INFO20003 Database Systems

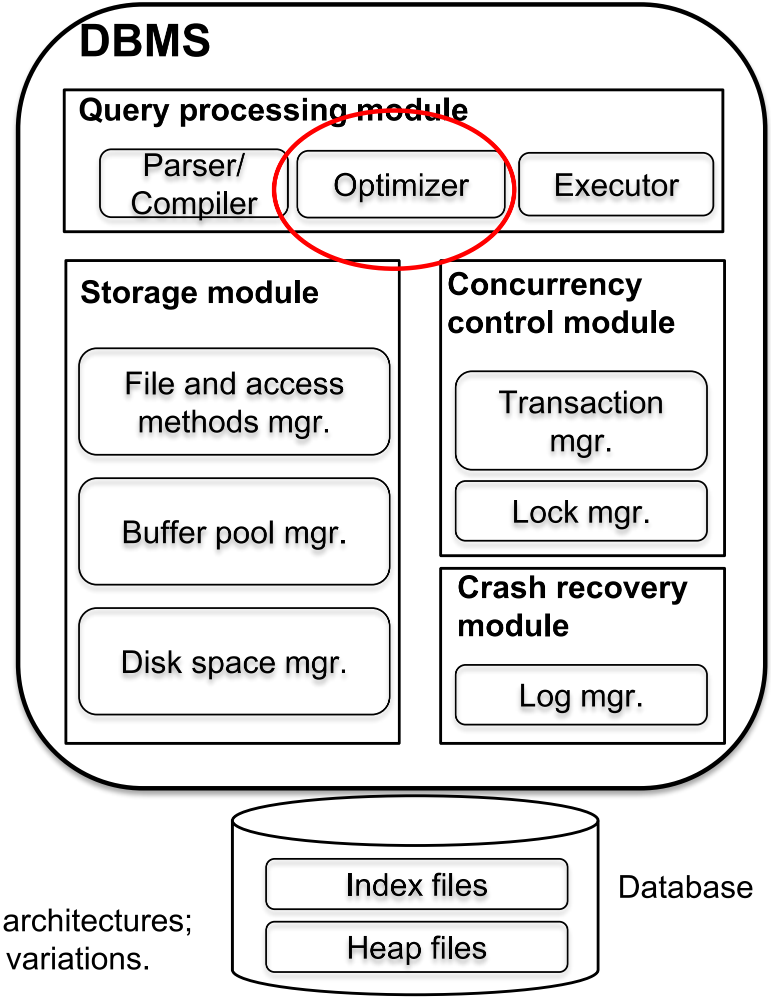
Dr Renata Borovica-Gajic

Lecture 14

Query Optimization Part II

Week 7

Remember this? Components of a DBMS



TODAY

Plan enumeration

This is one of several possible architectures; each system has its own slight variations.

Enumeration of Alternative Plans

- When enumerating alternative plans, there are two main cases:
  - Single-relation plans

    single table
  - Multiple-relation plans (joins)

- For queries over a **single relation:**

  ① heap scan ② index (especially when it has matching predicates)

  - Each *available* access path (file scan / index) is considered, and the one with the lowest estimated cost is chosen
    - Heap scan is always one alternative
    - Each index can be another alternative (if matching selection predicates)
  - Other operations can be performed on top of access paths, but they typically don't incur additional cost since they are done *on the fly* (e.g. projections, additional non-matching predicates)

Ex 1.

select

From A

where $a=5$ and $b>6$ and

$c<8$ and $d=10$;

$I_1(a,b)$ index $I_2(a,c,d)$ clustered

① Heap scan(A) = NPages(A) ✓

② $cost(I_1) = \underbrace{(NPages(I_1) + NPages(A))}_{\uparrow\ \text{cost for clustered index}} * RF(a) * RF(b)$ ✓

c & d will be checked on the fly

③ $cost(I_2) = (NPage(I_2) + NPages(A)) \times RF(a) \times RF(c) \times RF(d)$ ✓

Cost Estimates for Single-Relation Plans

1. Sequential (heap) scan of data file:

   Cost = NPages(R) ↑ or full table scan

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

   → cost of bringing the page which contain the data

   Cost(B+Tree)=Height(I)+1, Height is the index height

   Cost(HashIndex)= ProbeCost(I)+1, ProbeCost(I)~1.2

   → go to the bucket and find → bring the corresponding page

3. Clustered index matching one or more predicates:

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

   Cost(HashIndex)= NPages(R)$*\prod_{i=1..n} RF_i * \boxed{2.2}$ → 1.2+1 cost to find

4. Non-clustered index matching one or more predicates:

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

   Cost(HashIndex)= NTuples(R)$*\prod_{i=1..n} RF_i * 2.2$

Examples

Let's say that Sailors(S) has 500 pages, 40000 tuples, NKeys(rating) = 10

```
SELECT  S.sid FROM  Sailors S WHERE  S.rating=8
```

step 1

- Result size = (1/NKeys(rating)) * NTuples(S) = (1/10)*40000 =4000 tuples

1. If we have I(rating), NPages(I) = 50:
   - Clustered index:

     (50+500)× $\frac{1}{10}$ =55

     Cost = (1/NKeys(rating))*(NPages(I)+NPages(S))=(1/10)*(50+500) = 55 I/O
   - Unclustered index:

     (50+40000)× $\frac{1}{10}$

     Cost = (1/NKeys(rating))*(NPages(I)+NTuples(S))=(1/10)*(50+40000) = 4005 I/O
2. If we have an I(sid), NPages(I)= 50: no Reducting Factor => =1.
   - Cost = ?, Result size = ? 40000 tuple. In reality, such index will help us produce data in sorted order
   - Would have to retrieve all tuples/pages. With a clustered index, the cost is 50+500, with unclustered index, 50+40000 ✓
3. Doing a file scan: heap scan
   - Cost = NPages(S) = 500 ✓

Plan Enumeration for multi-relation plans

Steps:

1. Select **order** of relations
   - E.g. SxRxB, or SxBxR or RxSxB…
   - maximum possible orderings = N!
2. For each join, select **join algorithm**
   - E.g. Hash join, Sort-Merge Join…
3. For each input relation, select access method
   - Heap Scan, or various index alternatives

Q: How many plans are there for a query over N relations?

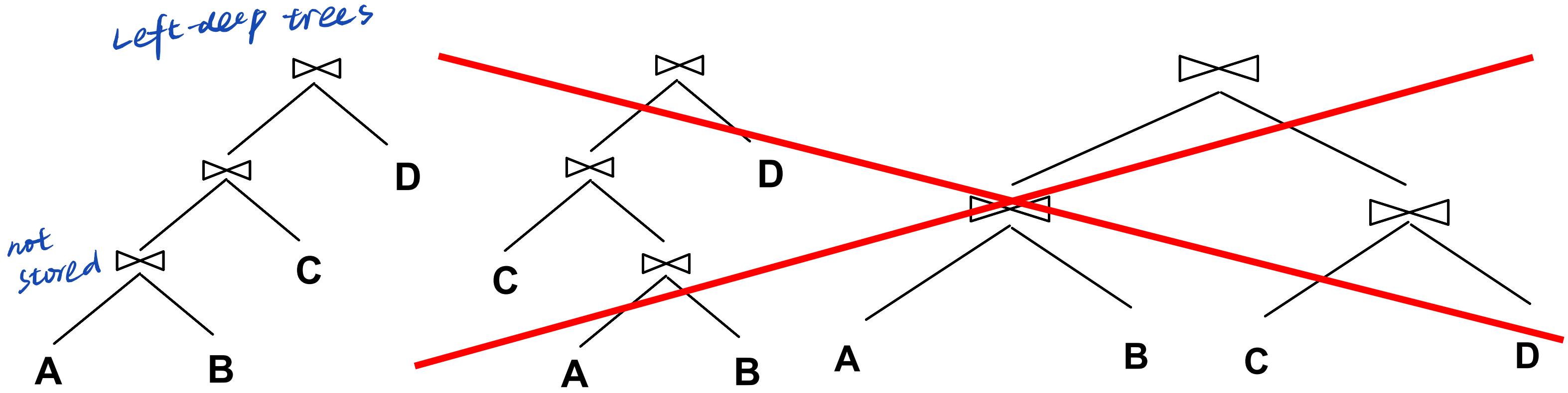
Back-of-envelope calculation:

- With 3 join algorithms, I indexes per relation:
  $\# \text{ plans} \approx [N!] * [\underset{\text{3 join algorithms}}{3}^{\overset{\text{\# of joins}}{(N-1)}}] * [(I + 1)^N] \xrightarrow{} \text{I index + 1 heap scan}$
- Suppose N = 3, I = 2: # plans ≈ $3! * 3^2 * 3^3 = 1458$ plans
- This is just for illustration – you don't need to remember this

Queries Over Multiple Relations

- As number of joins increases, number of alternative plans grows rapidly → *need to restrict search space*
- Fundamental decision in System R (first DBMS): **only left-deep join trees** are considered
  - Left-deep trees allow us to generate all *fully pipelined* plans
    - Intermediate results are not written to temporary files

don't store intermediate result, just push up



THE UNIVERSITY OF MELBOURNE

Plan Enumeration 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
```

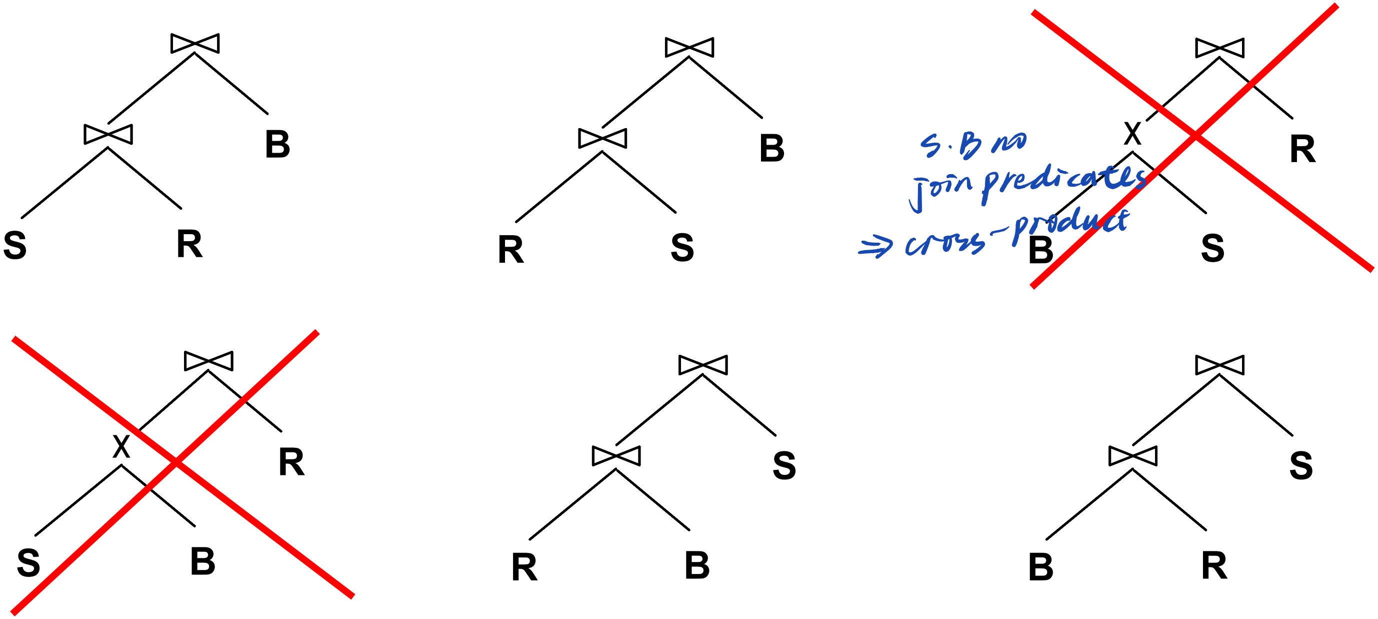
- Let's assume:
  - –Two join algorithms to choose from:
    - •Hash-Join
    - •NL-Join (page-oriented)
  - –Clustered B+Tree index: I(R.sid); NPages(I) = 50
  - –No other indexes
  - –S: NPages(S) = 500, NTuplesPerPage(S)= 80
  - –R: NPages(R) = 1000, NTuplesPerPage(R) = 100
  - –B: NPages(B) = 10
  - –100 R $\bowtie$ S tuples fit on a page

s

Candidate Plans

```
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:



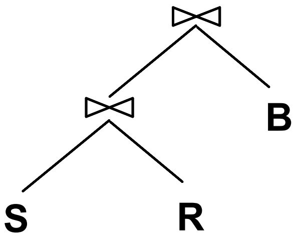
S.B no join predicates ⇒ cross-product

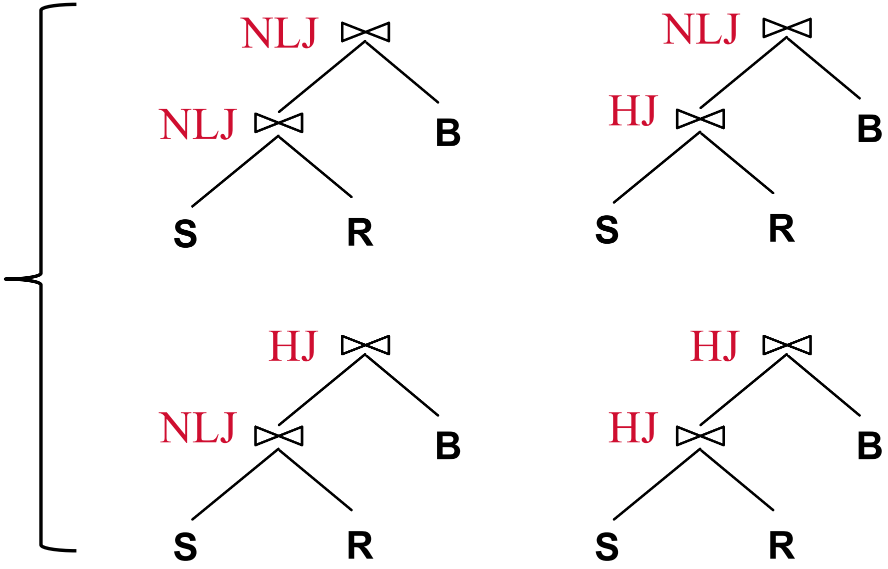
* Prune plans with cross-products immediately!

Candidate Plans

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

2. Enumerate join algorithm choices:



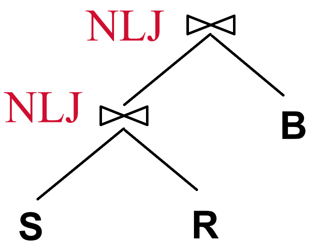


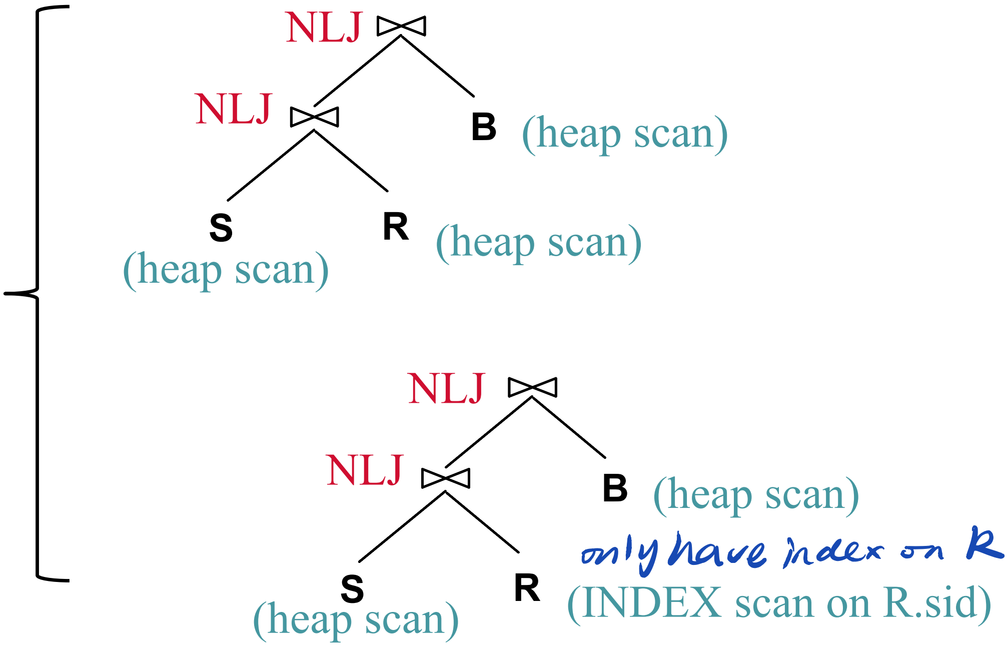
+ do the same for other plans

Candidate Plans

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

3. Enumerate access method choices:

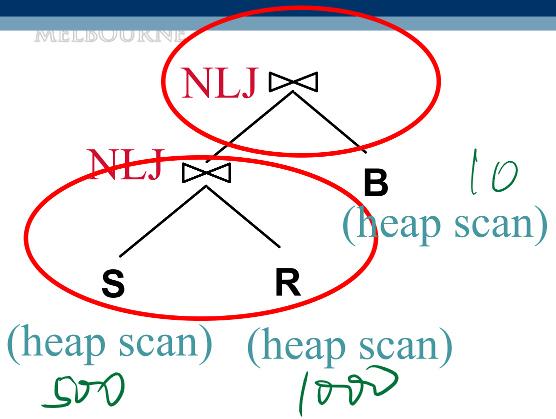




only have index on R

+ do same for other plans

Now estimate the cost of each plan



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 PNLJ.

Cost (SxR) = 500 + 500*1000 = 500500 s.sid=R.sid since sid is the primary key

(SxR)xB →(40,000+(100,000)) × $\frac{1}{40000}$ max(NKey(S), NKey(R)) so N keys = NTuple

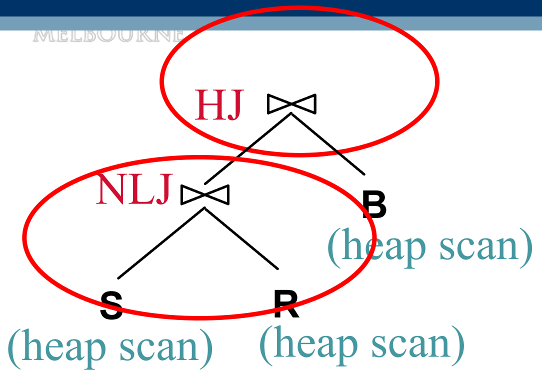
Result size (SxR) = 40000*100000 *1/40000 = 100000 tuples = 1000 pages ✓

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

Already read – left deep plans apply pipelining

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

Now estimate the cost of each plan



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

500 + 500 x1000 = 500500

~~40,000~~ x 100,000 x $\frac{1}{\cancel{40,000}}$ tuples
= 1000 page.

HJ : 2x1000 + 3x10
= 2030.

Calculating cost:

SxR

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

(SxR)xB

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

Cost(xB) = 3~~*1~~000 + 3*10 = 2*1000 + 3*10 = 2030

2

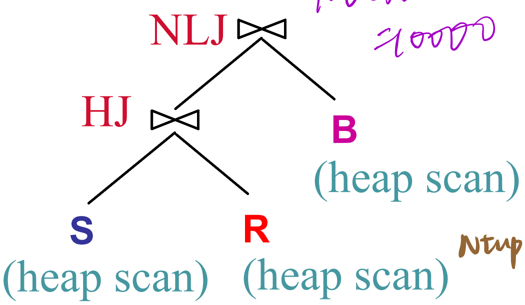
read data → write partition → read partition

Already read once – left deep plans apply pipelining

Total Cost = 500 + 500*1000 + 2*1000+ 3*10 = 502530 I/O

Your turn

Plan 3:



1000×10

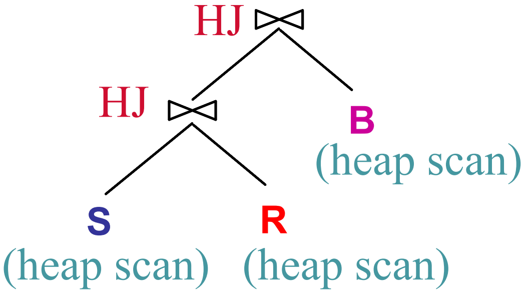
=10000

3×(1000+500)

Plan 4:

~~40000~~×100000 × $\frac{1}{4000}$

= 1000 page



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

R3. cost(S×R) = 3×500 + 3×1000 = 4500

Result size(S⋈R) = 40000 × 100000 × $\frac{1}{40000}$ = 100000.

⇒ 1000 pages

Ntuple(s) × Ntuple(R) × $\frac{1}{\text{NKeys(sid)}}$

cost(×B) = ~~1000~~ + 10×1000 = 10000

cost(P3) = 3×500 + 3×1000 + 10×1000 = 10000

Calculating cost:

Cost (P3) = ?

Cost (P4) = ?

R4. cost(S×R) = 3×500 + 3×1000 = 4500

Result size = 40000 × 100000 × $\frac{1}{40000}$ ⇒ 1000 page

cost(×B) = (3−1) × 1000 + 3×10 = 2030

cost(P4) = 3×1500 + 2×1000 + 3×10

At home

S: NPages(S) = 500, NTuplesPerPage(S)= 80

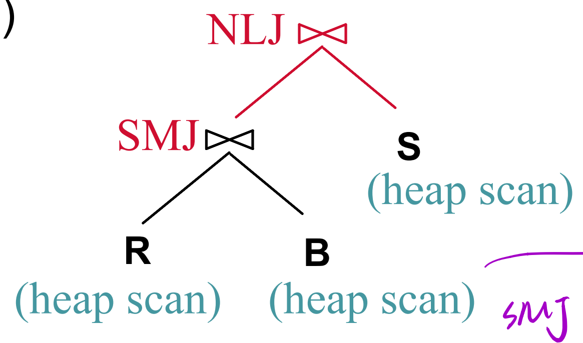
R: NPages(R) = 1000, NTuplesPerPage(R) = 100

B: NPages(B) = 10, NTuplesPerPage(B) = 10

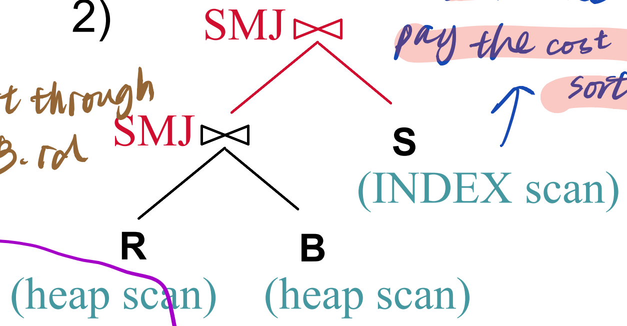
SMJ : 2 passes, RxB: 10 tuples per page

I(S.sid); NPages(I) = 50

1)



2)

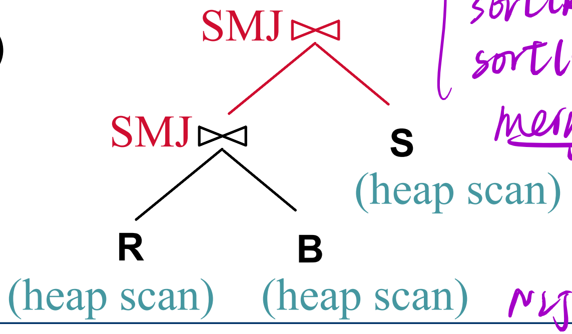


resort through s.id

sort through B.rid

sorted don't need to pay the cost of sorting

3)



SMJ

sort(R) = 2×2×1000 = 4000

sort(S) = 2×2× 10 = 40

merge = 1000+10 = 1010

→ 5050

resultsize.

100,000 × ~~100 × 1/100~~ = 10,000 Tuples = 10,000 page

NLJ 10000 × 500 = 5,000,000 → Total 5,005050

Plan 2.

SMJ 5050.

result size 10,000 pages

SMJ

sort(S) = 0 → read through index

sort(R×B)

$= 2 \times \underset{\substack{\uparrow \\ \text{passes}}}{2} \times 10000 = 40000$

SMJ 40000 + 500 + 50 = 40550

plan 3

SMJ $\underline{5050}$

result size 10,000 Page.

SMJ ② $2 \times 2 \times 10000 + 2 \times 2 \times 500$

$+ 500$

$= \underline{42500}$

47550

What’s examinable

- Understand plan enumeration and cost various plans

- Important for Assignment 3 as well

Next Lecture

- Normalization