Szymon Kasprzycki 193065, Konrad Bochenek 193747

Optimization report for ThunderRentals Data Warehouse

1. Introduction

Aim of the laboratory:

The aim of the task is to show issues concerning various physical cube models and aggregation design.

Testing environment physical specification:

CPU: Intel(R) Core(TM) i7-11800H @ 2.30GHz
Ram: 32GB (2x16GB) DDR4 3200Mhz, in dual-channel mode
Disk: Samsung 980 PRO 1TB Nvme (installed on PCI-E 4.0 slot)
OS: Windows 11 x64 (Version 23H2, build 22631.3593)

Testing environment software specification:

a) Microsoft SQL Server 2019, 15.0.2000.5, Developer Edition (64-bit) (Build 22631)
b) SQL Server Management Studio, version 19.2.56.2
c) Microsoft Visual Studio Enterprise 2019, version 16.11.34
d) Microsoft .NET 4.8.09032
e) Microsoft SQL Server Integration Services Projects version 16.0.5467.0
f) SQL Server Profiler version 16.0.4001.1

Query description:

1. **Query 1** - shows which cities see the highest demand for scooter rentals for the analyzed month compared to the previous month, uses:
   a. calculated member
   b. WHERE clause
   c. MDX function operating on the dimension hierarchy
2. **Query 2** - shows what is the number of rentals per age group for the analyzed month and for the previous month, uses:
   a. MDX function operating on the dimension hierarchy
3. **Query 3** - shows the average number of rentals per user, per month.
4. **Query 4** - shows the average battery life remaining on a scooter at the start of each rental in the current and previous month.
5. **Query 5** - compares the percentage distribution of rentals depending on gender for the current month with the percentage distribution for the previous month, uses:

    a. calculated member
    b. numerical operation
6. **Query 6** - compares the average expected travel distance at the end of the rental between cities, uses:
    a. MDX function operating on the dimension hierarchy
7. **Query 7** - shows what is the average rental duration this month compared to the previous month.
8. **Query 8** - shows the average battery life remaining on scooters at the end of each rental.
9. **Query 9** - shows what is the average travel distance per rental from each month for the last year, uses:
    a. MDX function operating on the dimension hierarchy
10. **Query 10** - shows which day of the week has the highest scooter usage for the last month
11. **Query 11** - shows how many rentals does a scooter have on average per day in the current and previous month, uses:
    a. calculated member
    b. numerical operation
12. **Query 12** - compares the share of rentals between various models of scooters for the last year.
13. **Query 13** - compares the number of rentals between various areas in cities.
14. **Query 14** - shows what is the number of rentals per age group for the analyzed month and for the previous month, uses:
    a. Top function

Relational data warehouse size report:

Table Name	**# Records**	**Reserved (KB)**	**Data (KB)**	**Indexes (KB)**	**Unused (KB)**
Areas [DT]	8	72	8	8	56
Customers [DT]	55,386	7,816	7,632	40	144
Date [DT]	1,826	264	136	16	112
Junk [DT]	2	72	8	8	56
Rentals [FT]	649,998	648,968	621,112	8	27,848
Scooters [DT]	1,050	200	56	16	128
ScooterState [DT]	1,199,998	63,112	52,048	200	10,864
Time [DT]	86,400	5,448	4,976	24	448
TOTAL	1,994,668	725,952	685,986	320	39,656

Chosen aggregations:

a) **Scooters measurement group** - aggregated by scooter model, scooter producer, month, day of week.
b) **Rentals measurement group** - aggregated by rental end year, rental end month, rental end day of week, scooter model, scooter producer.

2. Testing

In this section, the data collected throughout the process is presented. The process consisted of measuring queries (mentioned above) executing times, as well as cube processing times for different settings of data partitioning. Each testing unit was measured 5 at least 5 times to obtain trustworthy results. The measures were performed for three different partitioning modes: MOLAP, ROLAP and HOLAP. Before each query execution, the cached data in the data warehouse were cleared.

Additional note: if any query execution times are marked with ~~crossed out red color~~, then they are treated as outliers and they are not taken into consideration to calculate average time.

a) Measured query time executions without aggregations

MOLAP

Query No	Query execution times [ms]	Average [ms]
Query 1	6, 7, 5, 6, 6	6.0
Query 2	7, 5, 5, 7, 6	6.0
Query 3	206, 200, 200, 199, 199	200.8
Query 4	5, 5, 5, 4, 4	4.6
Query 5	6, 5, 5, 6, 6	5.6
Query 6	11, 9, 8, 8, 9	9.0
Query 7	4, 4, 5, 4, 4	4.2
Query 8	7, 6, 7, 6, 6	6.4
Query 9	6, 4, 4, 5, 6	5.0
Query 10	3, 3, 3, 3, 4	3.2
Query 11	54, 55, 55, 55, 56	55.0
Query 12	5, 7, 5, 6, 5	5.6
Query 13	9, 9, 9, 9, 9	9.0
Query 14	8, 10, 9, 10, 10	9.4

ROLAP

Query No	Query execution times [ms]	Average [ms]
Query 1	6, 2, 2, 5, 5	4.0
Query 2	6, 6, 7, 6, 6	6.2
Query 3	204, 197, 208, 198, 200	201.4
Query 4	5, 5, 4, 4, 5	4.6
Query 5	5, 5, 4, 5, 5	4.8
Query 6	11, 9, 10, 9, 9	9.6
Query 7	3, 3, 5, 4, 4	3.8
Query 8	5, 5, 6, 5, 6	5.4
Query 9	5, 5, 6, 6, 5	5.4
Query 10	3, 3, 2, 3, 3	2.8
Query 11	~~174~~, 102, 100, 97, 96, 97	98.4
Query 12	5, 6, 5, 5, 4	5.0
Query 13	9, 9, 9, 9, 9	9.0
Query 14	9, 9, 9, 10, 9	9.2

HOLAP

Query No	Query execution times [ms]	Average [ms]
Query 1	6, 6, 5, 6, 6	5.8
Query 2	8, 8, 6, 6, 8	7.2
Query 3	209, 207, 207, 208, 206	207.4
Query 4	5, 5, 4, 4, 5	4.6
Query 5	5, 5, 4, 5, 4	4.6
Query 6	10, 9, 9, 9, 9	9.2
Query 7	4, 3, 4, 5, 4	4.0
Query 8	5, 6, 6, 6, 5	5.6
Query 9	5, 6, 5, 5, 5	5.2
Query 10	3, 3, 4, 3, 3	3.2
Query 11	115, 101, 102, 102, 106	105.2
Query 12	7, 6, 6, 6, 5	6.0

Query 13	10, 9, 9, 8, 9	9.0
Query 14	9, 9, 10, 10, 9	9.4

b) Measured cube processing time without aggregations

Type	**Cube processing times [ms]**	**Average [ms]**
MOLAP	8711, 6944, 7562, 7216, 6839	7454.4
ROLAP	6927, 7125, 6955, 6953, 7251	7042.2
HOLAP	6912, 6810, 6796, 6907, 7287	6942.4

c) Measured query time executions with aggregations

MOLAP

Query No	**Query execution times [ms]**	**Average [ms]**
Query 1	3, 2, 3, 3, 3	2.8
Query 2	8, 7, 6, 8, 6	7.0
Query 3	184, 183, 185, 186, 182	184.0
Query 4	2, 1, 1, 2, 1	1.4
Query 5	~~7~~, 5, 4, 5, 4, 5	4.6
Query 6	1, 1, 1, 2, 2	1.4
Query 7	2, 2, 2, 2, 2	2.0
Query 8	1, 1, 2, 1, 1	1.2
Query 9	2, 1, 1, 2, 1	1.4
Query 10	1, 1, 2, 2, 1	1.4
Query 11	15, 14, 14, 14, 15	14.4
Query 12	1, 2, 1, 1, 1	1.2
Query 13	2, 1, 1, 1, 2	1.4
Query 14	3, 2, 2, 2, 1	2.0

ROLAP

Query No	Query execution times [ms]	Average [ms]
Query 1	5, 2, 2, 3, 2	2.8
Query 2	7, 6, 6, 7, 6	6.4
Query 3	185, 185, 184, 185, 185	184.8
Query 4	2, 2, 2, 2, 2	2.0
Query 5	5, 5, 4, 5, 4	4.6
Query 6	1, 2, 1, 1, 2	1.4
Query 7	2, 2, 1, 2, 1	1.2
Query 8	1, 1, 2, 1, 2	1.4
Query 9	2, 2, 1, 1, 2	1.2
Query 10	1, 1, 1, 1, 1	1.0
Query 11	~~117~~, 100, 95, 93, 98, 94	96.0
Query 12	2, 1, 1, 1, 1	1.2
Query 13	1, 2, 2, 2, 1	1.6
Query 14	2, 2, 2, 2, 2	2.0

HOLAP

Query No	Query execution times [ms]	Average [ms]
Query 1	3, 3, 3, 3, 3	3.0
Query 2	8, 7, 7, 6, 6	6.8
Query 3	187, 186, 192, 186, 184	187.0
Query 4	3, 2, 1, 2, 2	2.0
Query 5	6, 5, 5, 5, 4	5.0
Query 6	2, 1, 1, 1, 1	1.2
Query 7	2, 1, 2, 1, 1	1.4
Query 8	1, 2, 1, 1, 1	1.2
Query 9	2, 2, 1, 1, 1	1.4
Query 10	2, 1, 1, 2, 1	1.4
Query 11	15, 15, 14, 15, 15	14.8
Query 12	2, 1, 1, 2, 1	1.4

Query 13	1, 2, 2, 1, 1	1.4
Query 14	2, 2, 1, 1, 2	1.6

d) Measured cube processing time with aggregations

Type	**Cube processing times [ms]**	**Average [ms]**
MOLAP	10031, 9688, 9438, 10109, 9609	9775.0
ROLAP	7688, 7500, 7531, 7516, 7656	7578.2
HOLAP	7531, 7656, 7734, 7688, 7391	7600.0

3. Results

a) MOLAP vs ROLAP vs HOLAP comparison

Query execution times without aggregations

	Average query execution time [ms]		
Query No	**MOLAP**	**ROLAP**	**HOLAP**
Query 1	6.0	4.0	5.8
Query 2	6.0	6.2	7.2
Query 3	200.8	201.4	207.4
Query 4	4.6	4.6	4.6
Query 5	5.6	4.8	4.6
Query 6	9.0	9.6	9.2
Query 7	4.2	3.8	4.0
Query 8	6.4	5.4	5.6
Query 9	5.0	5.4	5.2
Query 10	3.2	2.8	3.2
Query 11	55.0	98.4	105.2
Query 12	5.6	5.0	6.0
Query 13	9.0	9.0	9.0
Query 14	9.4	9.2	9.4

Query execution times with aggregations

	Average query execution time [ms]		
Query No	**MOLAP**	**ROLAP**	**HOLAP**
Query 1	2.8	2.8	3.0
Query 2	7.0	6.4	6.8
Query 3	184.0	184.8	187.0
Query 4	1.4	2.0	2.0
Query 5	4.6	4.6	5.0
Query 6	1.4	1.4	1.2
Query 7	2.0	1.2	1.4
Query 8	1.2	1.4	1.2
Query 9	1.4	1.2	1.4
Query 10	1.4	1.0	1.4
Query 11	14.4	96.0	14.8
Query 12	1.2	1.2	1.4
Query 13	1.4	1.6	1.4
Query 14	2.0	2.0	1.6

b) MOLAP with and without aggregations, query execution time comparison

	Average query execution time [ms]	
Query No	**without aggregations**	**with aggregations**
Query 1	6.0	2.8
Query 2	6.0	7.0
Query 3	200.8	184.0
Query 4	4.6	1.4
Query 5	5.6	4.6
Query 6	9.0	1.4
Query 7	4.2	2.0
Query 8	6.4	1.2
Query 9	5.0	1.4

Query 10	3.2	1.4
Query 11	55.0	14.4
Query 12	5.6	1.2
Query 13	9.0	1.4
Query 14	9.4	2.0

c) MOLAP with and without aggregations, cube processing time comparison

Type	**Average [ms]**
without aggregations	7454.4
with aggregations	9775.0

d) ROLAP with and without aggregations, query execution time comparison

	Average query execution time [ms]	
Query No	**without aggregations**	**with aggregations**
Query 1	4.0	2.8
Query 2	6.2	6.4
Query 3	201.4	184.8
Query 4	4.6	2.0
Query 5	4.8	4.6
Query 6	9.6	1.4
Query 7	3.8	1.2
Query 8	5.4	1.4
Query 9	5.4	1.2
Query 10	2.8	1.0
Query 11	98.4	96.0
Query 12	5.0	1.2
Query 13	9.0	1.6
Query 14	9.2	2.0

e) ROLAP with and without aggregations, cube processing time comparison

Type	**Average [ms]**
without aggregations	7042.2
with aggregations	7578.2

f) HOLAP with and without aggregations, query execution time comparison

	Average query execution time [ms]	
Query No	**without aggregations**	**with aggregations**
Query 1	5.8	3.0
Query 2	7.2	6.8
Query 3	207.4	187.0
Query 4	4.6	2.0
Query 5	4.6	5.0
Query 6	9.2	1.2
Query 7	4.0	1.4
Query 8	5.6	1.2
Query 9	5.2	1.4
Query 10	3.2	1.4
Query 11	105.2	14.8
Query 12	6.0	1.4
Query 13	9.0	1.4
Query 14	9.4	1.6

g) HOLAP with and without aggregations, cube processing time comparison

Type	**Average [ms]**
without aggregations	6942.4
with aggregations	7600.0

4. Results discussion

First it is important to say what the theory states. In typical environment, under standard assumptions:

- the query execution times should be the quickest in MOLAP cube mode,
- the cube processing time should be the slowest in MOLAP cube mode,
- the query execution times should be average in HOLAP cube mode,
- the cube processing time should be average in HOLAP cube mode,
- the query execution times should be the slowest in ROLAP cube mode,
- the cube processing time should be the quickest in ROLAP cube mode,
- the query execution with aggregations should be quicker than without aggregations.

Our results are as follows:
Even though in theory MOLAP should be the most efficient for query execution and ROLAP for cube processing, in practice, our results are a bit different. Firstly, we analyze times without aggregations. The average query execution time appeared to be the best in ROLAP, where out of 14 queries 8 had the shortest execution time. However MOLAP, which theoretically should be the best, wasn't so far behind with 7 queries with the shortest execution time. HOLAP had only 3 queries with the shortest execution time. This is somewhat inconsistent with the theory according to which its results should be somewhere in the middle. The discrepancies between theoretical expectations and practical results may come from several factors like data characteristics or query structure. Moreover, the testing environment may also cause some variations from the assumptions because for the trustworthy results, it should be clear, prepared straight for this task. So, in our case, ROLAP went from theoretically slowest query execution time to the quickest. In cube processing time, ROLAP which should be the fastest had the second average cube processing time, little behind HOLAP. MOLAP was the slowest, so this appears to be consistent with the theory.

Now, we will analyze times with aggregations. The average query execution time the roles were reversed, because now MOLAP had 8 queries with the quickest execution time, while ROLAP - 7. For HOLAP it's one query more, so 4. Still ROLAP has been performing better than in theory suppose, but with aggregations MOLAP appeared to have the biggest number of queries with the fastest execution time, consistent with theory. Worth noticing is the average execution time for query 11, where ROLAP is very much slower than MOLAP and HOLAP, so we think this query quite well reflects what is supposed to happen according to theory. Finally, average cube processing time with aggregations is perfectly consistent with the theory - ROLAP had the lowest average cube processing time, HOLAP was somewhat in the middle, and the MOLAP had the biggest.

When it comes to analyzing differences between cube and query processing times with or without aggregations the results are pretty similar in MOLAP, ROLAP and HOLAP. 13 out of 14 queries had better average query execution time in all of them, and in all of them the average cube processing time without aggregations was noticeably quicker than the one with aggregations. There is no doubt that the aggregations are working as expected. They are optimizing the performance of our queries by several percent while extending the processing time a little.