## Query Optimization Example

- Sailors (<u>sid</u>, sname, rating, age)
- Boats(<u>bid</u>, bname, color)
- Reserves(<u>sid, bid, day</u>, rname)
- Query:

```
SELECT  S.sid, S.sname, S.age
FROM    Sailors S, Boats B, Reserves R
WHERE   B.bid = R.rid AND B.bid = R.bid AND
        B.color = “Red” AND S.age < 30;
```

- Reserves has 1000 pages, 10 tuples/page
- Sailors has 500 pages, 50 tuples/page
- Boats has 160 pages, 10 tuples/page
- Data is evenly distributed (assumption)



## Steps

- Query optimization steps:
  - Translate SQL Query to Relational Algebra Query
  - Create a query tree
  - Create left-deep alternative trees
  - Create query plans for our trees
  - Estimate costs of plans
  - Pick best one

## Step 1

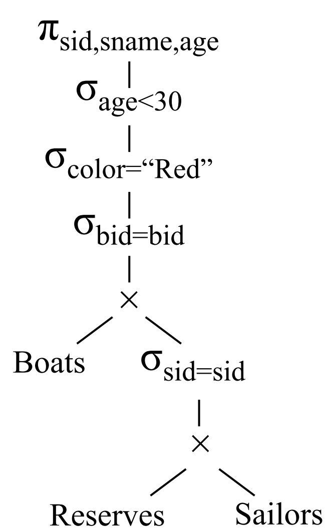
- Translate SQL Query to Relational Algebra Query

$$\pi_{sid,\ sname,\ age}(\sigma_{age<30}(\sigma_{color=\text{"Red"}}(\sigma_{bid=bid}(B \times \sigma_{sid=sid}(S \times R)))))$$



## Step 2

- Create a query tree

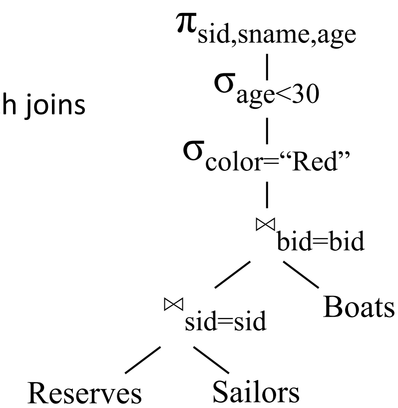



Note our query only involves joins (as opposed to plain old cross-products) …

Lets draw the trees with joins

## Step 3

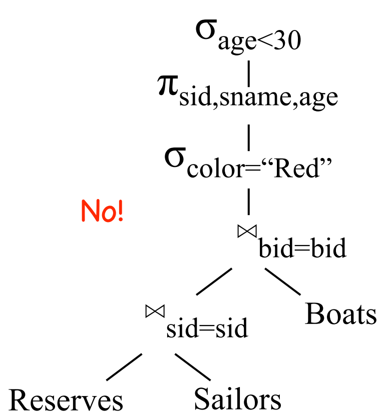
- Create left-deep alternative trees
  - We replace cross products if they are really joins
  - How?
    - Using RA equivalences!
  - This is a left-deep plan with joins



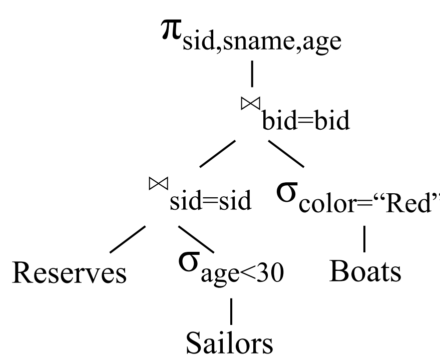



## Step 3 – More Alternatives

- Would we expect these to have different costs over previous alternative?




No!




Yes!

## Step 4 & 5 – Plans and Estimates

- Lets create some plans and estimates

Total cost: 160,160 + 1,002,000 = 1,162,160 I/Os !!!

M + M*N = ?+?*500 = ???
- How large is the result of the first join?
- Every reservation has a bid
- bid is unique for each boat
- So 10,000 tuples
- How many pages?
- 1 / ((1/10) + (1/10)) = 10/2 = 5 tuples/page

M + M*N = 2,000+2,000*500 = 1,002,000

M + M*N = 160+160*1,000 = 160,160
*(this is the cost of reading, not writing)*

$\pi_{\text{sid,sname,age}}$ (On-the-fly)

$\sigma_{\text{color=“Red”} \wedge \text{age<30}}$ (On-the-fly)

$\bowtie_{\text{sid=sid}}$ (Page NL)

Sailors

$\bowtie_{\text{bid=bid}}$ (Page NL)

Boats Reserves



## Step 4 & 5 – Plans and Estimates

- Assume Boats has a clustered B+ Tree on bid (key)
- Assume S has a clustered B+ Tree on sid (key)

Total cost: 5,160 + 8,500 = 13,660 I/Os !!!

Sort(M) + Sort(N) + M + N =
6,000 + 0 + 2,000 + 500 = 8,500

Result must be sorted on sid:
- 2*2,000 + 2*1*2,000 = 6,000 I/Os

Sort(M) + Sort(R) + M + N =
0 + 4,000 + 160 + 1,000 =
5,160

Boats sorted on bid, Reserves isn't
Assume buffers B = 50
Cost of sorting reserves:
- 2*1,000 + 2*1*1,000 = 4,000 I/Os

$\pi_{\text{sid,sname,age}}$ (On-the-fly)

$\sigma_{\text{color=“Red”} \wedge \text{age<30}}$ (On-the-fly)

$\bowtie_{\text{sid=sid}}$ (Sort Merge)

Sailors

$\bowtie_{\text{bid=bid}}$ (Sort Merge)

Boats Reserves

* We are assuming merge (in Sort Merge) takes M + N … this may not always be the case. Why?

## Note on Sort Merge

*This is a much better plan than our previous page nested loops one*

- But it is an *__overestimate__* for Sort-Merge join!
  - In Pass 0 we read all pages then write all pages (2*M)
  - We do not need to read all pages if they are pipelined to the next join (1*M)
  - In last merge pass, we don't have to write the last set of pages since we always pipeline them to the next operator
  - We also don't have to read in the file again:
    - Sort(M) + Sort(N)  if N needs to be sorted
    - Sort(M) + N  if N is sorted



## Step 4 & 5 – Plans and Estimates

- Assume Boats has a clustered B+ Tree on bid (key)
- Assume S has a clustered B+ Tree on sid (key)

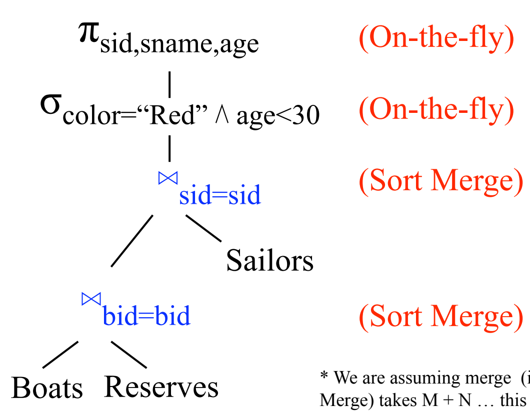
Total cost: 3,160 + 4,500 = 7,660 I/Os !!!

Sort(M) + N = 4,000 + 500 = 4,500

Result must be sorted on sid:
• 1*2,000 + 1*1*2,000 = 4,000 I/Os

Sort(M) + N = 3,000 + 160 = 3,160

Boats sorted on bid, Reserves isn't
Assume buffers B = 50
Cost of sorting reserves:
• 2*1,000 + 1*1*1,000 = 3,000 I/Os




* We are assuming merge (in Sort Merge) takes M + N … this may not always be the case. Why?

## Step 4 & 5 – Plans and Estimates

- Lets try block nested loop join (B=50)

Total cost: 4,160 + 22,500 = 26,660 I/Os !!!

$\pi_{\text{sid,sname,age}}$ (On-the-fly)

$\sigma_{\text{color="Red"} \wedge \text{age<30}}$ (On-the-fly)

M + (M/(B-1))*N = 2,000 + 41*500 = 22,500 — $\bowtie_{\text{sid=sid}}$ (Block NL)

Sailors

M + (M/(B-1))*N = 160+4*1,000 = 4,160
*(this is the cost of reading, not writing)* — $\bowtie_{\text{bid=bid}}$ (Block NL)

Boats Reserves



## Step 4 & 5 – Plans and Estimates

- Can we do better?

Total cost: 5,410 + 11,500 = 16,910 I/Os !!!

$\pi_{\text{sid,sname,age}}$ (On-the-fly)

$\sigma_{\text{color="Red"} \wedge \text{age<30}}$ (On-the-fly)

M + (M/(B-1))*N = 1,250 + 26*160 = 5,410 — $\bowtie_{\text{bid=bid}}$ (Block NL)

How many pages in the answer?
1,250 … why?

Boats

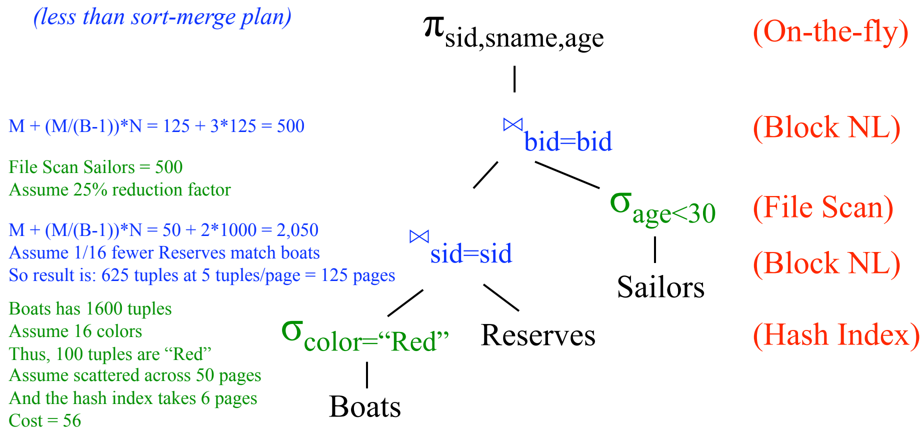
M + (M/(B-1))*N = 1000+21*500 = 11,500
*(this is the cost of reading, not writing)* — $\bowtie_{\text{sid=sid}}$ (Block NL)

Reserves Sailors

## Step 4 & 5 – Plans and Estimates

- Can we do even better?
  - Assume Boats has a hash index on color

Total cost: 56 + 2,050 + 500 + 500 = 3,106 I/Os !!!
*(less than sort-merge plan)*






## Step 4 & 5 – Plans and Estimates

- Can we still do better?
  - Yes!
  - E.g., we could project on bid after first join …
- Notice that by adding an index on color to Boats …
  - We reduced our best query time in half!
  - This is one reason we talk about query optimization
  - It drives physical database design
  - Improving performance (by adding indexes, e.g.) should be driving on how query optimization works!