COMP 3311 DATABASE MANAGEMENT SYSTEMS

LECTURE 18 EXERCISES QUERY OPTIMIZATION

EXAMPLE RELATIONAL DATABASE

Sailor(<u>sailorId</u>, sName, rating, age)   Boat(<u>boatId</u>, bName, color)

Reserves(<u>sailorId, boatId, rDate</u>)

- There are 10,000 Sailor tuples, 100,000 Reserves tuples and 1,000 Boat tuples.
- Assume that for all files there are 10 tuples per page.
  ☞ B_{Sailor} = 1,000 pages; $B_{Reserves}$ = 10,000 pages; B_{Boat} = 100 pages
- Assume a main memory buffer, *M*, of 100 pages.
- Assume that there are the following indexes:
  – hash index on sailorId for Sailor (no overflow buckets).
  – clustering B+-tree index on rDate for Reserves (2 levels).
  – hash index on boatId for Boat (no overflow buckets).
- Our goal is to process the query:

```
select *
from Sailor natural join Reserves natural join Boat
where rDate = '01-JAN-2020'
        and color = 'red';
```

Some useful statistics:

- Reserves has 1,000 unique rDates.
- 10% of boats are red.
- A sailor has on average 10 reservations.

EXERCISE 1

Estimate the query processing cost using materialization and the join order

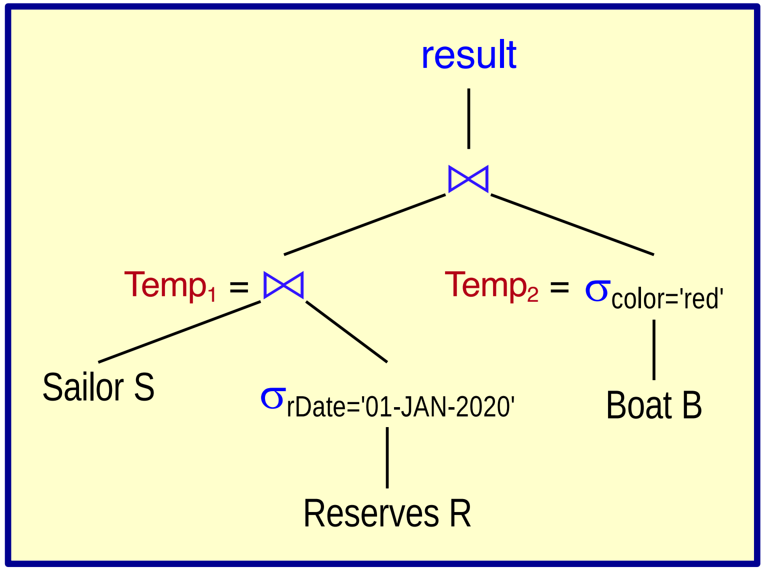
(Sailor JOIN $\sigma_{R.rDate='01-JAN-2020'}$Reserves) JOIN $\sigma_{color='red'}$Boat

C_1: Cost of computing $Temp_1$ = (Sailor JOIN $\sigma_{R.rDate='01-JAN-2020'}$Reserves)

C_2: Cost of computing $Temp_2$ = $\sigma_{color='red'}$Boat (no index on color)

C_3: Cost of $Temp_1$ JOIN $Temp_2$

Total cost: $C_1 + C_2 + C_3$



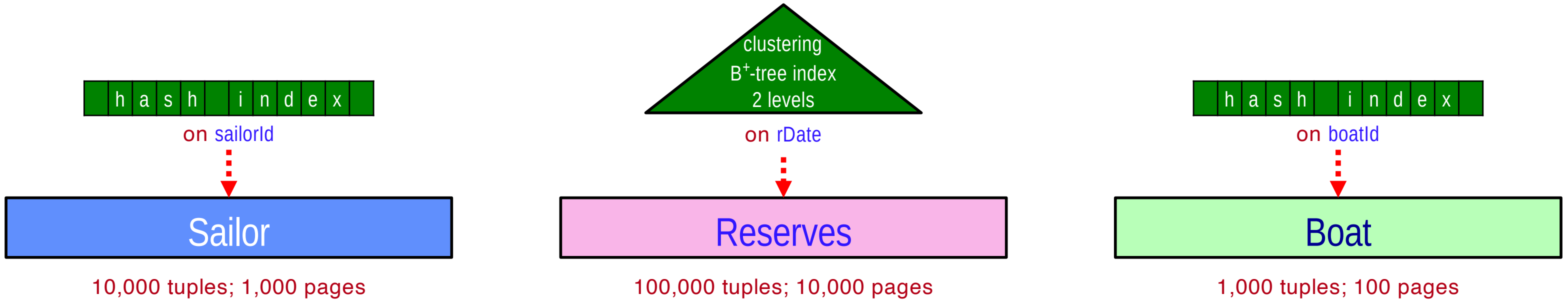
EXERCISE 1 (CONT'D)

bf: 10 tuples/page
B_{Sailor}: 1,000 pages
$B_{Reserves}$: 10,000 pages
B_{Boat}: 100 pages
M: 100 pages
Hash index: Sailor.sailorId
B^+-tree index: Reserves.rDate (2 levels)
Hash index: Boat.boatId
Unique Reserves rDates: 1000
Red boat reservations: 10%
Reservations/sailor: 10

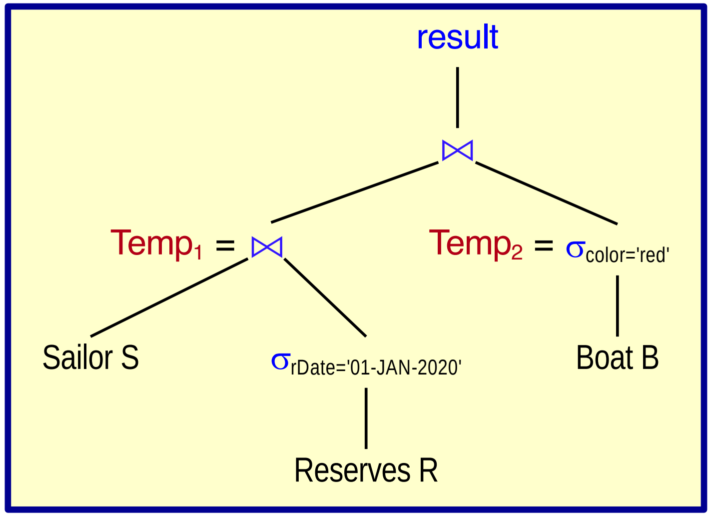
Sailor(<u>sailorId</u>, sName, rating, age)

Boat(<u>boatId</u>, bName, color)

Reserves(<u>sailorId, boatId, rDate</u>)




```
select *
from Sailor natural join Reserves natural join Boat
where rDate = '01-JAN-2020'
      and color = 'red';
```



Some useful statistics:

- Reserves has 1,000 unique rDates.
- 10% of boats are red.
- A sailor has on average 10 reservations

```
select *
from Sailor natural join Reserves natural join Boat
where rDate = '01-JAN-2020'
      and color = 'red';
```

EXERCISE 1 (CONT'D)

bf: 10 tuples/page
B_{Sailor}: 1,000 pages
$B_{Reserves}$: 10,000 pages
B_{Boat}: 100 pages
M: 100 pages
Hash index: Sailor.sailorId
B+-tree index: Reserves.rDate (2 levels)
Hash index: Boat.boatId
Unique Reserves rDates: 1000
Red boat reservations: 10%
Reservations/sailor: 10

C_1: Cost of computing $Temp_1$ = (Sailor JOIN $\sigma_{R.rDate='01\text{-}JAN\text{-}2020'}$Reserves)

In order to estimate C_1, we need to determine the best evaluation plan (sub-plan) for computing $Temp_1$.

Some alternative join strategies:

a) block nested-loop – Sailor as outer relation: $\lceil B_r / (M-2) \rceil * B_s + B_r$.

b) block nested-loop – Reserves as outer relation: $\lceil B_r / (M-2) \rceil * B_s + B_r$.

c) sort-merge join: $B_r + B_s$ + sorting cost: $2*B_r*(1+\lceil \log_{M-1}(B_r/M) \rceil)$.

d) hash join: $3 * (B_r + B_s)$.

e) indexed nested-loop – Reserves as outer relation: $B_r + n_r * c$.

```
select *
from Sailor natural join Reserves natural join Boat
where rDate = '01-JAN-2020'
      and color = 'red';
```

EXERCISE 1 (CONT'D)

B_{Sailor}: 1,000 pages
bf: 10 tuples/page
$B_{Reserves}$: 10,000 pages
B_{Boat}: 100 pages
M: 100 pages
Hash index: Sailor.sailorId
B+-tree index: Reserves.rDate (2 levels)
Hash index: Boat.boatId
Unique Reserves rDates: 1000
Red boat reservations: 10%
Reservations/sailor: 10

C_1: Cost of computing $Temp_1$ = (Sailor JOIN $\sigma_{R.rDate='01\text{-}JAN\text{-}2020'}$Reserves)

a) **Strategy:** block nested-loop – Sailor as outer relation.

Cost: $\lceil B_r / (M-2) \rceil * B_s + B_r = \lceil 1000 / (100-2) \rceil * 10000 + 1000$

$= \underline{110{,}100}$ page I/Os

b) **Strategy:** block nested-loop – Reserves as outer relation.

Cost: $\lceil B_r / (M-2) \rceil * B_s + B_r = \lceil 10000 / (100-2) \rceil * 1000 + 10000$

$= \underline{113{,}000}$ page I/Os

c) **Strategy:** sort-merge join: $B_r + B_s$ + sorting cost.

Sort cost Sailor: $2*B_r*(1+\lceil \log_{M-1}(B_r/M) \rceil)$

$= 2 * 1000 * (1+\lceil \log_{100-1}(1000/100) \rceil) = 6000$

Sort cost Reserves: $2*B_r*(1+\lceil \log_{M-1}(B_r/M) \rceil)$

$= 2 * 10000 * (1+\lceil \log_{100-1}(10000/100) \rceil) = 20000$

Join cost: $B_r + B_s = 1000 + 10000 = 11000$

Cost: $6000 + 20000 + 11000 = \underline{37{,}000}$ page I/Os

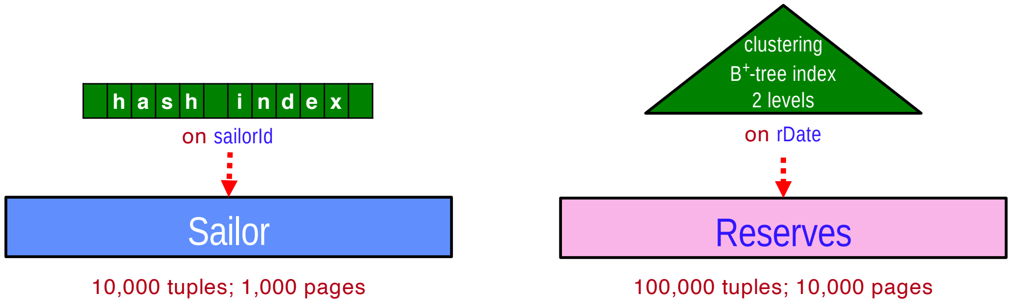
```
select *
from Sailor natural join Reserves natural join Boat
where rDate = '01-JAN-2020'
      and color = 'red';
```

EXERCISE 1 (CONT'D)

bf: 10 tuples/page
B_{Sailor}: 1,000 pages
$B_{Reserves}$: 10,000 pages
B_{Boat}: 100 pages
M: 100 pages
Hash index: Sailor.sailorId
B+-tree index: Reserves.rDate (2 levels)
Hash index: Boat.boatId
Unique Reserves rDates: 1000
Red boat reservations: 10%
Reservations/sailor: 10

d) **Strategy:** hash join: 3 * (B_r + B_s).
Cost: 3 * (1000 + 10000) = 33,000 page I/Os

e) **Strategy:** indexed nested-loop – Reserves as outer relation.



Sailor contains a hash index on the join attribute sailorId. Furthermore, we have a selective condition ($\sigma_{rDate='01\text{-}JAN\text{-}2020'}$Reserves) and a clustering index on Reserves.rDate.

How many reservations do we need to access?

Since there are 100,000 reservations and 1,000 unique rDates, on a given rDate there are 100000 / 1000 = 100 reservations. Therefore, we need to access 100 Reserves tuples for 01-JAN-2020.

```
select *
from Sailor natural join Reserves natural join Boat
where rDate = '01-JAN-2020'
      and color = 'red';
```

EXERCISE 1 (CONT'D)

bf: 10 tuples/page
B_{Sailor}: 1,000 pages
$B_{Reserves}$: 10,000 pages
B_{Boat}: 100 pages
M: 100 pages
Hash index: Sailor.sailorId
B^+-tree index: Reserves.rDate (2 levels)
Hash index: Boat.boatId
Unique Reserves rDates: 1000
Red boat reservations: 10%
Reservations/sailor: 10

How many pages need to be accessed, to find these reservations?

The index on Reserves.rDate has 2 levels and is ordered on rDate. Since, 10 Reserves tuples fit per page, to retrieve all reservations 2 index pages and 10 Reserves pages need to be accessed = 12 page I/Os. (Assumes all 100 Reserves tuples in the result are in 2 pages.)

For each of the 100 Reserves tuples, the corresponding Sailor tuple is retrieved using the hash index on Sailor.sailorId, with cost 2 * 100 = 200 page I/Os. The join result will contain 100 tuples.

Cost: 12 + 200 = 212 page I/Os

Strategy a)	Strategy b)	Strategy c)	Strategy d)	Strategy e)
110,100	113,000	37,000	33,000	212

Assuming the join result tuple size is twice that of each relation, we can fit 5 tuples per page and the result will occupy 100 / 5 = 20 pages.

Cost: 20 page I/Os (to write $Temp_1$)

Total cost C_1: 212 + 20 = 232 page I/Os

```
select *
from Sailor natural join Reserves natural join Boat
where rDate = '01-JAN-2020'
      and color = 'red';
```

EXERCISE 1 (CONT'D)

bf: 10 tuples/page
B_{Sailor}: 1,000 pages
$B_{Reserves}$: 10,000 pages
B_{Boat}: 100 pages
M: 100 pages
Hash index: Sailor.sailorId
B^+-tree index: Reserves.rDate (2 levels)
Hash index: Boat.boatId
Unique Reserves rDates: 1000
Red boat reservations: 10%
Reservations/sailor: 10

C_2: Cost of computing $Temp_2 = \sigma_{color='red'}$ Boat (no index on color)

Strategy: file scan

For Boat, there is only a hash index on boatId. Therefore, to find red boats, a scan of the entire relation (100 pages) is needed.

Cost: 100 page I/Os

Since only 10% of the boats are red, we expect to retrieve 10%*1000 = 100 Boat tuples, which can fit on 100 / 10 = 10 pages.

Cost: 10 page I/Os (to write $Temp_2$)

Total cost C_2: 100 + 10 = 110 page I/Os

```
select *
from Sailor natural join Reserves natural join Boat
where rDate = '01-JAN-2020'
        and color = 'red';
```

EXERCISE 1 (CONT'D)

bf: 10 tuples/page
B_{Sailor}: 1,000 pages
$B_{Reserves}$: 10,000 pages
B_{Boat}: 100 pages
M: 100 pages
Hash index: Sailor.sailorId
B+-tree index: Reserves.rDate (2 levels)
Hash index: Boat.boatId
Unique Reserves rDates: 1000
Red boat reservations: 10%
Reservations/sailor: 10

C_3: Cost of $Temp_1$ JOIN $Temp_2$ ($Temp_1$ is 20 pages; $Temp_2$ is 10 pages)

Strategy: block nested-loop join

We need to read the 20 pages of $Temp_1$.

Cost: 20 page I/Os

We need to read the 10 pages of $Temp_2$.

Cost: 10 page I/Os

Since there are 100 memory pages, we can simply do the join in memory after reading both relations.

Total cost C_3**:** 30 page I/Os

Total cost: $C_1 + C_2 + C_3 = 232 + 110 + 30 = 372$ page I/Os

EXERCISE 2

Estimate the query processing cost using materialization and the join order

Sailor JOIN ($\sigma_{rDate='01\text{-}JAN\text{-}2020'}$Reserves JOIN $\sigma_{color='red'}$Boat)

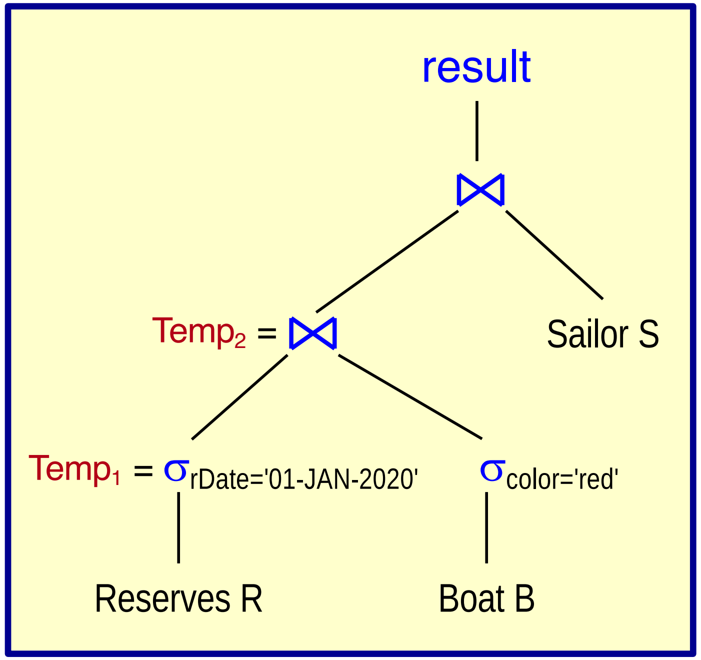
C_1: Cost of computing $Temp_1 = \sigma_{R.rDate='01\text{-}JAN\text{-}2020'}$Reserves

C_2: Cost of computing $Temp_2 = Temp_1$ JOIN $\sigma_{color='red'}$Boat (no index on color)

C_3: Cost of Sailor JOIN $Temp_2$

Total cost: $C_1 + C_2 + C_3$

```
select *
from Sailor natural join Reserves natural join Boat
where rDate = '01-JAN-2020'
        and color = 'red';
```



EXERCISE 2 (CONT'D)

bf: 10 tuples/page
B_{Sailor}: 1000
$B_{Reserves}$: 10000
B_{Boat}: 100
M: 100 pages
Hash index: Sailor.sailorId
B^+-tree index: Reserves.rDate
Hash index: Boat.boatId
Unique Reserves rDates: 1000
Red boat reservations: 10%
Reservations/sailor: 10

C_1: Cost of computing $Temp_1 = \sigma_{R.rDate='01\text{-}JAN\text{-}2020'}$Reserves (no index on color)

Strategy: index lookup using B^+-tree index on Reserves.rDate

As previously determined, to find all reservations on '01-JAN-2020' we retrieve 2 index pages and 10 Reserves pages.

Cost: 10 + 2 = 12 page I/Os

The 100 result Reserves tuples occupy 100 / 10 = 10 pages.

Cost: 10 page I/Os (to write $Temp_1$)

Total cost C_1: 12 + 10 = 22 page I/Os

EXERCISE 2 (CONT'D)

bf: 10 tuples/page
B_{Sailor}: 1000
$B_{Reserves}$: 10000
B_{Boat}: 100
M: 100 pages
Hash index: Sailor.sailorId
B^+-tree index: Reserves.rDate
Hash index: Boat.boatId
Unique Reserves rDates: 1000
Red boat reservations: 10%
Reservations/sailor: 10

C_2: Cost of computing $Temp_2 = Temp_1$ JOIN $\sigma_{color='red'}$Boat

Strategy: block nested-loop join

We need to read the 10 pages of $Temp_1$.

Cost: 10 page I/Os

Do not use the hash index on Boat. Instead, we use block nested loop to join Boat with $Temp_1$ and discard tuples where the color is not red.

(Note that the cost to use the hash index on Boat is 2 pages I/Os —1 to the hash index plus 1 to access the Boat tuple — per reservation for a total cost of 2*100 = 200 page I/Os).

Cost: 100 = 100 page I/Os

Assuming the join result tuple size is twice that of each relation, we can fit 5 tuples per page and the result will occupy 10% * 100 / 5 = 2 pages.

Cost: 2 page I/Os (to write $Temp_2$)

Total cost C_2: 10 + 100 + 2 = 112 page I/Os.

EXERCISE 2 (CONT'D)

bf: 10 tuples/page
B_{Sailor}: 1000
$B_{Reserves}$: 10000
B_{Boat}: 100
M: 100 pages
Hash index: Sailor.sailorId
B^+-tree index: Reserves.rDate
Hash index: Boat.boatId
Unique Reserves rDates: 1000
Red boat reservations: 10%
Reservations/sailor: 10

C_3: Cost of Sailor JOIN $Temp_2$

Strategy: indexed nested-loop join using Sailor.sailorId hash index

We need to read the 2 pages of $Temp_2$ to do the join for C_3.

Cost: 2 page I/Os

For the 10 (10%*100) reservations that satisfy the first two operations, we use the hash index on sailorId to find the information about the sailor.

The cost is 2 page I/Os (1 to the hash index and 1 to access the Sailor tuple) per reservation.

Cost: 2 * 10 = 20 page I/Os

Total cost C_3: 2 + 20 = 22 page I/Os

Total Cost: $C_1 + C_2 + C_3$ = 22 + 112 + 22 = 156 page I/Os