UNIT-2 DBT : Important formulas & conditions

•> Sort Scan :

if clustered: B(R)
else: T(R)

V(R, a) -> Value set of Relation R for attr a.

•> || for π & σ in one pass algorithm.

•> δ & γ 1-pass
dup group

B(R) ≤ (M-1)

Disk I/O = B

If B(R) > M high cost to Evaluating.

for MAX, MIN, AVG : I/O same but now req lesser.

•> 1-pass R ∪ S & R ∩ S

min (B(S), B(R)) ≤ M-1

I/O : B(R) + B(S)

for Intersection it is most for larger Rel to be Swapped in M^th block for exhaustive Search.

•> Same as above for

R - S & R ⋈ S. & R × S

•> Nested Loop Join:

Using Smaller Rel on outer better

$\frac{B(S)}{(M-1)}$ # of iter of outer loop & B(R) of inner loop.

In each iter M-1 of S & B(R) of R are read.

$I/O = \{(M-1) * B(R)\} * \frac{B(S)}{(M-1)}$

- 2 pass Multi way merge sort:

  B(R) should not be greater than (M-1) sublist of each size M $\Rightarrow$ $\frac{B(R)}{M} \leq M \Rightarrow B(R) \leq M^2$

  $\Rightarrow \sqrt{B(R)} \leq M$.

  I/O : 3 B(R)

- Pop elim 2phase:

  I/O: 3 B(R) Same as sort.

  Mem : $M \geq \sqrt{B(R)}$

- U 2 phase : first have to sort sublist of R&S

  Load 1 block from each sublist & keeping smallest & keep cycling thru:

  I/O: $3 * (B(S) + B(R))$

  Mem : $B(R) + B(S) \leq M^2$

- 2phase $\cap$, - Same as above

- Sort based join 2Mpps

  first fully sort R&S

  use 2 buffers to find least match & write join.

  I/O: $5 (B(R) + B(S))$

  Mem: $\max(B(R), B(S)) \leq M^2$

- Sort - merge - join

  Only have to created sorted sublist & join on merge, leading to:

I/O: $3(B(R) + B(S))$ But

Mem: $(B(R) + B(S)) \leq M^2$

⇒ Hashing to M-1 blocks on disk adv:

$2\,B(R)$

⇒ Dup elim I/O = $3B(R)$

Same for grp agg

⇒ $\cup$, $\cap$, $-$

Hash R & S using Same, all related tuples in same bucket:

Cost = $3^*(B(R) + B(S))$

Mem: $\text{Min}(B(R), B(S)) \leq M^2$

⇒ Hash base join Same as above

⇒ Hybrid hash join

Mem: $\frac{M^* B(S)}{k} + (k-m) \leq M$.

I/O Saving = 2 per M $\quad k \approx \frac{B(S)}{M}$ $\quad \forall M=1$

$= 2\left(\frac{M}{k}\right)(B(R) + B(S))$

$= 2\left(\frac{M}{B(S)}\right)(B(R) + B(S))$

3- Saving

⇒ Index based Selection

I/O: $B(R)$ / $U(R,a)$ if clustered else:

$T(R)$ / "

•) Join

I/O: $T(R) * \frac{T(S)}{V(S,y)}$ or $T(R) * \frac{B(S)}{V(S,y)}$

•) Sorted index Join.

for sort 2 B(R) for sort on already sorted B-tree index. B(R) + B(S)