# Data Warehouse Optimization - Report

Piotr Sulewski, Tomasz Sankowski

January 2025

## Contents

# 1. Aim of the Laboratory

The aim of this task is to address issues related to various physical cube models and the design of aggregations. The focus is on demonstrating the impact of these models on performance and usability in a data warehouse environment.

# 2. Preliminary Assumptions

## 2.1. Database Specifications

- **Size of the Data Warehouse:** 976.00 MB
- **Number of rows in the monitoring fact table (main):** 1,176,609

## 2.2. Testing Environment

Measurements were taken on the following setup:

- **Processor:** Intel Core i7-6700K
- **Memory:** 16 GB RAM
- **Storage:** 1 TB NVMe SSD
- **Operating System:** Windows 10
- **Tools Used:**
  - SQL Server Management Studio (SSMS) with SQL Server Profiler extension
  - Visual Studio 2022 with Data Tools extension

## 2.3. Conditions During Testing

To ensure reliable results, the following conditions were maintained:

- Only the necessary applications were running during measurements:
  - SQL Server Management Studio (SSMS)
  - Visual Studio 2022
  - A web browser with the instructions open

# 3. Theoretical Assumptions

Table 1: Comparison of MOLAP, HOLAP, and ROLAP

| | MOLAP | HOLAP | ROLAP |
|---|---|---|---|
| **Querying time** | Short | Moderate[1] | Long |
| **Processing time** | Long | Moderate[2] | Short |
| **Total size** | Large[3] | Moderate | Small |

[1] Querying time can be short if well-designed aggregations are used.

[2] Processing time is moderate, but if no aggregations are designed, it will be short.

[3] Total size can be smaller if no aggregations are designed for measure groups.

# 4. Testing

## 4.1. Brief description of the queries

Testing query execution times for different models, with and without defined aggregations. Testing cube processing times in the same testing settings

Figure 1: Query 2

Which car models generate the highest earnings per vehicle?

```
WITH
MEMBER [Measures].[Vehicle Count] AS
    COUNT(EXISTING [Car].[Car ID].[Car ID])

MEMBER [Measures].[Revenue Per Vehicle] AS
    IIF(
        [Measures].[Vehicle Count] > 0,
        [Measures].[Rental Cost] / [Measures].[Vehicle Count],
        NULL
    )

SELECT
    NON EMPTY
    TOPCOUNT(
        [Car].[Brand].[Brand].MEMBERS *
                [Car].[Model].[Model].MEMBERS,
        10,
        [Measures].[Revenue Per Vehicle]
    ) ON ROWS,
    [Measures].[Revenue Per Vehicle] ON COLUMNS
FROM
    [Car Sharing];
```

Figure 2: Query 5

Create a ranking of the 5 car models with the highest total mileage driven by customers in the last year.

```
SELECT
NON EMPTY
    TOPCOUNT(
                [Car].[Brand].[Brand].MEMBERS *
        [Car].[Model].[Model].MEMBERS,
        5,
        [Measures].[Driven Km]
    ) ON ROWS,
    {[Measures].[Driven Km]} ON COLUMNS
FROM
    [Car Sharing]
WHERE
    ([Date].[DateHierarchy].[Year].&[2023]);
```

Figure 3: Query 9

Considering the number of rides taken and the number of reported damages, what is the theoretical probability of causing damage within each age group of users?

```
WITH
  MEMBER [Measures].[Damaged Rides] AS
    (
      [Was Damaged].[Was Damaged].[Damaged],
      [Measures].[Rental Count]
    )
  MEMBER [Measures].[Probability of Damage] AS
    IIF(
      [Measures].[Rental Count] = 0,
      null,
      [Measures].[Damaged Rides] / [Measures].[Rental Count]
    )
SELECT
  { [Measures].[Probability of Damage] } ON COLUMNS,
  [User].[Age Category].Children ON ROWS
FROM
  [Car Sharing];
```

## 4.2. MOLAP Performance

Table 2: Performance Comparison of MOLAP with and without Aggregators

| | MOLAP | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Without Aggregators | | | | With Aggregators | | | |
| | Cube Processing | Query 2 | Query 5 | Query 9 | Cube Processing | Query 2 | Query 5 | Query 9 |
| | 4740 | 45 | 43 | 43 | 4910 | 4 | 2 | 2 |
| | 4635 | 43 | 46 | 44 | 4830 | 4 | 2 | 2 |
| | 4567 | 44 | 44 | 43 | 4907 | 4 | 3 | 3 |
| | 4732 | 45 | 43 | 44 | 4879 | 4 | 2 | 2 |
| | 4629 | 43 | 40 | 43 | 4867 | 4 | 5 | 1 |
| Average | 4660.6 | 44 | 43.2 | 43.4 | 4878.6 | 4 | 2.8 | 2 |
| SD | 66.05 | 0.89 | 1.94 | 0.49 | 29.29 | 0.0 | 1.17 | 0.63 |

## 4.3. HOLAP Performance

Table 3: Performance Comparison of HOLAP with and without Aggregators

| | HOLAP | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Without Aggregators | | | | With Aggregators | | | |
| | Cube Processing | Query 2 | Query 5 | Query 9 | Cube Processing | Query 2 | Query 5 | Query 9 |
| | 483 | 141 | 135 | 301 | 513 | 5 | 2 | 2 |
| | 487 | 135 | 136 | 295 | 493 | 4 | 2 | 2 |
| | 479 | 136 | 150 | 299 | 498 | 5 | 2 | 1 |
| | 481 | 137 | 132 | 294 | 513 | 5 | 2 | 2 |
| | 482 | 132 | 163 | 316 | 558 | 4 | 2 | 1 |
| Average | 482.4 | 136.2 | 143.8 | 301.0 | 515.0 | 4.6 | 2.0 | 1.6 |
| SD | 2.65 | 2.93 | 11.16 | 7.92 | 22.93 | 0.49 | 0.0 | 0.49 |

## 4.4. ROLAP Performance

Table 4: Performance Comparison of ROLAP with and without Aggregators

| | ROLAP | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Without Aggregators | | | | With Aggregators | | | |
| | Cube Processing | Query 2 | Query 5 | Query 9 | Cube Processing | Query 2 | Query 5 | Query 9 |
| | 447 | 131 | 148 | 306 | 511 | 138 | 154 | 309 |
| | 526 | 135 | 142 | 307 | 566 | 139 | 137 | 305 |
| | 471 | 134 | 140 | 310 | 482 | 149 | 137 | 296 |
| | 487 | 137 | 151 | 299 | 499 | 133 | 139 | 323 |
| | 502 | 138 | 132 | 309 | 486 | 132 | 145 | 301 |
| Average | 486.6 | 135.0 | 142.6 | 306.2 | 508.8 | 138.2 | 142.4 | 306.8 |
| SD | 26.84 | 2.45 | 6.62 | 3.87 | 30.37 | 6.05 | 6.50 | 9.17 |

## 4.5. Comparison of processing time for different cube models

To achieve optimal results of the processing time of a cube we decided to take approximately 5 samples for each modification. The obtained results are presented in the following tables

### 4.5.1 Comparison of cube processing performance

Table 5: Performance Comparison of Cube Processing with and without Aggregators in different cubes

| | Cube Processing | | | | | |
|---|---|---|---|---|---|---|
| | Without Aggregators | | | With Aggregators | | |
| | MOLAP | HOLAP | ROLAP | MOLAP | HOLAP | ROLAP |
| Average | 4660.6 | 482.4 | 486.6 | 4878.6 | 515.0 | 508.8 |
| SD | 66.05 | 2.65 | 26.64 | 29.29 | 22.93 | 30.37 |

### 4.5.2 Comparison of Query 2 performance

Table 6: Performance Comparison of Query 2 with and without Aggregators in different cubes

| | Query 2 | | | | | |
|---|---|---|---|---|---|---|
| | Without Aggregators | | | With Aggregators | | |
| | MOLAP | HOLAP | ROLAP | MOLAP | HOLAP | ROLAP |
| Average | 44.0 | 136.2 | 135.0 | 4.0 | 4.6 | 138.2 |
| SD | 0.89 | 2.93 | 2.45 | 0.0 | 0.49 | 6.05 |

### 4.5.3 Comparison of Query 5 performance

Table 7: Performance Comparison of Query 5 with and without Aggregators in different cubes

| | Query 5 | | | | | |
|---|---|---|---|---|---|---|
| | Without Aggregators | | | With Aggregators | | |
| | MOLAP | HOLAP | ROLAP | MOLAP | HOLAP | ROLAP |
| Average | 43.2 | 143.8 | 142.6 | 2.8 | 2.0 | 142.4 |
| SD | 1.94 | 11.6 | 6.62 | 1.17 | 0.0 | 6.50 |

### 4.5.4 Comparison of Query 9 performance

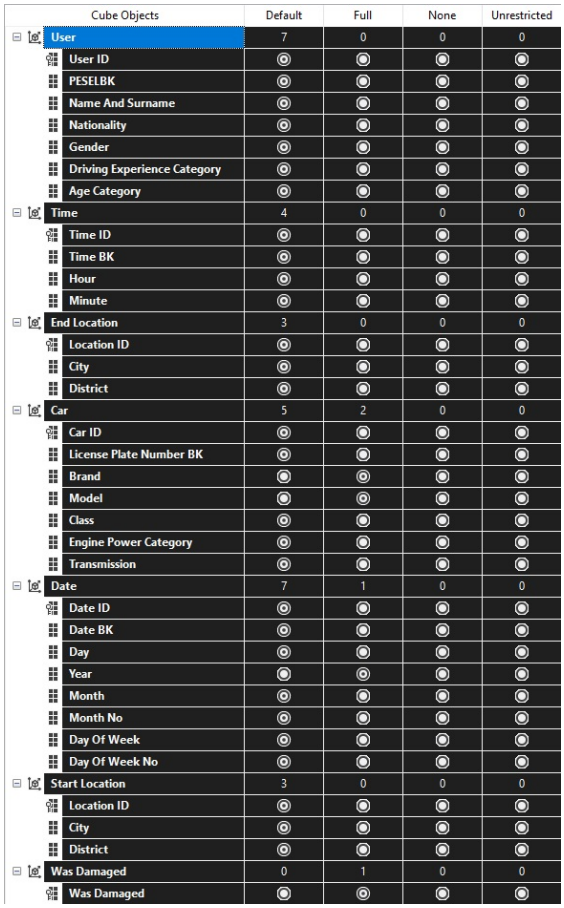
Table 8: Performance Comparison of Query 9 with and without Aggregators in different cubes

| | Query 9 | | | | | |
|---|---|---|---|---|---|---|
| | Without Aggregators | | | With Aggregators | | |
| | MOLAP | HOLAP | ROLAP | MOLAP | HOLAP | ROLAP |
| Average | 43.4 | 301.0 | 306.2 | 2.0 | 1.6 | 306.8 |
| SD | 0.49 | 7.92 | 3.87 | 0.63 | 0.49 | 9.17 |

# 5. Cache and Aggregation Settings

## 5.1. Aggregation Design

The aggregations were designed with the "Perforfamous gain reaches" option set to 100%, which gave us 7 aggregations with an optimization level of 40% (4.4MB).



| Cube Objects | Default | Full | None | Unrestricted |
|---|---|---|---|---|
| User | 7 | 0 | 0 | 0 |
| User ID | ◉ | ◉ | ◉ | ◉ |
| PESELBK | ◉ | ◉ | ◉ | ◉ |
| Name And Surname | ◉ | ◉ | ◉ | ◉ |
| Nationality | ◉ | ◉ | ◉ | ◉ |
| Gender | ◉ | ◉ | ◉ | ◉ |
| Driving Experience Category | ◉ | ◉ | ◉ | ◉ |
| Age Category | ◉ | ◉ | ◉ | ◉ |
| Time | 4 | 0 | 0 | 0 |
| Time ID | ◉ | ◉ | ◉ | ◉ |
| Time BK | ◉ | ◉ | ◉ | ◉ |
| Hour | ◉ | ◉ | ◉ | ◉ |
| Minute | ◉ | ◉ | ◉ | ◉ |
| End Location | 3 | 0 | 0 | 0 |
| Location ID | ◉ | ◉ | ◉ | ◉ |
| City | ◉ | ◉ | ◉ | ◉ |
| District | ◉ | ◉ | ◉ | ◉ |
| Car | 5 | 2 | 0 | 0 |
| Car ID | ◉ | ◉ | ◉ | ◉ |
| License Plate Number BK | ◉ | ◉ | ◉ | ◉ |
| Brand | ◉ | ◉ | ◉ | ◉ |
| Model | ◉ | ◉ | ◉ | ◉ |
| Class | ◉ | ◉ | ◉ | ◉ |
| Engine Power Category | ◉ | ◉ | ◉ | ◉ |
| Transmission | ◉ | ◉ | ◉ | ◉ |
| Date | 7 | 1 | 0 | 0 |
| Date ID | ◉ | ◉ | ◉ | ◉ |
| Date BK | ◉ | ◉ | ◉ | ◉ |
| Day | ◉ | ◉ | ◉ | ◉ |
| Year | ◉ | ◉ | ◉ | ◉ |
| Month | ◉ | ◉ | ◉ | ◉ |
| Month No | ◉ | ◉ | ◉ | ◉ |
| Day Of Week | ◉ | ◉ | ◉ | ◉ |
| Day Of Week No | ◉ | ◉ | ◉ | ◉ |
| Start Location | 3 | 0 | 0 | 0 |
| Location ID | ◉ | ◉ | ◉ | ◉ |
| City | ◉ | ◉ | ◉ | ◉ |
| District | ◉ | ◉ | ◉ | ◉ |
| Was Damaged | 0 | 1 | 0 | 0 |
| Was Damaged | ◉ | ◉ | ◉ | ◉ |

Figure 4: Aggregation Usage

## 5.2. Cache removal

Figure 5: Cache

Removing cache

```
<ClearCache xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
    <Object>
        <DatabaseID>CarSharingImplementation</DatabaseID>
    </Object>
</ClearCache>
```

# 6. Discussion

## 6.1. Cube Processing

As observed, MOLAP required nearly ten times more time to process the cube compared to the ROLAP and HOLAP approaches, both with and without aggregations. This is because the MOLAP server needed to copy the entire warehouse storage into the cube, significantly slowing down the processing time. Additionally, adding aggregations only slightly affected the cube processing time, increasing it by approximately 10%. The higher processing time is due to the need to group measures by dimension members during cube processing. Our results confirm the assumption that MOLAP's cube processing time is considerably slower than ROLAP's and HOLAP's.

## 6.2. Query Execution

MOLAP query execution was, on average and depending on the specific query, 3 to 8 times faster than executing queries on the ROLAP server. The query execution time on the HOLAP server was similar to that of ROLAP. This supports our assumption that, despite requiring more time for cube processing, the MOLAP server significantly outperforms the other types of data warehouses in query execution because it already contains the necessary data within the cube.

Moreover, adding aggregations had no noticeable effect on the ROLAP server's query execution time. However, it made MOLAP and HOLAP almost equally efficient, with aggregations speeding up MOLAP query execution by about 20 times and HOLAP by approximately 70 to 150 times. This highlights the effectiveness of using aggregations. Aggregations accelerate query execution because the cube doesn't need to spend time grouping measures by dimension attributes; this grouping is already done during cube processing. HOLAP's query execution time was significantly better because, during the cube processing, aggregations are transferred to the cube, eliminating the need to retrieve data from the relational warehouse during aggregation.

## 6.3. General Conclusions

The tests on MOLAP, ROLAP, and HOLAP reveal key performance differences:

1. **Cube Processing Time:** MOLAP is slower to process the cube, as it requires transferring data into the multidimensional structure. ROLAP, however, directly accesses the relational database, making it faster. HOLAP offers a hybrid approach, with performance similar to ROLAP.

2. **Query Execution Time:** MOLAP accelerates query execution by storing data in the cube, significantly outperforming ROLAP. HOLAP, with aggregations, also sees major query performance improvements, as aggregated data is stored within the cube, bypassing the need for relational data access.

3. **Impact of Aggregations:** Adding aggregations enhances query performance in MOLAP and HOLAP, reducing query time by eliminating the need for grouping measures during execution. ROLAP shows no improvement as it still relies on the relational database for query execution.

4. **Overall Performance:** MOLAP excels in query speed but has slower processing times. ROLAP is faster for cube processing but slower in query execution. HOLAP offers a balanced solution, with faster query performance when aggregations are used, making it ideal for systems requiring both efficient data storage and quick queries.

In conclusion, the choice between OLAP models depends on specific needs: MOLAP is best for fast queries, ROLAP for quicker processing, and HOLAP offers a middle ground.