
# INFO20003 Database Systems



Dr Renata Borovica-Gajic



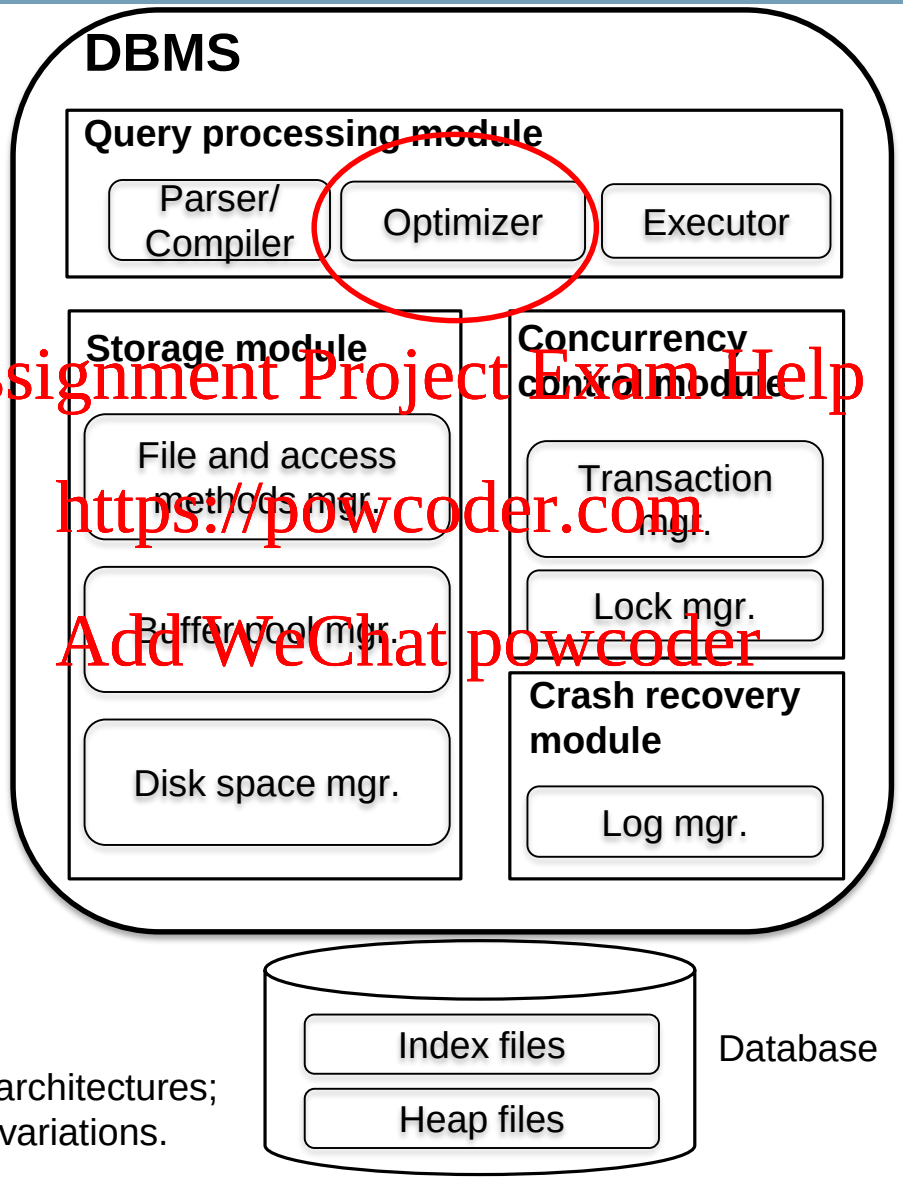
Lecture 14
Query Optimization Part II

Semester 2 2018, Week 7

# MST

- **When**: Tuesday 11/09/2018 @ 2:45 – 4:00pm
  - Seated at 2:45pm, the test starts at 3:00pm and runs for 45min
- **Where**: Wilson Hall
- **Rules**:
  - Bring student ID and pen/pencil (no cheat sheet)
  - Cover [Lec1, Lec9]: Modelling, Relational Algebra, SQL
- Best way to prepare:
  - There is a sample test in "Practice on your own"
  - Look at mistakes you've made in Assignment 1 and learn from them
  - Solution for Assignment 1 is in Resources
  - Attempt first 5,6 questions of Assignment 2 (maximize the effort) and look at the provided labs material plus the given practice study (in Practice on your own)
  - For simple joins with filter predicates try to express them with RA

# 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
  - Multiple-relation plans (joins)



- For queries over a single relation:
  - 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)

# Cost Estimates for Single-Relation Plans

1. Sequential (heap) scan of data file:
   **Cost = NPages(R)**

2. Index selection over a primary key (just a single tuple):
   **Cost(B+Tree)=Height(I)+1,** Height is the index height
   **Cost(HashIndex)= ProbeCost(I)+1,** ProbeCost(I)~1.2



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)*** $\prod_{i=1..n} RF_i * 2.2$



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
```

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

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

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

     Cost = (1/NKeys(I))*(NPages(I)+NTuples(S))=(1/10)*(50+40000) = 4005 I/O
2. If we have an I(sid), NPages(I)= 50:
   - Cost = ?, Result size = ?
   - 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:
   - Cost = NPages(S) = 500

# Plan Enumeration

**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?

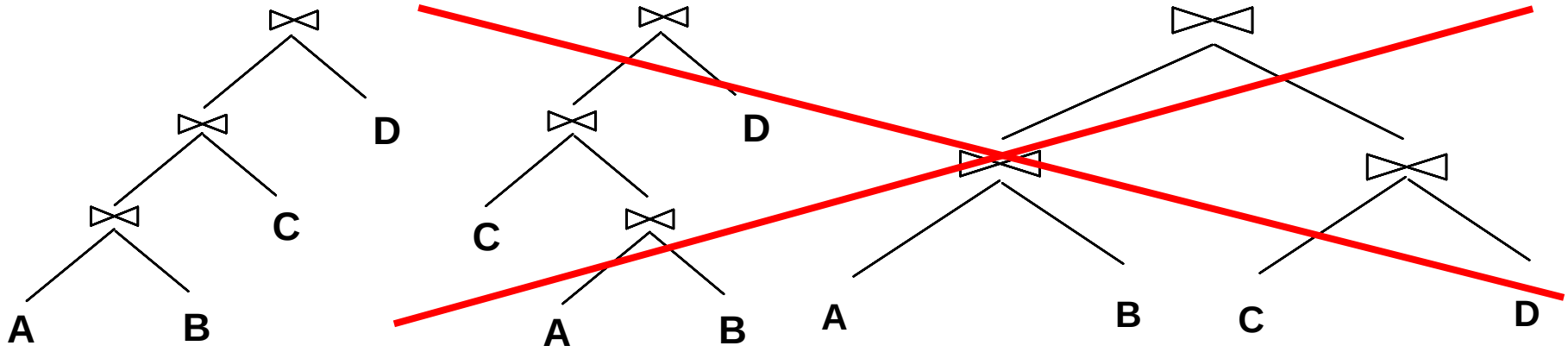
Back-of-envelope calculation:

- With 3 join algorithms, I indexes per relation:
  # plans ≈ [N!] * [$3^{(N-1)}$] * [$(I + 1)^N$]
- 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





# Plan Enumeration Example

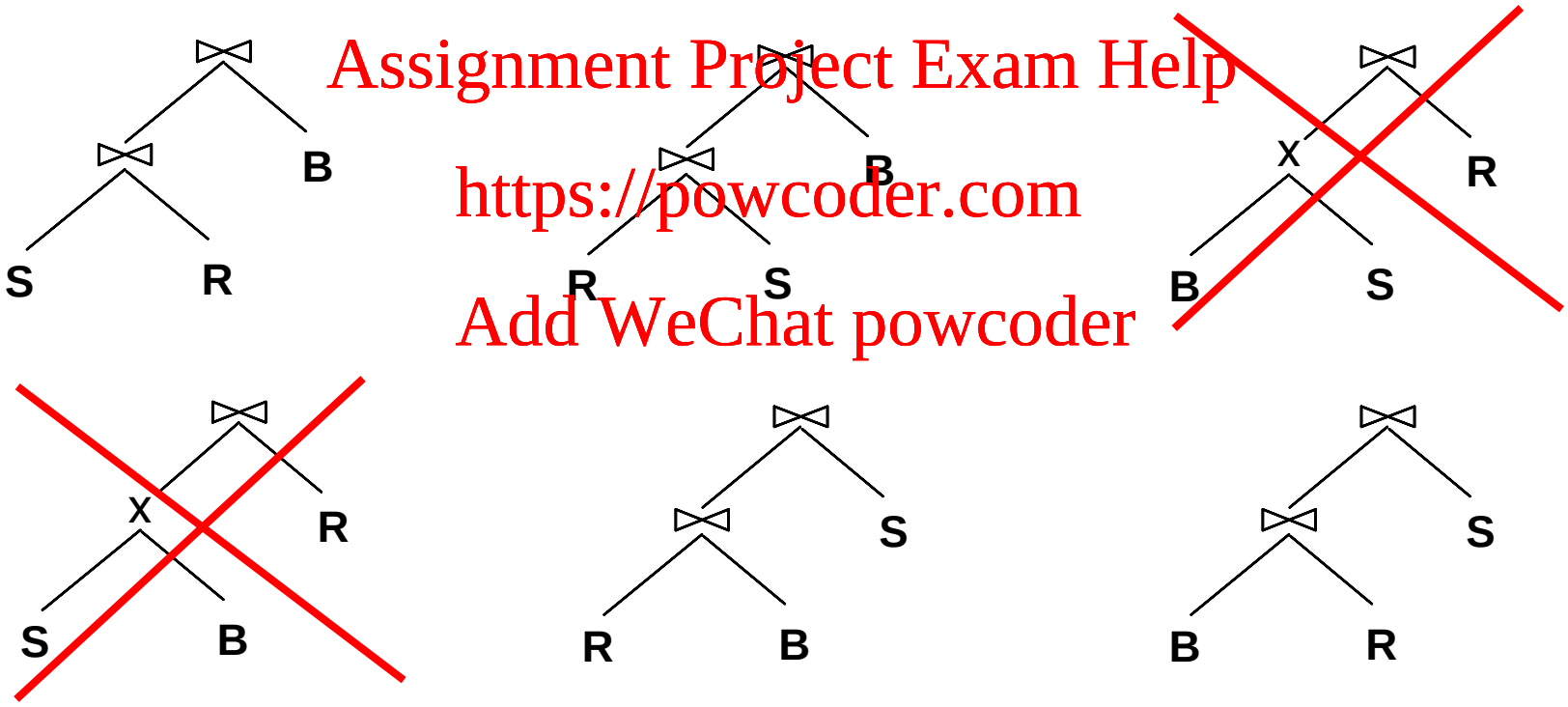
```
SELECT  S.sname, B.bname, R.day
FROM  Sailors S, Reserves R, Boats B
WHERE  S.sid = R.sid AND R.bid = B.bid
```

- Let's assume:
  - Two join algorithms to choose from:
    - Hash-Join
    - NL-Join (page-oriented)
  - Unclustered 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

# 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:



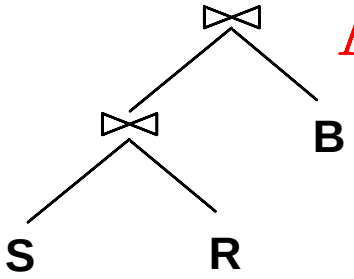



* Prune plans with cross-products immediately!

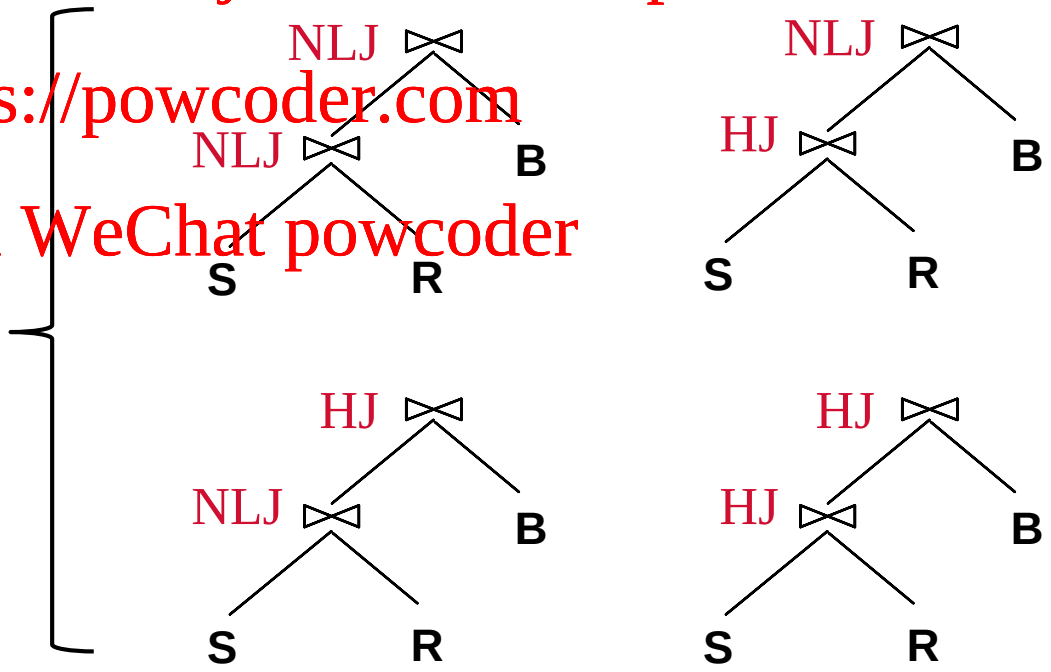
# Candidate Plans

```
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:





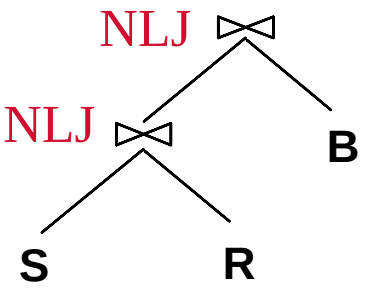




+ do the same for other plans

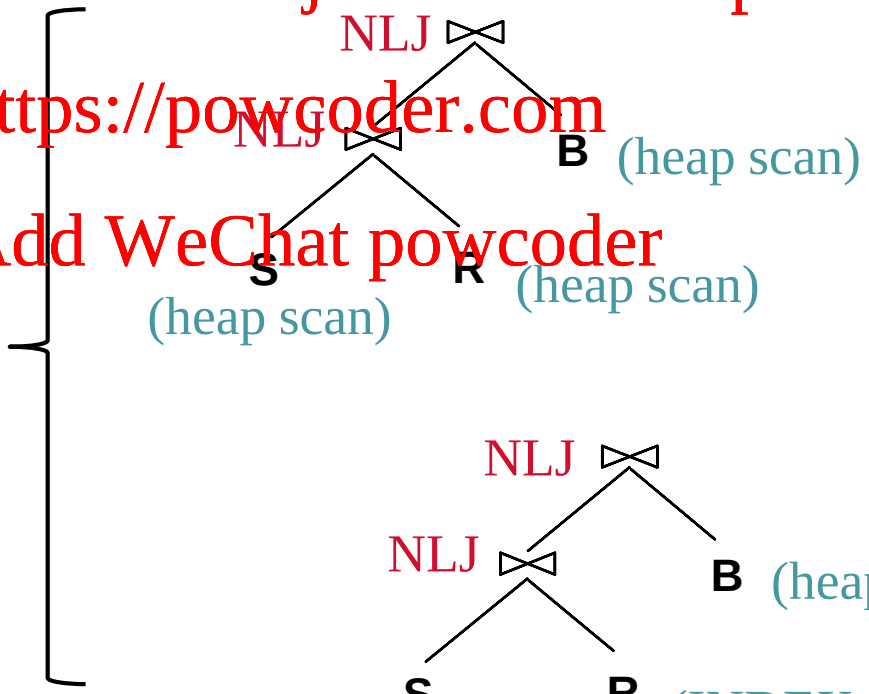
# Candidate Plans

```
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:

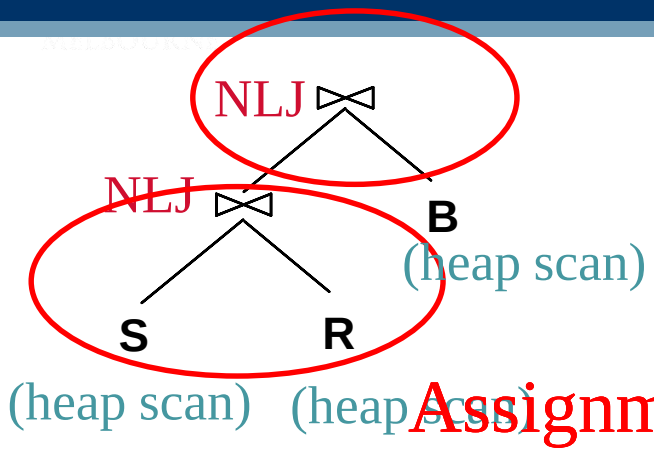





+ do same for other plans

# 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



**Calculating cost:**

**SxR**

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

**(SxR)xB**

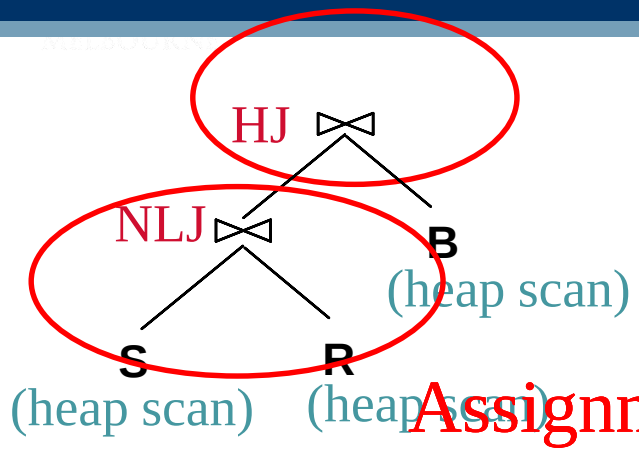
Result size (SxR) = 100000*40000 *1/40000 = 100000 tuples = 1000 pages

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

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



**Calculating cost:**

**SxR**

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

**(SxR)xB**

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

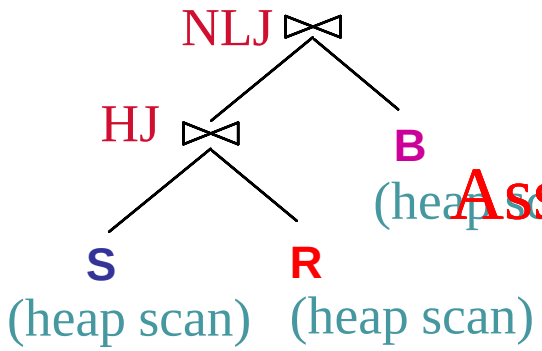
Cost(xB) = ~~3*1000~~ 2 + 3*10 = 2*1000 + 3*10 = 2030

Already read once – left deep plans apply pipelining

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

# Your turn

## Plan 3:



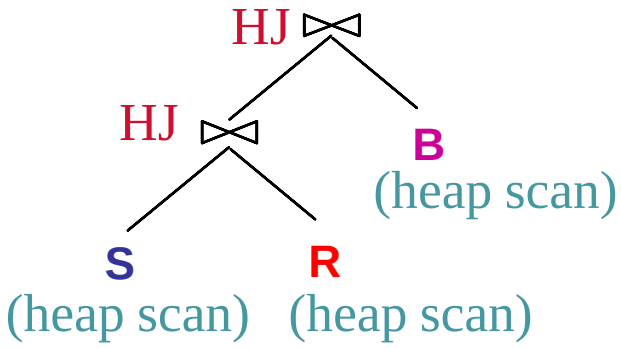

S: NPages(S) = 500, NTuplesPerPage(S)= 80
R: NPages(R) = 1000, NTuplesPerPage(R) = 100
B: NPages(B) = 10
100 R ⋈ S tuples fit on a page
All 3 relations are Heap Scan



**Calculating cost:**

Cost (P3) = ?

## Plan 4:

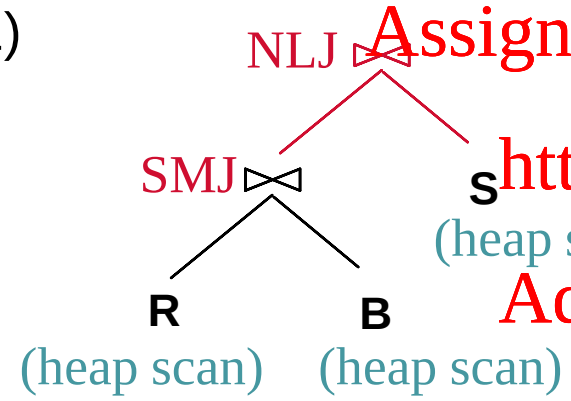



Cost (P4) = ?

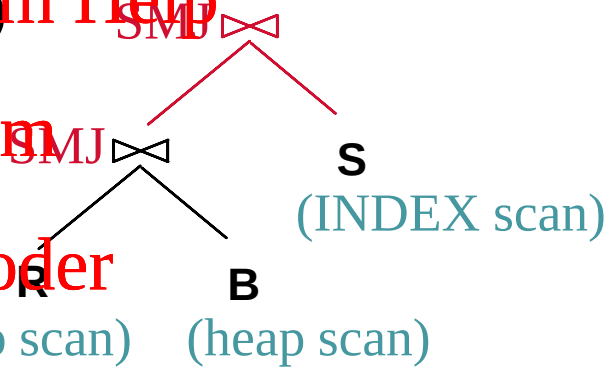
# At home

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

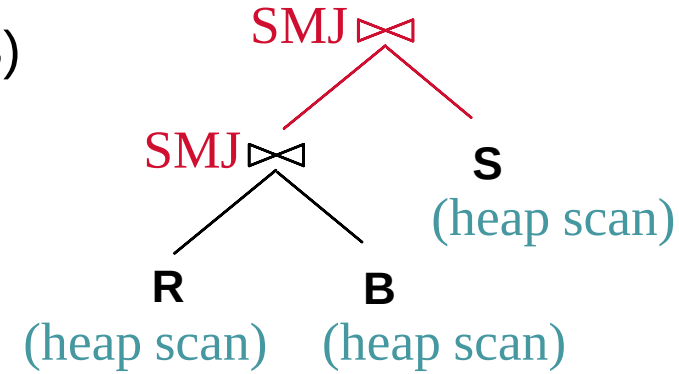
1)




2)




3)



# What's examinable

- Understand plan enumeration and cost various plans

- Important for Assignment 3 as well

# Next Lecture

- Normalization