Query Optimization

Reduction Factor

1. Col $=$ value:

$$RF = 1/NKeys(Col)$$

2. Col $>$ value:

$$RF = (High(Col) - value)/(High(Col) - Low(Col))$$

3. Col $<$ value:

$$RF = (val - Low(Col))/(High(Col) - Low(Col))$$

4. ColA $=$ ColB (for joins): ??

$$RF = 1/(\max(NKeys(ColA), NKeys(ColB)))$$

5. No information about Nkeys or intervals

$$RF = 1/10$$

Result Size Estimation

1. Single table selection:

$$ResultSize = NTuples(R) \times \prod_{i=1,..,n} RF_i$$

2. Joins (over k tables):

$$ResultSize = \prod_{j=1,...,k} NTuples(R_j) \prod_{i=1,..,n} RF_i$$

if no selections (no predicates), $RF = 1$

Single-relation plans

1. Sequenctial (heap) scan:

$$Cost = NPages(R)$$

2. Index selection over a primary key (single tuple):

$$Cost(B + Tree) = Height(I) + 1$$

$$Cost(HashIndex) = ProbeCost(I) + 1, \quad ProbeCost(I) \approx 1.2$$

3. Clustered index matching one or more predicates:

$$Cost(B + Tree) = (NPages(I) + NPages(R)) \times \prod_{i=1,...n} RF_i$$

$$Cost(HashIndex) = 2.2 \times NPages(R) \times \prod_{i=1,...n} RF_i$$

4. Non-clustered index ...

$$Cost(B + Tree) = (NPages(I) + NTuples(R)) \times \prod_{i=1,...n} RF_i$$

$$Cost(HashIndex) = 2.2 \times NTuples(R) \times \prod_{i=1,\ldots n} RF_i$$

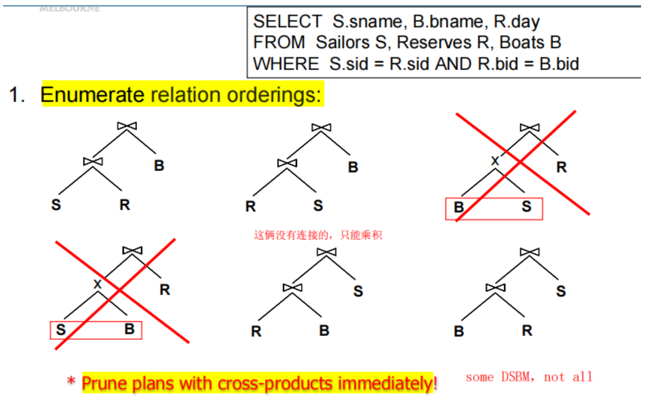
Multi-relation Plans

Step-by-step:

1. Select order of relations: $S \times R \times B,\ S \times B \times R, \ldots \quad \Rightarrow N!$
2. For each join, select join algorithm: Hash join, Sort-merge join...
3. For each input relation, select access method: Heap scan, various index alternatives..
4. Calculate ResultSize and Costs for each step, then compute the total cost

Onluy left-deep joint trees are considered: Intermediate results are not written to temporary files

Example:

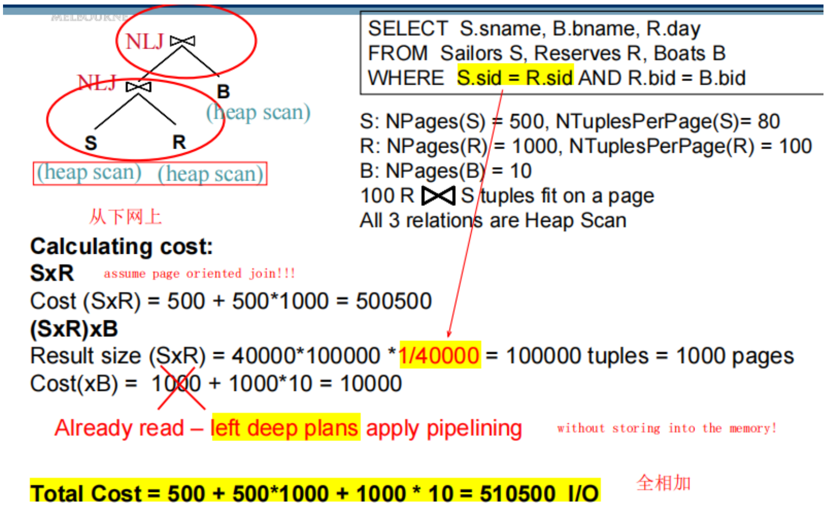


SELECT S.sname, B.bname, R.day
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid

1. Enumerate relation orderings:

* Prune plans with cross-products immediately!

some DSBM, not all



SELECT S.sname, B.bname, R.day
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid

S: NPages(S) = 500, NTuplesPerPage(S)= 80
R: NPages(R) = 1000, NTuplesPerPage(R) = 100
B: NPages(B) = 10
100 R ⋈ S tuples fit on a page
All 3 relations are Heap Scan

Calculating cost:

SxR assume page oriented join!!!

Cost (SxR) = 500 + 500*1000 = 500500

(SxR)xB

Result size (SxR) = 40000*100000 *1/40000 = 100000 tuples = 1000 pages

Cost(xB) = ~~1000~~ + 1000*10 = 10000

Already read – left deep plans apply pipelining without storing into the memory!

Total Cost = 500 + 500*1000 + 1000 * 10 = 510500 I/O 全相加