# Advanced Databases Course project

**Team 9**

Ayman Mohamed Reda
Ahmed Ayman
Ammar Mohamed
Omar Ahmed Rezk

## Proposes:

1) **Schema Optimizations:**

we will merge 2 tables together instead of having them discrete.
we will modify column data Types, for example: from Float to Int, from BigInt to Int, from nvarchar to varchar or to char, from char to tinyInt or Bool (in gender column).
As we are trying to minimize Table sizes as much as possible.

2) **Memory and Cache optimization:**

We will increase Buffer Size from the DBMS.
We tried to increase Block size but we couldn't do it, DBMS didn't let us to.
Tried to store ready procedures, DBMS is far better in this task, it optimizes the query very heavily if you run the query multiple times in a row.

3) **Index Tuning:**

We added 2 indices, one for Salary on table Employee only and the other (Salary, SSN) on table Employee. Besides the default SSN index on table Employee.

4) **Query Optimization:**

We tried to get an optimized version for all our queries, we succeeded in 2 of them and saw great improvements.

## Validation and Testing:

All tests were done on the same computer, same environment with the charger plugged in and zero power savings.

**PC specs:**

| | |
|---|---|
| Device name | LAPTOP-66BOPJMV |
| Processor | Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz |
| Installed RAM | 16.0 GB (15.8 GB usable) |
| System type | 64-bit operating system, x64-based processor |

OS Details:

| | |
|---|---|
| Edition | Windows 10 Home |
| Version | 21H2 |
| Installed on | 4/7/2021 |
| OS build | 19044.1645 |
| Experience | Windows Feature Experience Pack 120.2212.4170.0 |

GPU: NVIDIA GEFORCE GTX 1650

DISKS: ST1000LM048-2E7172 (1 TB)
SSD: WDC PC SN530 SDBPNPZ-256G-1014 (256GB)

# Experimentation steps:

We implemented and filled 9 variants of databases, with their description below:

| Type | Size | Optimization |
|---|---|---|
| SQL | 10K | False |
| SQL | 100K | False |
| SQL | 1M | False |
| SQL | 10M | False |
| SQL | 100K | Index Tuning |
| SQL | 100K | Increase Buffer Size + Index Tuning |
| SQL | 100K | Optimized Schema only |
| SQL | 100K | Full optimization (schema + index + Buffer size) |
| noSQL | 100K | False |

- For the Raw Databases, we test with normal queries.
- Then we test with optimized queries
- Then we test index-tuned DB with normal queries vs no index-tuning
- Then we test increased Buffer Sized DB vs normal conditions
- Then we test Full optimization DB vs zero optimization
- Then we test Full optimization DB vs noSQL DB

### Results and Queries Follows:

# Proposes:

1) **Schema Optimizations:**
we will merge 2 tables together instead of having them discrete.
we will modify column data Types, for example: from Float to Int, from BigInt to Int, from nvarchar to varchar or to char, from char to tinyInt or Bool (in gender column).
As we are trying to minimize Table sizes as much as possible.

2) **Memory and Cache optimization:**
We will increase Buffer Size from the DBMS.
We tried to increase Block size but we couldn't do it, DBMS didn't let us to.
Tried to store ready procedures, DBMS is far better in this task, it optimizes the query very heavily if you run the query multiple times in a row.

3) **Index Tuning:**
We added 2 indices, one for Salary on table Employee only and the other (Salary, SSN) on table Employee. Besides the default SSN index on table Employee.

4) **Query Optimization:**
We tried to get an optimized version for all our queries, we succeeded in 2 of them and saw great improvements.

# Validation and Testing:

All tests were done on the same computer, same environment with the charger plugged in and zero power savings.

**PC specs:**

| | |
|---|---|
| Device name | LAPTOP-66BOPJMV |
| Processor | Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz   2.50 GHz |
| Installed RAM | 16.0 GB (15.8 GB usable) |
| System type | 64-bit operating system, x64-based processor |

OS Details:

| | |
|---|---|
| Edition | Windows 10 Home |
| Version | 21H2 |
| Installed on | 4/7/2021 |
| OS build | 19044.1645 |
| Experience | Windows Feature Experience Pack 120.2212.4170.0 |

GPU: NVIDIA GEFORCE GTX 1650

DISKS: ST1000LM048-2E7172     (1 TB)
SSD: WDC PC SN530 SDBPNPZ-256G-1014    (256GB)

# Experimentation steps:

We implemented and filled 9 variants of databases, with their description below:

| Type | Size | Optimization |
|---|---|---|
| SQL | 10K | False |
| SQL | 100K | False |
| SQL | 1M | False |
| SQL | 10M | False |
| SQL | 100K | Index Tuning |
| SQL | 100K | Increase Buffer Size + Index Tuning |
| SQL | 100K | Optimized Schema only |
| SQL | 100K | Full optimization (schema + index + Buffer size) |
| noSQL | 100K | False |

- For the Raw Databases, we test with normal queries.
- Then we test with optimized queries
- Then we test index-tuned DB with normal queries vs no index-tuning
- Then we test increased Buffer Sized DB vs normal conditions

- Then we test Full optimization DB vs zero optimization
- Then we test Full optimization DB vs noSQL DB

**Results and Queries Follows:**

```
another_complex3 = 'select count(SSN) from Employee a, Department b where
a.SSN < b.Dnumber and a.Salary < b.Dnumber and b.Mgr_SSN < 80000'
```

```
another_complex_ayman = 'select DISTINCT Salary, Fname, Dname from
Employee e, Department d, Contracts c where e.SSN = c.Emp_SSN and e.DNO =
d.Dnumber'
```

```
another_complex4 = 'select SSN , Salary from Employee a, Department b
where a.SSN < b.Dnumber and a.Salary < b.Dnumber and b.Mgr_SSN < 80000
group by Salary, SSN;'
```

Times in minutes

| | Query1 | Query2 | Query3 | Query4 |
|---|---|---|---|---|
| 10k | 0.0621 | 0.112985 | 0.03771735 | 2.354 |
| 100k | 0.7951 | 1.382995 | 4.403953 | 224.1677 |
| 1M | 1.078 | 0.930993 | 81.8903 | |
| 10M | 1.181997 | 7.10634 | | |

AFTER QUERY OPTIMIZATIONS:

```
another_complex3_OPT = 'select count(SSN) from Employee a inner join Department b on a.SSN < b.Dnumber inner join Department d on a.Salary < d.Dnumber where d.Mgr_SSN < 80000'
```

```
another_complex4_OPT = 'select SSN, Salary from (Select SSN, Salary from Employee)as a inner join (select Dnumber, Mgr_SSN from Department where Mgr_SSN < 80000)as b on a.SSN < b.Dnumber group by Salary, SSN'
```

Times in minutes

| | Q3 opt | Q4 opt | | |
|---|---|---|---|---|
| 10k | 0.064999 | 0.005024 | | |
| 100k | 3.4 | 0.00506 | | |
| 1M | 78.58 | 0.00499 | | |
| 10M | | | | |

Why on Q4 we can see that the query takes a constant time?
Because we selected Mgr_SSN < 80000 as early as possible, shrinking our search space to multiples of 80000 only, not multiples of millions of records.

After putting index on column Salary we will clearly see Optimizations when executing Q4

Time in seconds

| | Query 4 without optimizations After Salary Index |
|---|---|
| 10k | 2.59904 |
| 100k | 1.94004 |
| 1M | 78.32 |
| 10M | 102.4084717 |

Now we will increase the buffer size

From 65536 B to 655360 B

| | Query 4 without optimizations |
|---|---|
| 10k | 2.26903 |
| 100k | 1.86799 |
| 1M | 120.9788 |
| 10M | 108.405499 |

We will try to push it up to 65536000 B

| | Query 4 without optimizations |
|---|---|
| 10k | 2.343 |
| 100k | 1.9679 |
| 1M | 107.55343 |

Now reverting all our optimizations again (deleting the additional indices, decreasing buffer size, no optimizations) and getting worse results again, to prove that our optimizations were effective.

Results in seconds

| | Query 4 |
|---|---|
| 10k | 3.313 |
| 100k | 315.242124 |
| 1M | 2490.514 |

Try to optimize the schema by merging Works_On with Employee, by adding columns “PNO” & “Hours” to Employee instead of having an entire different table.

And came up with a new query:

```
Same_query = 'select sum(Hours) from  Employee as e, Project as p,
Department as d where e.PNO = p.Pnumber and p.DNO = d.Dnumber and
d.Dnumber > 50 and d.Mgr_SSN = e.SSN and e.Hours > 10 group by e.Salary,
d.Dnumber'
```

Results in seconds:

| | Before merging | After Merging |
|---|---|---|
| 100k DB | 6.09847 | 0.124001 |

Space improvement:
Employee size + Works_On size before merging = 40.352 kB

Employee size after merging = 39.291

Space improvement = 102.62%

# OPTIMIZATIONS DONE:

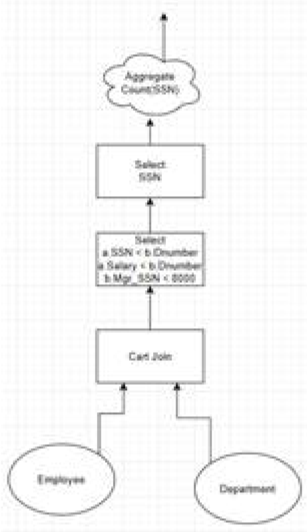
1) final DB statistics
2) Schema modifications: merged Works_On with Employee, created new Indices (Salary) & (Salary, SSN), decreased size of column data types.
3) Memory optimization: increased Buffer size from 65536 B to 65536000 B.
4) Index-Tuning: created Indices (Salary) & (Salary,SSN)
5) for each query: modifications

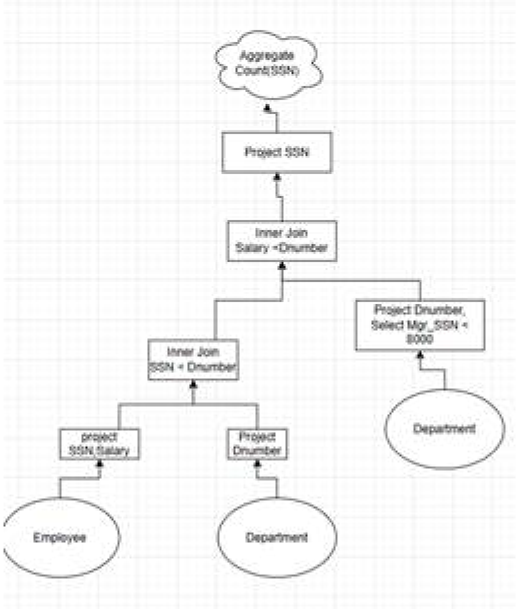
Original Query 3:

```
another_complex3 = 'select count(SSN) from Employee a, Department b where a.SSN <
b.Dnumber and a.Salary < b.Dnumber and b.Mgr_SSN < 80000'
```

Optimized Query 3:

```
another_complex3_OPT = 'select SSN from (select SSN from Employee) as a inner
join (select Dnumber from Department)as b on a.SSN < b.Dnumber inner join (select
Dnumber, Mgr_SSN from Department) as d on a.Salary < d.Dnumber where d.Mgr_SSN <
80000'
```



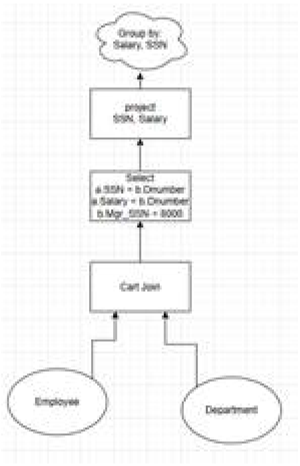


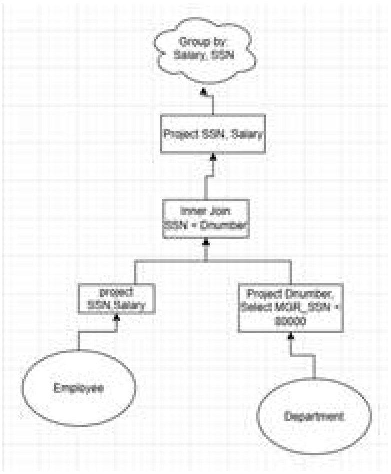
Original Query 4:

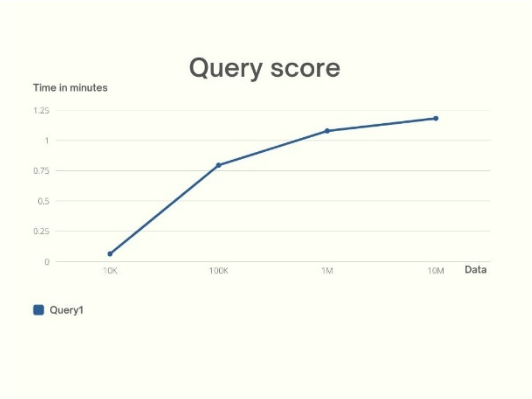
```
another_complex4 = 'select SSN , Salary from Employee a, Department b where a.SSN
< b.Dnumber and a.Salary < b.Dnumber and b.Mgr_SSN < 80000 group by Salary, SSN;'
```

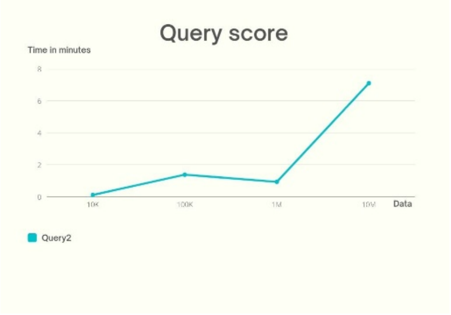
Optimized Query 4:

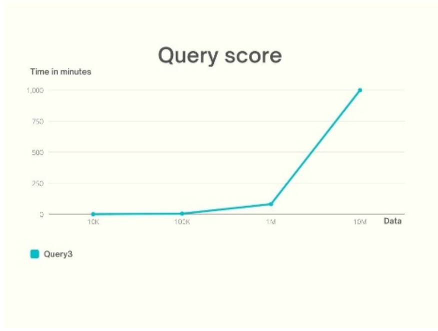
```
another_complex4_OPT = 'select SSN, Salary from (Select SSN, Salary from
Employee)as a inner join (select Dnumber, Mgr_SSN from Department where Mgr_SSN <
80000)as b on a.SSN < b.Dnumber group by Salary, SSN'
```

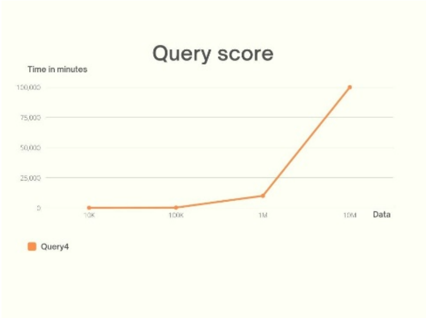

Group by:
Salary, SSN
project
SSN, Salary
Select
a.SSN = b.Dnumber
a.Salary = b.Dnumber
b.Mgr_SSN = 8000
Cart Join
Employee
Department

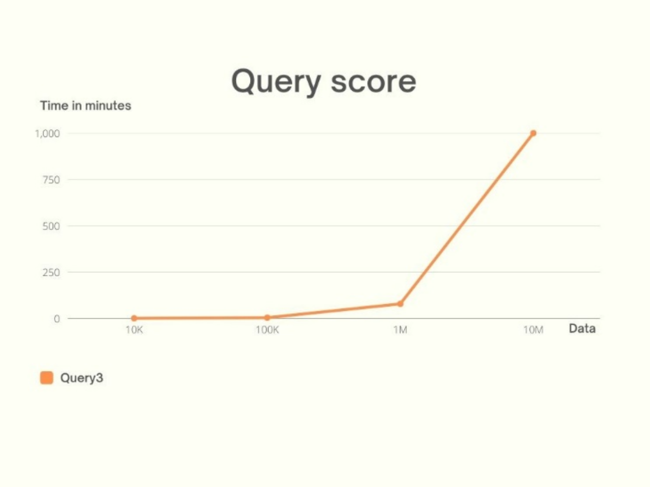

Group by:
Salary, SSN
Project SSN, Salary
Inner Join
SSN = Dnumber
project
SSN,Salary
Project Dnumber,
Select MGR_SSN =
80000
Employee
Department

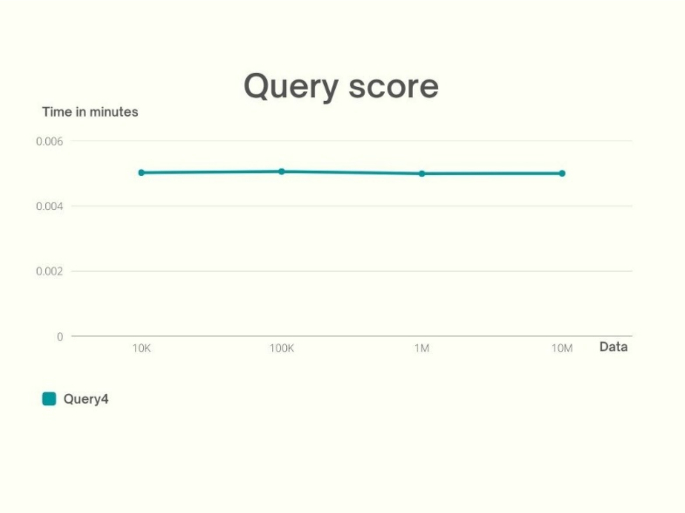

Query score
Time in minutes
1.25
1
0.75
0.5
0.25
0
10K
100K
1M
10M
Data
Query1

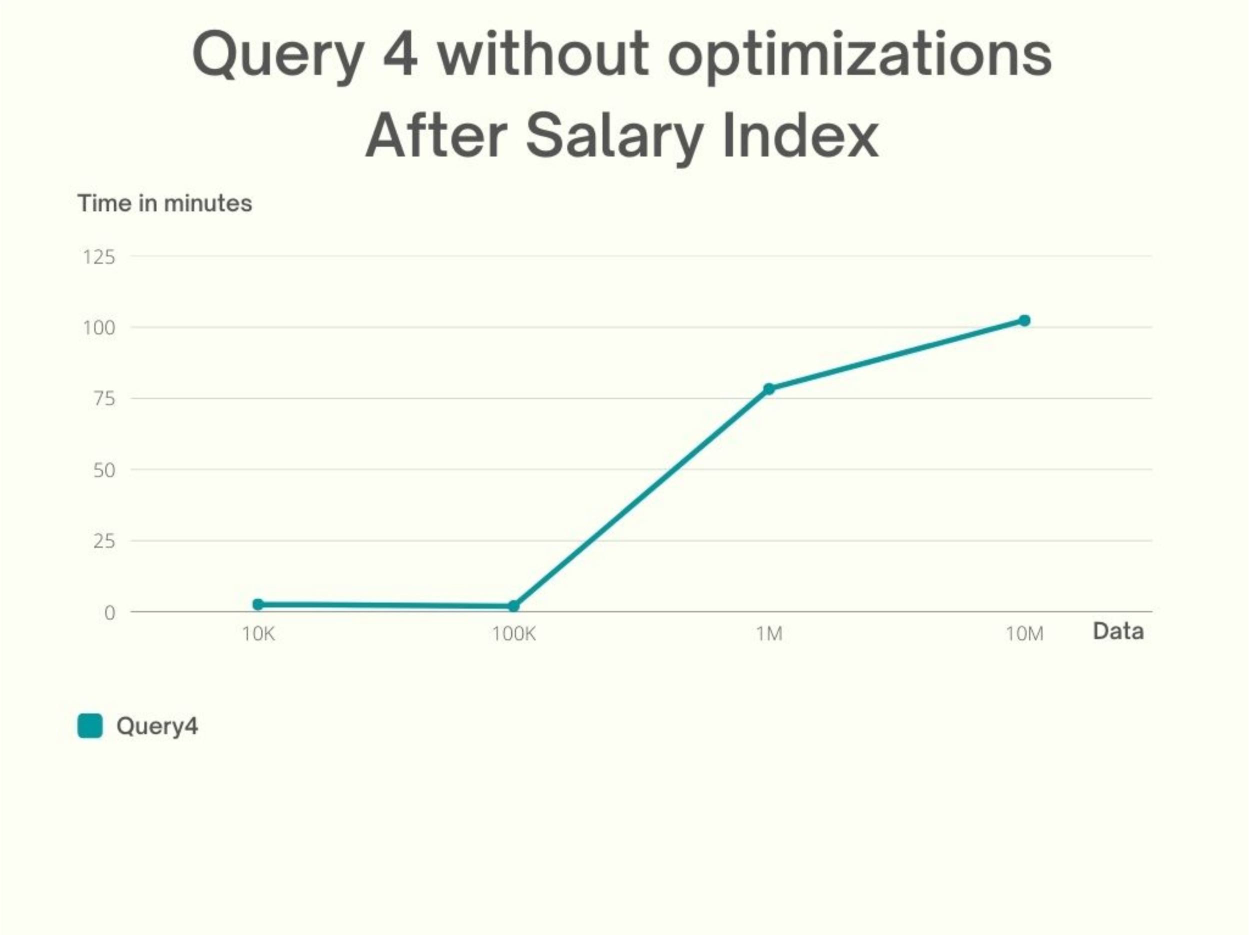

Query score
Time in minutes
8
6
4
2
0
10K
100K
1M
10M
Data
Query2


Query score
Time in minutes
1,000
750
500
250
0
10K
100K
1M
10M
Data
Query3

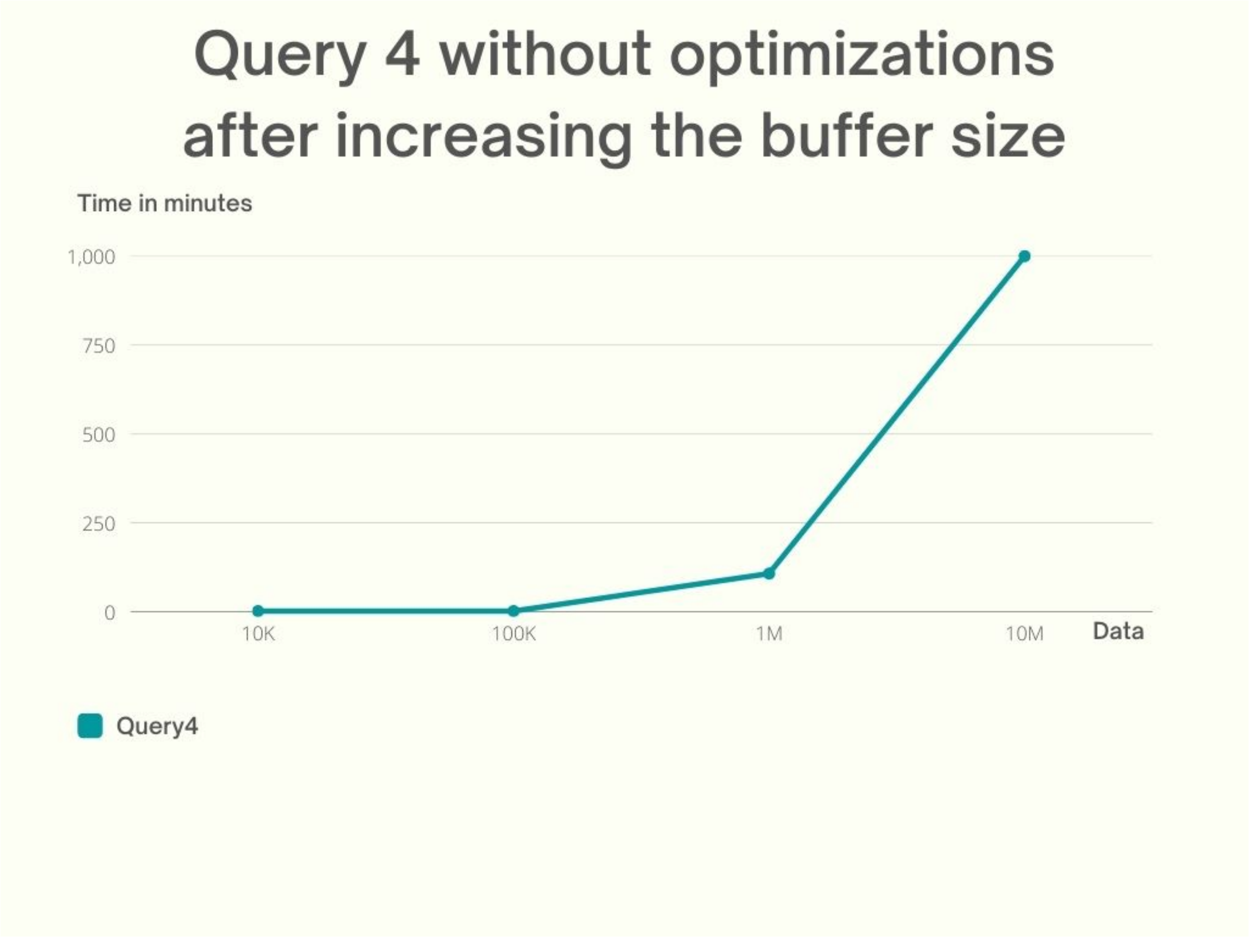

Query score
Time in minutes
100,000
75,000
50,000
25,000
0
10K
100K
1M
10M
Data
Query4


Query score
Time in minutes
1,000
750
500
250
0
10K
100K
1M
10M
Data
Query3

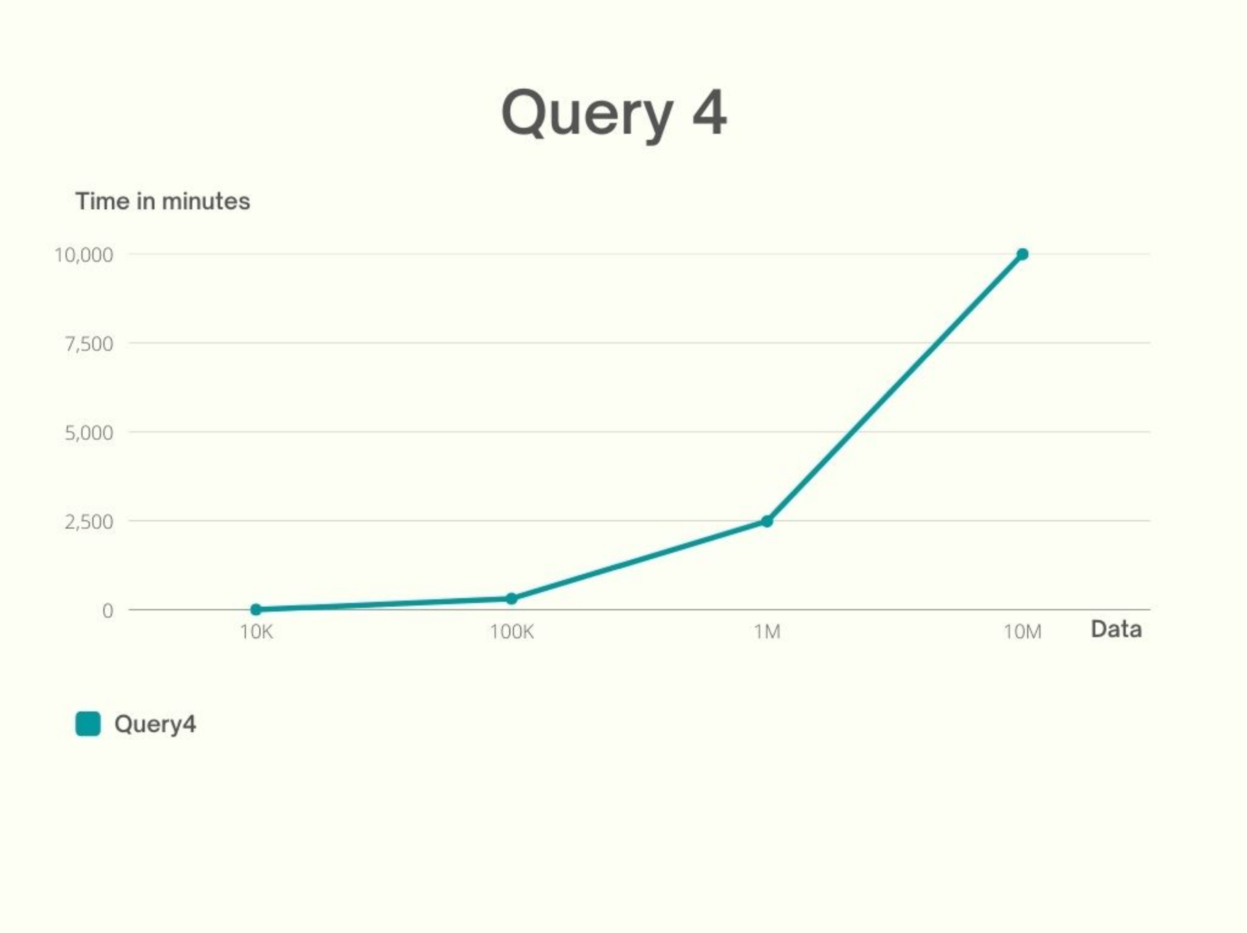

Query score
Time in minutes
0.006
0.004
0.002
0
10K
100K
1M
10M
Data
Query4



Query 4 without optimizations
After Salary Index



Query 4 without optimizations after increasing the buffer size


Query 4
Time in minutes
10,000
7,500
5,000
2,500
0
10K
100K
1M
10M
Data
Query4