Advanced Database

Phase 2

JANUARY 16

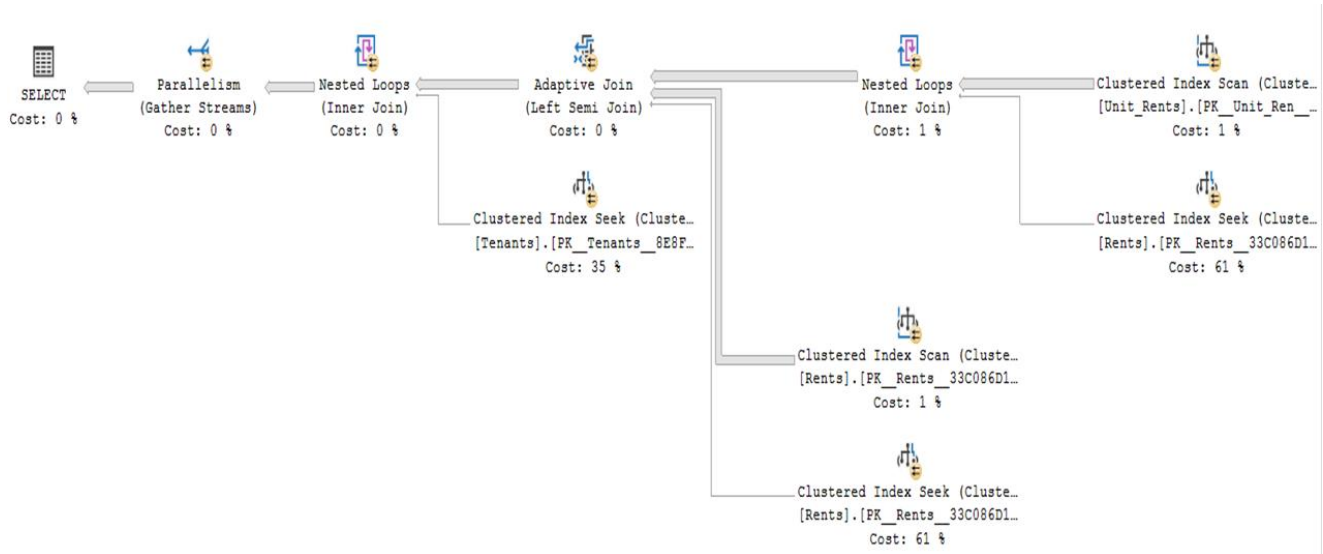
Ahmad Nader Adel – 1162296
Kamel Mohsen Kamel – 1162325
Ahmed Mohamed Amrawy – 1162173
Fatema Othman Mahmoud – 1162040

Contents

Query 1

Before Optimization:

```
SELECT T.Tenant_Id,T.Tenant_FirstName,T.Tenant_LastName
FROM Unit_Rents UR
INNER LOOP JOIN Rents R ON
UR.Unit_Rents_RentId = R.Rent_Id
INNER LOOP JOIN Tenants T ON
UR.Unit_Rents_TenantId = T.Tenant_Id
WHERE R.Rent_DueDate >= '1900-01-01'
AND R.Rent_DueDate <= '2020-12-31'
            AND R.Rent_Id in
            (
             SELECT R.Rent_Id
             FROM  Rents R
             WHERE R.Rent_Statues = 'False'
            )
```

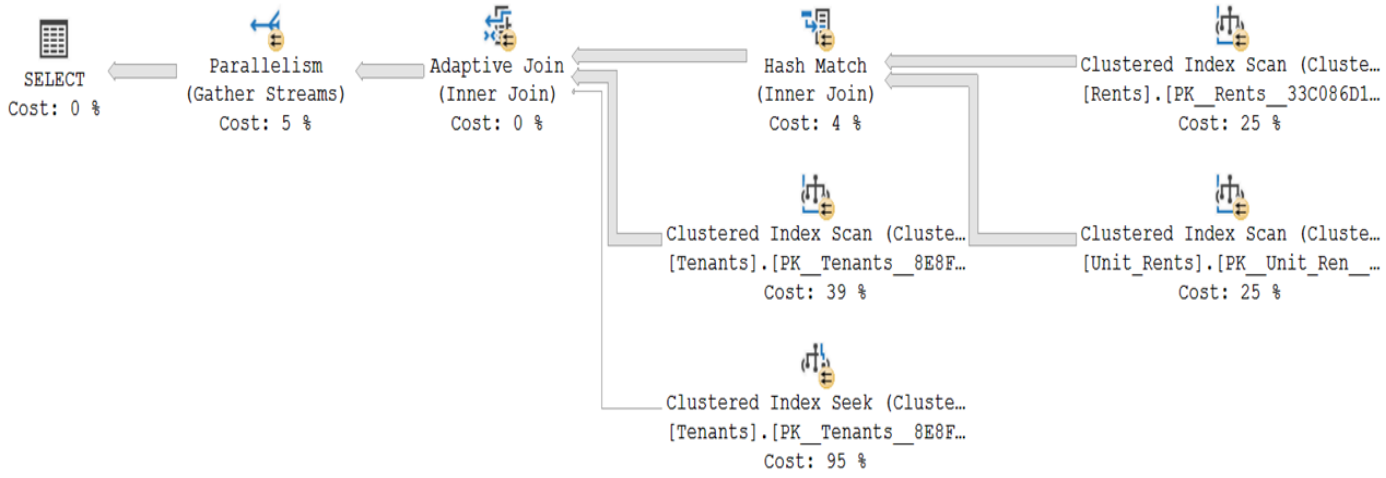


Rows Affected = 980,165 rows

Query 1

After Optimization

```
SELECT T.Tenant_Id,T.Tenant_FirstName,T.Tenant_LastName
FROM Unit_Rents UR
INNER JOIN Rents R ON
UR.Unit_Rents_RentId = R.Rent_Id
INNER JOIN Tenants T ON
UR.Unit_Rents_TenantId = T.Tenant_Id
WHERE R.Rent_DueDate >= '1900-01-01'
AND R.Rent_DueDate <= '2020-12-31'
AND R.Rent_Statues = 'False'
```



Query 1

Optimization Done:

- Inner Join Instead of Nested loop join
- Non Clustered Index on : **`Rent_DueDate`**
- Non Clustered Index on : **`Rent_Statues`**

Action	CPU TIME	ELAPSED TIME
Query 1(No optimization)	9468 ms	8341 ms
Query 1(Inner Join and rewriting the query)	2667 ms	7026 ms
Query 1(Indexes Added)	2216 ms	6920 ms
Query 1(Mongo)	144,301 ms	

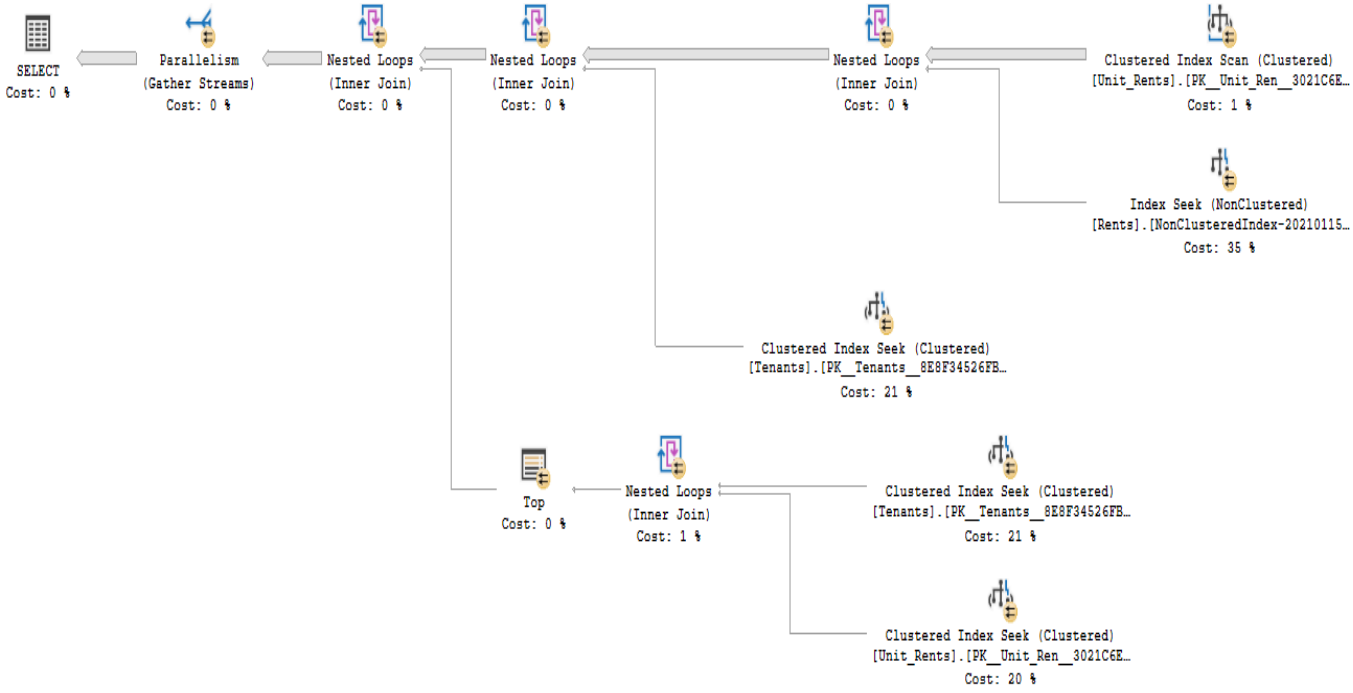
Parallel Query Processing Report (theoretically):

- We can make parallel execution as follows:
  - Clustered index scan on rents table
  - Clustered index scan on unit_rents table
  - Clustered index scan on tenants
  - Clustered index seek on tenants table

Query 2

Before Optimization:

```
SELECT T.Tenant_FirstName, T.Tenant_LastName
FROM Tenants T, Rents R , Unit_Rents UR
WHERE R.Rent_Statues = 'False'
        AND UR.Unit_Rents_TenantId = T.Tenant_Id
        AND UR.Unit_Rents_RentId = R.Rent_Id
        AND T.Tenant_Id
                        in (
                              SELECT      T.Tenant_Id
                              FROM Unit_Rents UR, Units U,Tenants T
                              WHERE T.Tenant_Id = UR.Unit_Rents_TenantId
                                    AND U.Unit_Id = UR.Unit_Rents_UnitId
                              )
```

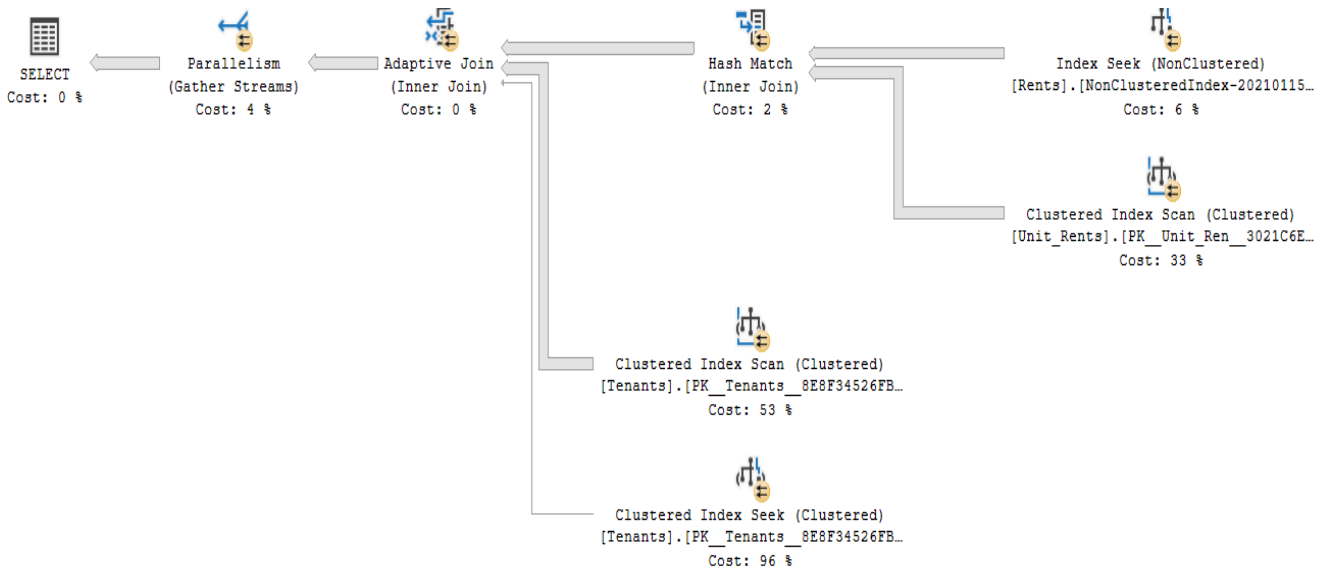


Rows Affected = 990,071 rows

Query 2

After Optimization

```
SELECT T.Tenant_FirstName, T.Tenant_LastName
FROM
Unit_Rents UR inner join Rents R on UR.Unit_Rents_RentId = R.Rent_Id
inner join
Units U on U.Unit_Id = UR.Unit_Rents_UnitId
inner join
Tenants T on T.Tenant_Id  = UR.Unit_Rents_TenantId
WHERE R.Rent_Statues = 'False'
```



Query 2

Optimization Done:

- Inner Join Instead of Nested loop join
- Non Clustered Index on : **Rent_Statues**

Action	CPU TIME	ELAPSED TIME
Query 2(No optimization)	9140 ms	5608 ms
Query 2(Inner Join and rewriting the query)	1705 ms	7084 ms
Query 2(Indexes Added)	1372 ms	5463 ms
Query 2(Mongo)	99,822 ms	

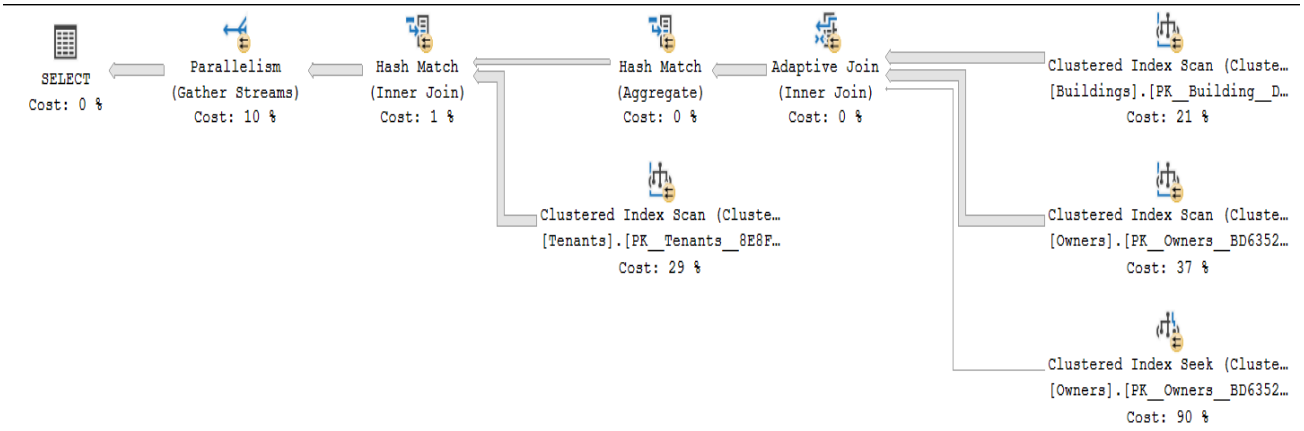
Parallel Query Processing Report (theoretically):

- We can make parallel execution as follows:

- Non Clustered index seek on rents table
- Clustered index scan on unit_rents table
- Clustered index scan on tenants
- Clustered index seek on tenants table

Query 3

```
SELECT *
FROM Tenants T
WHERE T.Tenant_FirstName in
(
      SELECT O.Owner_FirstName
      From Buildings B, Owners O
      WHERE B.Building_OwnerId = O.Owner_Id
)
```



Rows Affected = 1,979,936 rows

Query 3

Optimization Done:

- Changed **Owner_FirstName**, and **Tenant_FirstName** from VARCHAR(20) to VARCHAR(10).
- Added the query as a stored procedure

Action	**CPU TIME**	**ELAPSED TIME**
Query 3(No optimization)	**4907 ms**	**22588 ms**
Query 3(VARCHAR(10))	**4836 ms**	**16785 ms**
Query 3(stored procedure)	**5047 ms**	**15902 ms**
Query 1(Mongo)	**310,585 ms**	

Parallel Query Processing Report (theoretically):

- We can make parallel execution as follows:

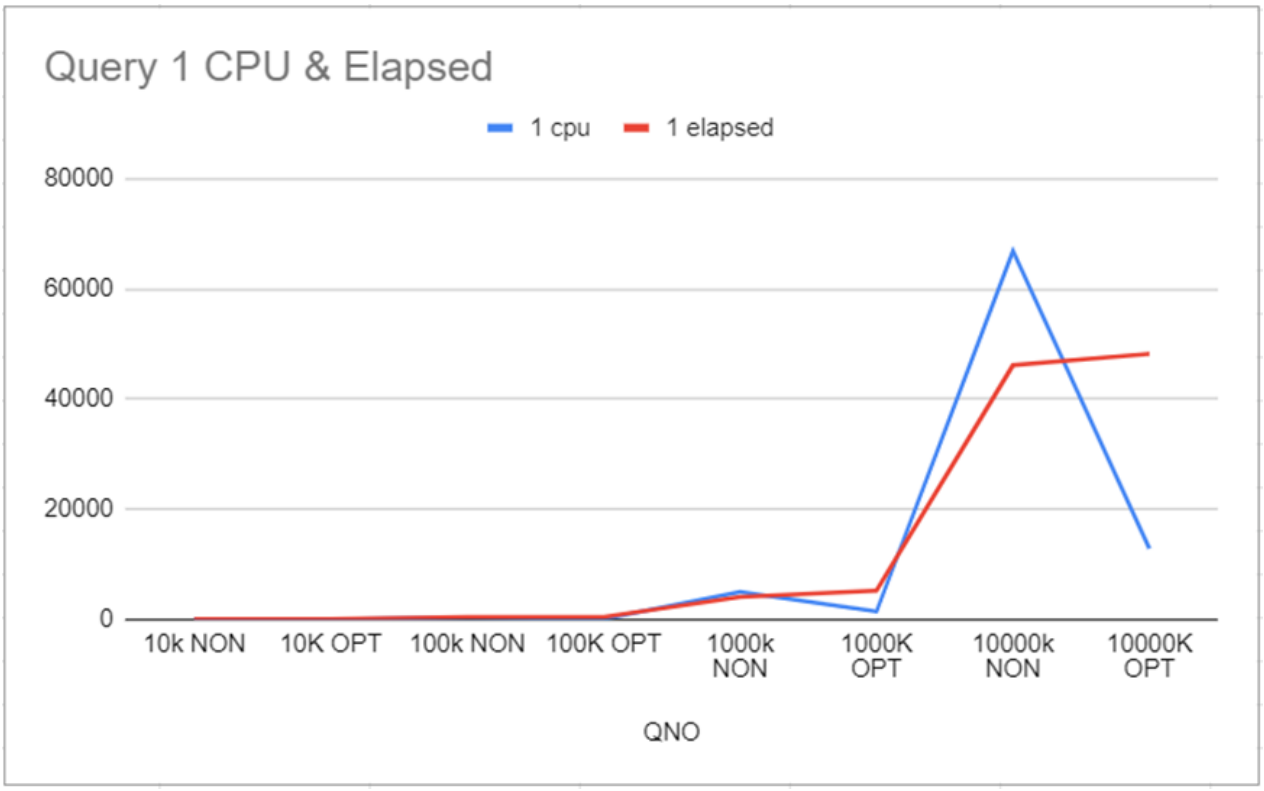
- Clustered index scan on buildings table
- Clustered index scan on Owners table
- Clustered index seek on Owners table
- Clustered index scan on Tenants table

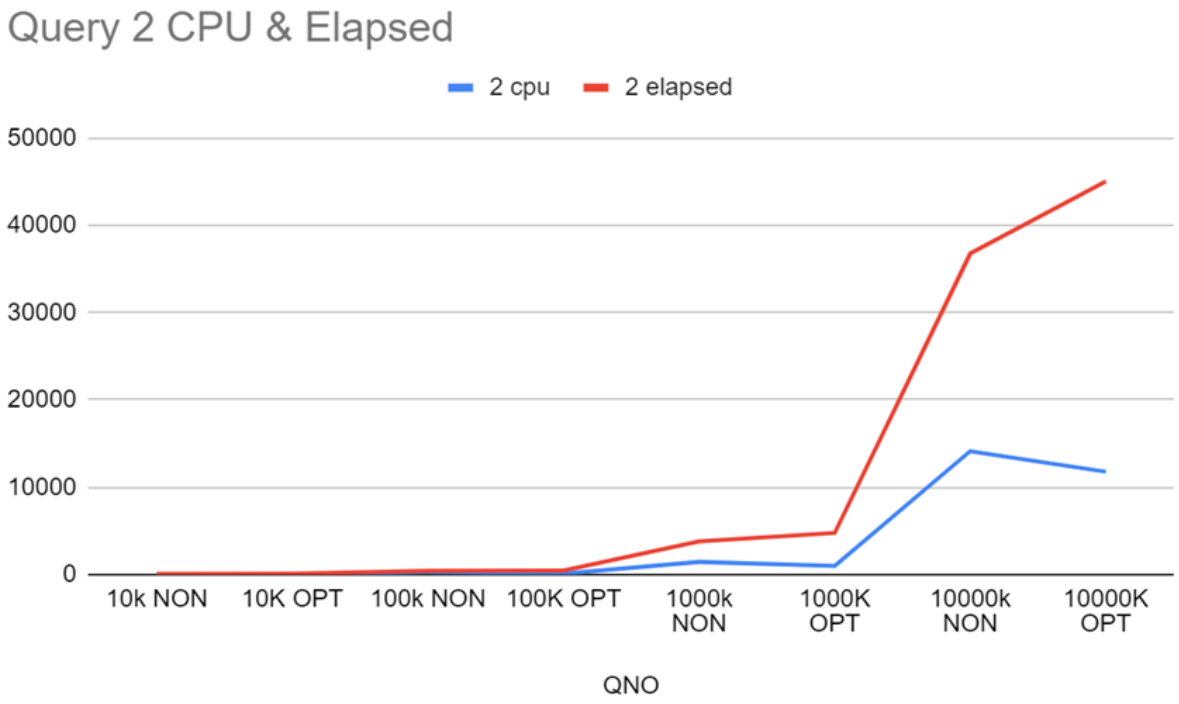
New Database Statistics

- The Tenants Max row size decreased by 10 bytes
- The Owners Max row size decreased by 10 bytes
- 2 indexes were added to the rents table

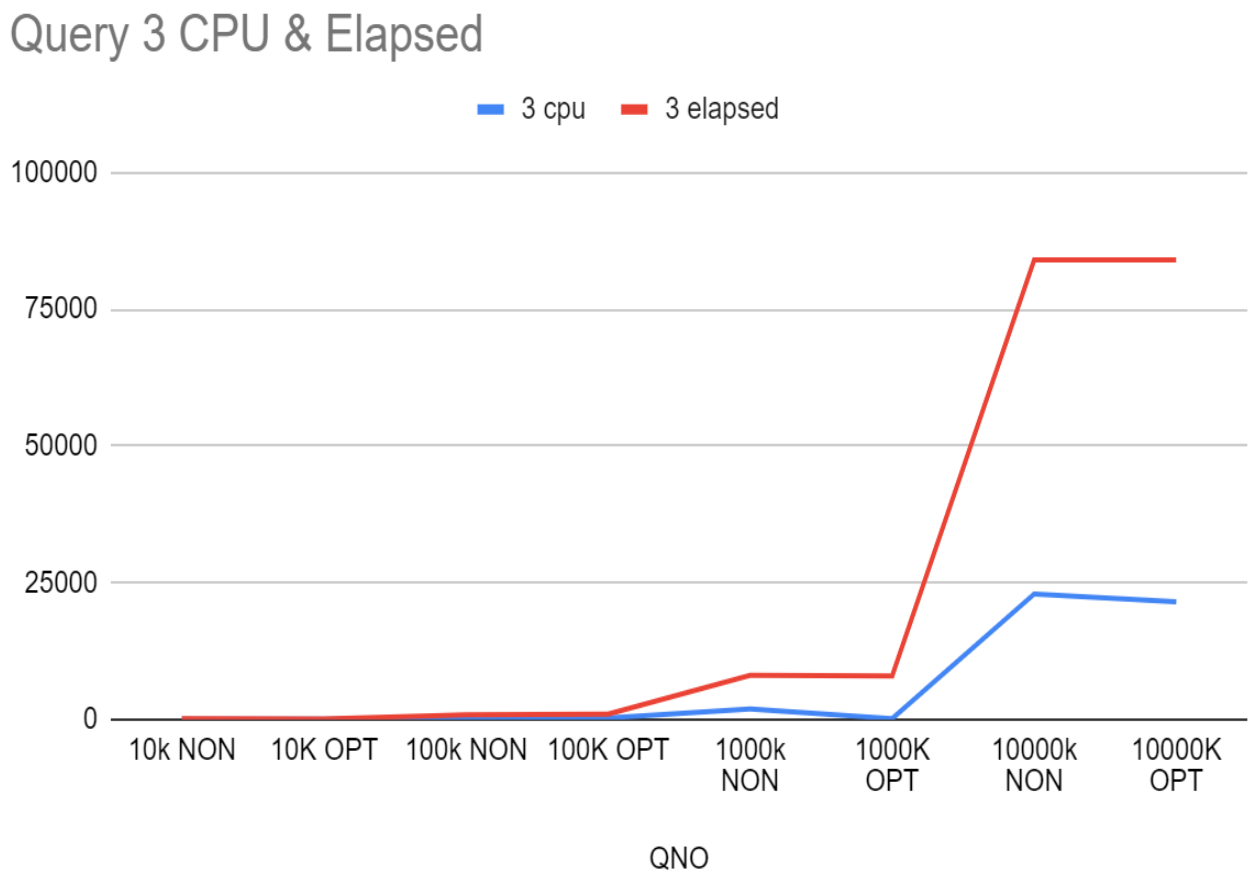
Table Name	Row Count	Main Key	Indexes	FK	Identity Column	Max Row Size (Bytes)
Buildings	1076560	YES	2	1	YES	96
Contacts	2000000	YES	1	0	YES	134
Employees	2000000	YES	1	3	YES	139
Expenses	2000000	YES	1	1	YES	43
Leases	2000000	YES	1	1	YES	46
Owners	2000000	YES	1	0	YES	86
Rents	2000000	YES	3	2	YES	24
Tenants	2000000	YES	1	1	YES	62
Unit_Rents	2000000	YES	1	3	YES	12
Units	2000000	YES	1	1	YES	65
Users	969	YES	1	0	YES	18

Time Analysis





Time Analysis



Optimizations

The Enhancement in the Schema

- No modifications in the schema since all the relations between the tables normalized before

The Enhancement of Memory Management

- Added some queries as stored procedure to enhance the memory and cache usage.
- We have changed the type/reduced the size of the some column like:
  - Changed Owner_FirstName, and Tenant_FirstName from CHAR(20) to VARCHAR(10).

The Modification in Indexes

- We have added some indexes to boost the selection and update of data
  - Non Clustered Index on : **Rent_DueDate**
  - Non Clustered Index on : **Rent_Statues**
  - Non Clustered Index on **: Rent_Statues**

Conclusion

- In time Optimization: index tuning and query optimization affected the time greatly, it improved by almost 50% in most of the queries
- Simple queries took a lot of time even though the schema was normalized correctly, the main key was which columns to choose as the indexes, specially from foreign keys
- Memory Enhancements were the same in most of the queries like data selection and data update, but in data selection, the most complicated query, there was a great improvement and reduction in memory usage.
- In data insertion query, as the time optimization had great results, memory usage was notably increasing in change.
- When trying different database sizes, there was barley any increase in memory or time, which is expected due to all the made optimizations.
- The overall optimization had good results