COMP 3311
DATABASE MANAGEMENT SYSTEMS

LECTURE 17 EXERCISES
QUERY OPTIMIZATION

EXAMPLE RELATIONAL DATABASE

Sailor(sailorId, sName, rating, age)   Boat(boatId, bName, color)

Reserves(sailorId, boatId, rDate)

- There are 10,000 Sailor tuples, 100,000 Reserves tuples and 1,000 Boat tuples.
- For all files, *bf* = 10 tuples per page.
  ☞ B_{Sailor} = 1,000 pages; $B_{Reserves}$ = 10,000 pages; B_{Boat} = 100 pages
- For a join result *bf* = 5 tuples per page.
- The buffer *M* = 100 pages.
- 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 minimum page I/O cost to process the query using materialization and the join order

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

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

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

C_3: Cost of $Temp_1$ JOIN $Temp_2$

Query processing page I/O cost: $C_1 + C_2 + C_3$

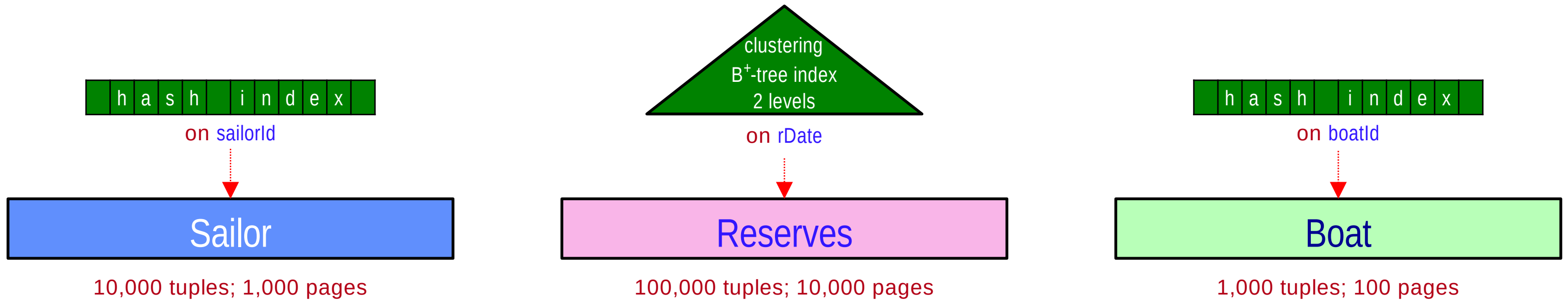
EXERCISE 1 (CONT'D)

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

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

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

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 boats: 10% (100 tuples)
Reservations/sailor: 10




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

<u>Some useful statistics:</u>

- 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 I (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 boats: 10% (100 tuples)
Reservations/sailor: 10

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

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 boats: 10% (100 tuples)
Reservations/sailor: 10

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

a) Strategy 1: block nested-loop – Sailor as outer relation.

Join page I/O cost: $\lceil B_r / (M-2) \rceil * B_s + B_r$

$= \lceil 1000 / (100-2) \rceil * 10000 + 1000 = \underline{111,000}$

b) Strategy 2: block nested-loop – Reserves as outer relation.

Join page I/O cost: $\lceil B_r / (M-2) \rceil * B_s + B_r$

$= \lceil 10000 / (100-2) \rceil * 1000 + 10000 = \underline{113,000}$

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

EXERCISE I (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 boats: 10% (100 tuples)
Reservations/sailor: 10

c) Strategy 3: sort-merge join: $B_r + B_s$ + sorting cost.

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

$= 2 * 1000 * (1 + \lceil \log_{100-1}(1000/100) \rceil) = 4{,}000$

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

$= 2 * 10000 * (1 + \lceil \log_{100-1}(10000/100) \rceil) = 60{,}000$

Merge page I/O cost: $B_r + B_s = 1000 + 10000 = 11{,}000$

Join page I/O cost: 4000 + 60000 + 11000 = 75,000

d) Strategy 4: hash join: $3 * (B_r + B_s)$.

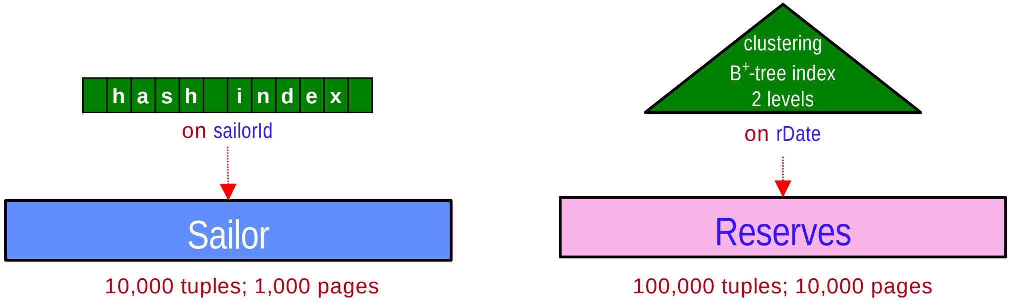
Join page I/O cost: 3 * (1000 + 10000) = 33,000

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

EXERCISE I (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 boats: 10% (100 tuples)
Reservations/sailor: 10

e) Strategy 5: 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 I (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 boats: 10% (100 tuples)
Reservations/sailor: 10

How many pages need to be accessed, to find these 100 reservations?

The index on Reserves.rDate has 2 levels and is ordered on rDate. Since, 10 Reserves tuples fit per page, to retrieve 100 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.

Join page I/O cost: 12 + 200 = 212

Strategy 1	Strategy 2	Strategy 3	Strategy 4	Strategy 5
111,000	113,000	75,000	33,000	212

Since *bf*=5 for a join result, $Temp_1$ will occupy $\lceil 100 / 5 \rceil$ = 20 pages.

Page I/O cost to write $Temp_1$: 20

Total page I/O cost for C_1: $212_r + 20_w$ = 232

```
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 boats: 10% (100 tuples)
Reservations/sailor: 10

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

Strategy: file scan – linear search

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.

File scan page I/O cost: 100

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

Page I/O cost to write $Temp_2$: 10

Total page I/O cost for C_2: $100_r + 10_w$ = 110 page I/Os

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

EXERCISE I (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 boats: 10% (100 tuples)
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

Read the 20 pages of $Temp_1$.

Page I/O cost to read $Temp_1$: 20

Read the 10 pages of $Temp_2$.

Page I/O cost to read $Temp_2$: 10

Since there are 100 buffer pages, the join can be done in memory after reading both relations.

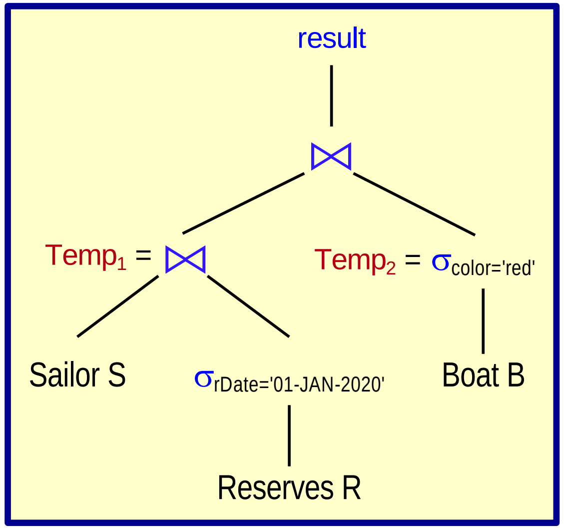
Join page I/O cost C_3: 20 + 10 = 30

Query processing page I/O cost:

$C_1 + C_2 + C_3 = 232 + 110 + 30 = \underline{372}$

Note: As usual, we do not include the cost to write the final output.

Relational Algebra Tree



EXERCISE 2

Estimate the minimum page I/O cost to process the query using materialization and the join order

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

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

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

C_3: Cost of Sailor JOIN $Temp_2$

Query processing page I/O 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 boats: 10% (100 tuples)
Reservations/sailor: 10

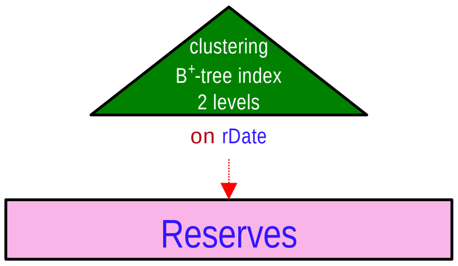
C_1: Cost of computing $Temp_1 = \sigma_{rDate='01\text{-}JAN\text{-}2020'}Reserves$

<u>Strategy 1</u>: file scan – linear search

File scan page I/O cost: 10,000

<u>Strategy 2</u>: 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 100 Reserves tuples which occupy $\lceil 100 / 10 \rceil$ = 10 pages in the Reserves relation (recall that Reserves is ordered on rDate).



Index lookup page I/O cost: 2 + 10 = <u>12</u>

The 100 result Reserves tuples occupy 10 pages.

Page I/O cost to write $Temp_1$: <u>10</u>

Total page I/O cost for C_1: $\underset{r}{12} + \underset{w}{10}$ = <u>22</u>

<u>**Question:**</u> Will all these Reserves tuples be for red boats? No!

How many are expected to be for red boats on average? 10%

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 boats: 10% (100 tuples)
Reservations/sailor: 10

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

Strategy: block nested-loop join

Join $Temp_1$ with Boat and discard tuples where the color is not red. $Temp_1$ fits in memory so join cost is $B_{Temp1} + B_{Boat}$.

Page I/O cost to read $Temp_1$: 10

Page I/O cost to read Boat: 100

Join page I/O cost: 10 + 100 = 110

Of the 100 $Temp_1$ tuples that join with a Boat tuple, 10% * 100 = 10 tuples are expected to join with a Boat tuple whose color is red since 10% of boats are red.

Since bf=5 for a join result, $Temp_2$ will occupy $\lceil 10 / 5 \rceil = 2$ pages.

Page I/O cost to write $Temp_2$: 2

Total page I/O cost for C_2: $\underset{r}{110} + \underset{w}{2} = \underline{112}$

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 boats: 10% (100 tuples)
Reservations/sailor: 10

C_3: Cost of Sailor JOIN $Temp_2$

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

For each of the 10 $Temp_2$ tuples, the hash index on sailorId is used 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.

Page I/O cost to read $Temp_2$: 2

Page I/O cost to use hash index on Sailor:

$2 * 10 = 20$

Total page I/O cost for C_3: $2 + 20 = \underline{22}$

Query processing page I/O cost:

$C_1 + C_2 + C_3 = 22 + 112 + 22 = \underline{156}$

Note: As usual, we do not count the cost to write the final output.

Relational Algebra Tree