves R1 , Sailors S.
Where R1.sid = S1.sid.
```



S table – outer table
R table – inner table

500 pg
80 rec

1000 pg
100 rec

- first bring 1 page from outer table and in memory. Select first record and join it with all pages of inner table.

Equation of BNLJ

$$\text{Cost BNLJ} = M + (M \times P_R \times N)$$

M → pages of outer table
P_R → no of record in one page
N → page count of inner table.

S = 500 + (500 x 80 x 1000)

= 40 000 500 I/O

R = 1000 + (1000 x 100 x 500)

= 50 001 000 I/O

02 Page PNLJ (Page oriented loop join)

- less no of page table outer table.

- bring one page from outer table and join it with all the pages of inner table.

Cost PNLJ = M + M * N

M → page of outer
N → page of inner

$$9 = 500 + (500 \times 1000)$$

$$= 500\,500 \text{ I/o}$$

$$R = 1000 + (1000 \times 500$$

$$= 501\,000 \text{ I/o}$$

03 BNLJ (Block nested loop join)

- block is a set of pages. (collection of pages).
- We want to know exact block size.

block size = buffer size - 2

buffer size = 52.

$(2.3<) \rightarrow 2$

Date No

* block that we make using m pages $= \left[\frac{M}{B-2}\right]$ → ceiling value

Cost BNLJ = M + (# no of blocks x N)

$$= M + \left(\left[\frac{M}{B-2}\right] \times N\right)$$

M → outer page count ~~+ (no of~~

N → inner page count

→ block count.

B → buffer size

S table as outer $= M + \left(\frac{M}{B-2} \times N\right)$

$$= 500 + \left(\left[\frac{500}{52-2}\right] \times 1000\right)$$

$$= 500 + (10 \times 1000)$$

~~10~~ 10 500

$= \cancel{5\,000\,500}$ 1l0

* record to block cost is reducing a lot.

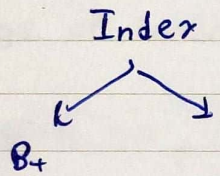
④ INLJ (Index nested loop join)

S table ⋈ R table

↑
Join

- check inner relation has index and it is the joining column.

S.a = R(b) → index column.



Cost INLJ = M + (m x P_S x Index Cost)

m → outer page count.

- no need to search in inner table.

Index cost

differ to the type of the index.

hash → 1.2
B+ tree → 2/3/4
height

} index cost

if not cost of index mentioned cost is (2)

Cost = M + (m x P_S x Index cost)

B+ = 500 + (500 x R_S^{80} x 2)

= 80 500 I/o

05 Sort merg join (smj) (R ⋈ S)

- first Sort the table and join
It have Sorting and merging cost. Sort using common Column.

cost = Sort R + Sort S + Merge cost

How to merge.

- Consider 1 record of table S and find matchin record in R table.
- like wise do it for each record.

Cost = O (M log M) + O (N log N) + (M+N)

- in exam they will give O (M log M) and O (N log N)

Date

Summery of algorithms and every processing.

σ → used in where conditions with table name.

*/ $\bowtie$ → join tables together

I → output

① table join cost = t1 page count x t2 page count

Algorithms (5)

(01) SNLJ (Simple nested loop join)

Cost SNLJ = $M + (M \times P_S \times N)$

(02) POLJ (Page oriented Loop join)

Cost = $M + M * N$

(03) Block nested Loop join (BNLJ)

① → block size = buffer size - 2

② → # no of blocks = $\left[\frac{M}{B-2}\right]$

③ → Cost = $M + \left(\left[\frac{M}{B-2}\right] \times N\right)$

Date No

④ INLJ (Index nested loop join)

hash → 1,2

B+ → 2, 3, 4 (default 2)

Cost = M + (M × P_s × Index cost)

⑤ Sort - merg join (R ⋈ S)

Cost = O(M log M) + O(N log N) + (M+N)

- they will give Sorting cost of both table.

M → outer page count.
N → inner page count.
P_s → no of record in one page.
→ block count.
B → buffer size.