# (67506) Databases – Spring 2022 – Exercise (3)

**Muaz Abdeen 300575297**

**Mohammad Ghanayem 208653220**

---

## *Question (1):*

$\boldsymbol{A}$ := authors (<u>name</u>, <u>conference</u>, <u>year</u>, institution, count, adjustedcount)

$\boldsymbol{C}$ := conferences (<u>conference</u>, area, subarea)

$\boldsymbol{I}$ := institutions (<u>institution</u>, region, country)

First, some primary calculations:

For $\boldsymbol{A}$ there are $\left\lfloor \frac{8192}{3\times10+3\times4} \right\rfloor = \left\lfloor \frac{8192}{42} \right\rfloor = 195$ rows per block, and for $\boldsymbol{C}$ there are $\left\lfloor \frac{8192}{30} \right\rfloor = 273$ rows per block.

Therefore, $T(A) = 165000$ , $\boldsymbol{B(A)} = \left\lceil \frac{165000}{195} \right\rceil = \mathbf{847}$ , $T(C) = 20000$ , $\boldsymbol{B(C)} = \left\lceil \frac{20000}{273} \right\rceil = \mathbf{74}$ .

1.a. BNL cost: The outer table is the one with less blocks, that is, conferences ($\boldsymbol{C}$).

$$B(C) + \left\lceil \frac{B(C)}{M-2} \right\rceil \cdot B(A) = 74 + \left\lceil \frac{74}{25-2} \right\rceil \cdot 847 = \mathbf{3462}$$

1.b. Hash: Notice that: $\left\lceil \frac{B(A)}{M-1} \right\rceil = \left\lceil \frac{847}{24} \right\rceil = 36 > 23 = M-2$ , but $\left\lceil \frac{B(C)}{M-1} \right\rceil = \left\lceil \frac{74}{24} \right\rceil = \mathbf{4} \leq \mathbf{23} = M-2$ , then we can join using hash tables. The cost is:

$$3\big(B(A) + B(C)\big) = 3 \cdot (847 + 74) = \mathbf{2763}$$

1.c. Notice that: $\left\lceil \frac{B(A)}{M} \right\rceil = \left\lceil \frac{847}{25} \right\rceil = \mathbf{34} > \mathbf{25} = M$ and $\left\lceil \frac{B(C)}{M} \right\rceil = \left\lceil \frac{74}{25} \right\rceil = 3 < 25 = M$.

Since the sequences generated by divide table $A$ into $M$ partitions, that is $\left\lceil \frac{B(A)}{M} \right\rceil$ , are greater than the buffer size, we can't sort it. So, **we CAN'T execute the Sort Merge join**.

2.a. BNL cost: The outer table is the one with less blocks, that is, conferences ($\boldsymbol{C}$).

$$B(C) + \left\lceil \frac{B(C)}{M-2} \right\rceil \cdot B(A) = 74 + \left\lceil \frac{74}{30-2} \right\rceil \cdot 847 = \mathbf{2615}$$

2.b. Hash: Notice that: $\left\lceil \frac{B(A)}{M-1} \right\rceil = \left\lceil \frac{847}{29} \right\rceil = 30 > 28$ , but $\left\lceil \frac{B(C)}{M-1} \right\rceil = \left\lceil \frac{74}{29} \right\rceil = \mathbf{3} \leq \mathbf{28}$ , then we can join using hash tables. The cost is:

$$3\big(B(A) + B(C)\big) = 3 \cdot (847 + 74) = \mathbf{2763}$$

2.c. Notice that: $\left\lceil \frac{B(A)}{M} \right\rceil = \left\lceil \frac{847}{30} \right\rceil = \mathbf{29} < \mathbf{30} = M$ and $\left\lceil \frac{B(C)}{M} \right\rceil = \left\lceil \frac{74}{30} \right\rceil = \mathbf{3} < \mathbf{30} = M$ . then we can join using Sort Merge. Now, since $\left\lceil \frac{B(A)}{M} \right\rceil + \left\lceil \frac{B(C)}{M} \right\rceil = \mathbf{32} > \mathbf{30} = M$ we can't merge the sorted sequences before merging each table sequences to one sorted table, so the cost is:

$$5\big(B(A) + B(C)\big) = 5 \cdot (847 + 74) = \mathbf{4605}$$

3.a. BNL: minimal buffer size: **3** . Since the buffer must have at least one page for table $A$, and one page for table $C$, and a page for the output.

3.b. Hash-Join: minimal buffer size: **11** . Since we must have at least one table such that its biggest bin size is less that the buffer size.

$$\left\lceil \frac{\min\big(B(A), B(C)\big)}{M-1} \right\rceil \leq M - 2 \iff \left\lceil \frac{74}{M-1} \right\rceil \leq M - 2$$

$$\iff \lceil M^2 - 3M - 72 \rceil \geq 0 \iff M \geq 11$$

3.c. Sort-Merge: minimal buffer size: **30** . Since we must be able to sort each table, that is, each table number of sorted sequences must be less than the buffer size.

$$\left\lceil \frac{B(A)}{M} \right\rceil < M \iff \lceil M^2 \rceil > 847 \iff M \geq 30$$

$$\left\lceil \frac{B(C)}{M} \right\rceil < M \iff \lceil M^2 \rceil > 74 \iff M \geq 9$$

So, for $M = 30$ both tables can be sorted, and we can execute Sort-Merge-Join.

3.d. To execute Sort-Merge-Join in more efficient way, in addition to the condition in the previous section, it must hold:

$$\left\lceil \frac{B(A)}{M} \right\rceil + \left\lceil \frac{B(C)}{M} \right\rceil < M$$

$$\left\lceil\frac{847}{M}\right\rceil + \left\lceil\frac{74}{M}\right\rceil < M \quad \Leftrightarrow \quad \lceil M^2 \rceil > 921 \quad \Leftrightarrow \quad M > 31$$

So, the minimal buffer size is: **32** .

## *Question (2):*

1. Output size of $\sigma_{B=6}S(B,C) =$ **1** block.

$$|\sigma_{B=6}S(B,C)| = \frac{T(S)}{V(S,B)} = \frac{B(S) \times 150}{V(S,B)} = \frac{200 \times 150}{250} = \mathbf{120\ \ rows}$$

Since each block in ***S*** contains 150 rows, then there is $\left\lceil\frac{120}{150}\right\rceil = \mathbf{1\ block}$.

2. Output size of $\sigma_{A<25}R(A,C) =$ **500** blocks.

$$|\sigma_{A<25}R(A,C)| = \frac{T(R)}{V(R,A)} = \frac{B(R) \times 150}{3} = \frac{1500 \times 60}{3} = \mathbf{30,000\ \ rows}$$

Since each block in ***A*** contains 60 rows, then there are $\left\lceil\frac{30000}{60}\right\rceil = \mathbf{500\ blocks}$.

3. Output size of $\sigma_{A<25 \wedge B=6}\big(R(A,C) \bowtie S(B,C)\big) =$ **120** rows.

$$\begin{aligned}
\left|\sigma_{A<25 \wedge B=6}\big(R(A,C) \bowtie S(B,C)\big)\right| &= \frac{T(R) \times T(S)}{3 \times V(S,B) \times \max\{V(R,C), V(S,C)\}} \\
&\overset{*}{=} \frac{T(R) \times T(S)}{3 \times V(S,B) \times \max\{V(R,C), T(S)\}} \\
&= \frac{90{,}000 \times 30{,}000}{3 \times 250 \times \max\{50\,, 30000\}} \\
&= \frac{90{,}000 \times 30{,}000}{3 \times 250 \times 30{,}000} = \mathbf{120\ \ rows}
\end{aligned}$$

* S.C is a key so $V(S,C) = T(S)$ .

4. The most efficient algorithm is:

From previous sections we conclude that pushing down the selection decreases the size of tables to be joined, so whenever we can push, we do that.

First, notice that for the expression $\sigma_{A<25}R(A,C)$ we have:

$READ(E_R) = B(R) = 1500$ , we can read it by full table scan only since there is no index.

$B(E_R) = 500$ , and $T(E_S) = 30{,}000$ as calculated in section (2).

Second, for the expression $\sigma_{B=6}S(B,C)$ we have:

$READ(E_S) = B(S) = 200$ , by full scan, and $READ(E_S) = T(E_S) = 120$ by index scan, since it is given that the cost of accessing the index is negligible.

$B(E_S) = 1$ , and $T(E_S) = 120$ as calculated in section (1).

Now, for the **BNL** query plan the cost is:

$$READ(E_S) + \left\lceil \frac{B(E_S)}{M-2} \right\rceil \cdot READ(E_R) = 120 + \left\lceil \frac{1}{8} \right\rceil \cdot 1500 = \mathbf{1620}$$

We used $S(B,C)$ as outer relation since the expression $\sigma_{B=6}S(B,C)$ has smaller size $B(E_S) = 1$ .

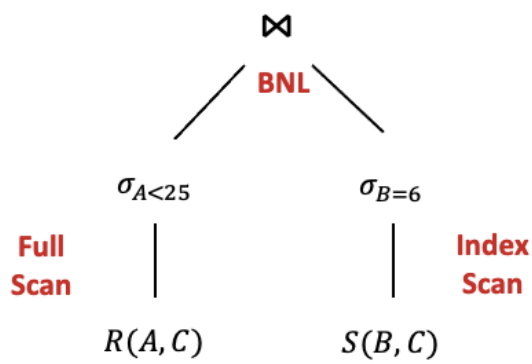
We can notice that the queries **SM** and **Hash** can't be more efficient since they require at least 3 passes over relations, whereas BNL required a single pass in our case.

For the **INL** query plan the cost is:

$$READ(E_R) + T(E_R) \times \text{ cost\_of\_select } = 1500 + 30{,}000 \times 1 = \mathbf{31{,}500}$$

We used $R(A,C)$ as outer relation and $S(B,C)$ as inner relation since there is an index on the join attribute $C$ in the relation $S$ only ($= S.C$). For the cost of selection, notice that for each tuple in $E_R$ there is one matching tuple in $S$, since $S.C$ is a key, and because the cost of accessing the index is negligible, for every tuple we just calculate the cost of getting the corresponding block from the disk (each tuple may be in a different block), so cost_of_select $= 1$ .

Finally, we conclude that **BNL** is the optimal query plan with cost **1620** .




5. The cost of the most efficient algorithm is:

**BNL** is the optimal query plan with cost **1620** .

## Question (3):

1. In $R$ there are $\left\lceil \frac{1500}{30} \right\rceil = 50$ rows, in $S$ there are $\left\lceil \frac{1500}{20} \right\rceil = 75$ rows, so we got that $T(R) = 1000 \times 50 = 50{,}000$ rows, and $T(S) = 3000 \times 75 = 225000$ rows.

   Now to find out how much rows results, we compute the following ($C$ is a key for $R$):

$$\frac{T(R) * T(S)}{\max\{V(R,C), V(S,C) * V(S,D) * V(R,B)\}} = \frac{T(R) * T(S)}{T(R) * V(R,B) * V(S,D)} = \frac{225000}{5 * 3} = \mathbf{15000}\ \textbf{rows}$$

2. In our results we get that each row is of size 20 bytes, so in each block there are $\left\lceil \frac{1500}{20} \right\rceil = 75$ rows, so we got that the size of our result in blocks is $\left\lceil \frac{15000}{75} \right\rceil = \mathbf{200}$ **blocks**.

3. To begin with, we need to compute the costs and sizes for each $E_s = \pi_{C,D}\sigma_{D<4}(S(C,D))\ and\ E_R = \pi_{A,C}\sigma_{B=15}(R(A,B,C))$.

   Now, we want to compute $B(E_R)$:

   There is $\frac{50000}{5} = 10000$ matching rows, after running projection each row needs 20 bytes, so we got $\left\lceil \frac{10000}{\frac{1500}{20}} \right\rceil = 1000$ blocks.

   So, we have:

   $Read(E_R) = B(R) = 1000, B(E_R) = 134, T(E_R) = 10000$

   $Read(E_S) = B(S) = 3000, B(E_S) = 1000, T(E_S) = 75000$

   Let's compute the cost of each possibility:

   Two possibilities for BNL Join:

   - $Read(E_S) + Read(E_R) * \left\lceil \frac{B(E_S)}{M-2} \right\rceil = 3000 + \left\lceil \frac{1000}{18} \right\rceil * 1000 = \mathbf{59000}$
   - $Read(E_R) + Read(E_S) * \left\lceil \frac{B(E_R)}{M-2} \right\rceil = 1000 + 3000 * \left\lceil \frac{134}{18} \right\rceil = \mathbf{25000}$

   For the next possibility, Sort Join, we should check if the condition holds, so:

$$\left\lceil \frac{B(E_R)}{M} \right\rceil < M \wedge \left\lceil \frac{B(E_S)}{M} \right\rceil < M \Rightarrow \left\lceil \frac{134}{20} \right\rceil = 7 < 30 \wedge \left\lceil \frac{1000}{20} \right\rceil = 50 \nless 20$$

   So, SM not working.

   Now we going to check the condition for Hash Join, if it is holds or not:

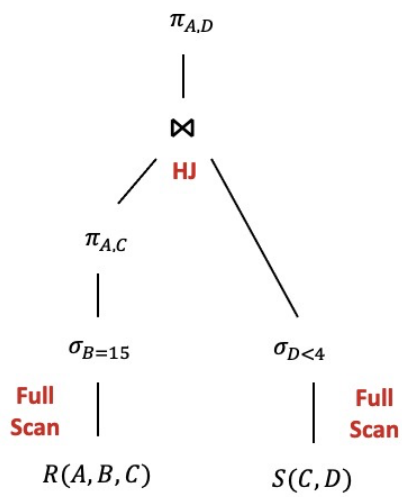
$$\left\lceil \frac{\min\{B(E_R), B(E_S)\}}{M-1} \right\rceil < M - 1 \Rightarrow \left\lceil \frac{134}{19} \right\rceil = 8 < 29$$

And the cost is:

$$Read(E_R) + Read(E_S) + 2\big(B(E_R) + B(E_S)\big) = 1000 + 3000 + 2268 = \mathbf{6268}$$

And from all these checks we got that the most effective way to compute the results is **Hash Join.**

This is the query plan tree:




4. $Read(E_R) + Read(E_S) + 2\big(B(E_R) + B(E_S)\big) = 1000 + 3000 + 2268 =$ ***6268 I/O operations***

## *Question (4):*

1. The query is canceled due to timeout error.

```
ERROR:  canceling statement due to statement timeout
```

The query plan according to `EXPLAIN` command:

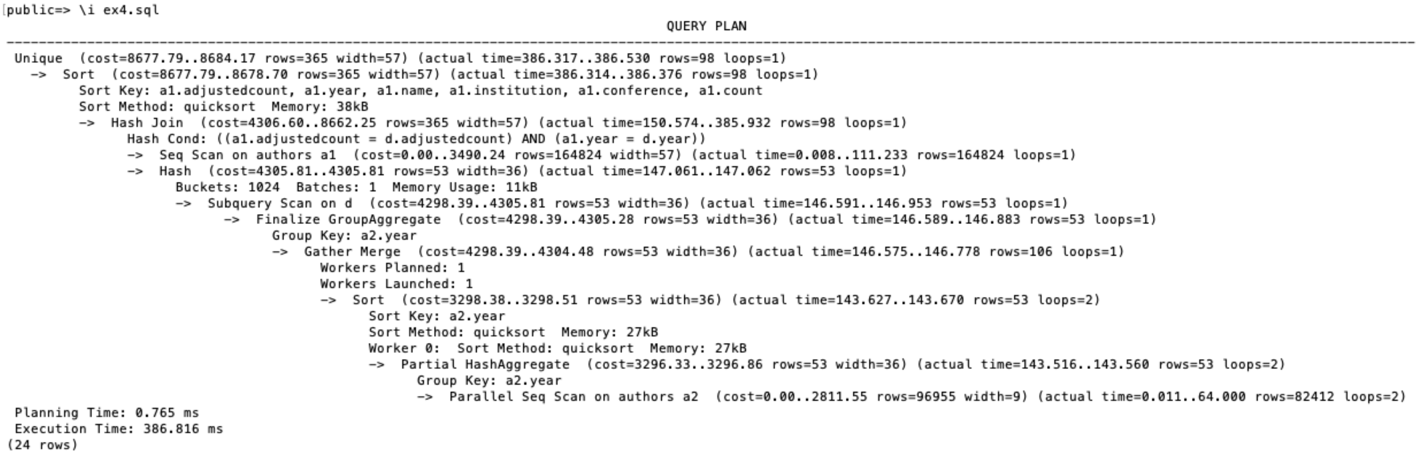
```
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Unique  (cost=644479792.25..644479806.67 rows=806 width=57)
   ->  Sort  (cost=644479792.25..644479794.31 rows=824 width=57)
         Sort Key: a1.name, a1.institution, a1.conference, a1.count, a1.adjustedcount, a1.year
         ->  Seq Scan on authors a1  (cost=0.00..644479752.34 rows=824 width=57)
               Filter: (adjustedcount = (SubPlan 1))
               SubPlan 1
                 ->  Aggregate  (cost=3910.08..3910.09 rows=1 width=32)
                       ->  Seq Scan on authors a2  (cost=0.00..3902.30 rows=3110 width=5)
                             Filter: (year = a1.year)
 JIT:
   Functions: 10
   Options: Inlining true, Optimization true, Expressions true, Deforming true
(12 rows)
```

2. I change the query like this:

```
SELECT DISTINCT *
FROM authors a1 NATURAL JOIN (SELECT year, max(adjustedcount) AS adjustedcount
                               FROM authors a2
                               GROUP BY year) D;
```

I think the cause is we don't rescan the table sequentially for every tuple in the table, this what the condition `WHERE` does, instead my new query scans the table once to create the table `D`, then joins both `authors` and `D`.

The query plan is:



```
[public=> \i ex4.sql
                                                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=8677.79..8684.17 rows=365 width=57) (actual time=386.317..386.530 rows=98 loops=1)
   ->  Sort  (cost=8677.79..8678.70 rows=365 width=57) (actual time=386.314..386.376 rows=98 loops=1)
         Sort Key: a1.adjustedcount, a1.year, a1.name, a1.institution, a1.conference, a1.count
         Sort Method: quicksort  Memory: 38kB
         ->  Hash Join  (cost=4306.60..8662.25 rows=365 width=57) (actual time=150.574..385.932 rows=98 loops=1)
               Hash Cond: ((a1.adjustedcount = d.adjustedcount) AND (a1.year = d.year))
               ->  Seq Scan on authors a1  (cost=0.00..3490.24 rows=164824 width=57) (actual time=0.008..111.233 rows=164824 loops=1)
               ->  Hash  (cost=4305.81..4305.81 rows=53 width=36) (actual time=147.061..147.062 rows=53 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 11kB
                     ->  Subquery Scan on d  (cost=4298.39..4305.81 rows=53 width=36) (actual time=146.591..146.953 rows=53 loops=1)
                           ->  Finalize GroupAggregate  (cost=4298.39..4305.28 rows=53 width=36) (actual time=146.589..146.883 rows=53 loops=1)
                                 Group Key: a2.year
                                 ->  Gather Merge  (cost=4298.39..4304.48 rows=53 width=36) (actual time=146.575..146.778 rows=106 loops=1)
                                       Workers Planned: 1
                                       Workers Launched: 1
                                       ->  Sort  (cost=3298.38..3298.51 rows=53 width=36) (actual time=143.627..143.670 rows=53 loops=2)
                                             Sort Key: a2.year
                                             Sort Method: quicksort  Memory: 27kB
                                             Worker 0:  Sort Method: quicksort  Memory: 27kB
                                             ->  Partial HashAggregate  (cost=3296.33..3296.86 rows=53 width=36) (actual time=143.516..143.560 rows=53 loops=2)
                                                   Group Key: a2.year
                                                   ->  Parallel Seq Scan on authors a2  (cost=0.00..2811.55 rows=96955 width=9) (actual time=0.011..64.000 rows=82412 loops=2)
 Planning Time: 0.765 ms
 Execution Time: 386.816 ms
(24 rows)
```

3. Create index on authors(year, adjustedcount);

```
Unique  (cost=433655.95..433670.37 rows=806 width=57) (actual time=2110.832..2111.077 rows=98 loops=1)
  ->  Sort  (cost=433655.95..433658.01 rows=824 width=57) (actual time=2110.830..2110.900 rows=98 loops=1)
        Sort Key: a1.name, a1.institution, a1.conference, a1.count, a1.adjustedcount, a1.year
        Sort Method: quicksort  Memory: 38kB
        ->  Seq Scan on authors a1  (cost=0.00..433616.04 rows=824 width=57) (actual time=36.861..2110.511 rows=98 loops=1)
              Filter: (adjustedcount = (SubPlan 2))
              Rows Removed by Filter: 164726
              SubPlan 2
                ->  Result  (cost=2.60..2.61 rows=1 width=32) (actual time=0.010..0.011 rows=1 loops=164824)
                      InitPlan 1 (returns $1)
                        ->  Limit  (cost=0.42..2.60 rows=1 width=5) (actual time=0.007..0.008 rows=1 loops=164824)
                              ->  Index Only Scan Backward using authors_year_adjustedcount_idx on authors a2  (cost=0.42..6901.85 rows=3170 width=5)
(actual time=0.006..0.006 rows=1 loops=164824)
                                    Index Cond: ((year = a1.year) AND (adjustedcount IS NOT NULL))
                                    Heap Fetches: 164824
Planning Time: 0.197 ms
JIT:
  Functions: 10
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 1.180 ms, Inlining 0.000 ms, Optimization 0.336 ms, Emission 6.048 ms, Total 7.564 ms
Execution Time: 2112.418 ms
(20 rows)
```

It took 2112.418 ms = 2.112418 seconds to run the query from section 1, it ran faster because of the fact that for each row in the table it uses the index to scan the wanted year and adjustedcount, and that's absolutely faster than full table scan, and that's it.