Stanisław Lemański, 193333 & Jakub Kobyliński, 193618

Data Warehouse Optimization – report

1. Aim of the laboratory

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

2. Preliminary assumptions

Size of the database (Data Warehouse): 720 MB

Number of rows in monitoring fact table (Material Transfers): 999590

Number of rows in the second fact table (Material Orders): 153323

Testing environment: The measurements were taken on a Dell Laptop equipped with an Intel Core i7-8850H processor, 32GB of RAM, and 512GB of Internal Storage, running Windows 11. To evaluate the processing time of a cube, we used SQL Server Management Studio 19 with the SQL Server Profiler extension. During the measurements, the only active applications on the laptop were SSMS, a web browser with open instructions, and Visual Studio 2022.

3. Testing

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

Brief description of the queries:

1. Query that shows the top 10 projects by "Total quantity transferred".

```
SELECT
    NON EMPTY { [Measures].[Total quantity transferred] } ON COLUMNS,
    NON EMPTY {
        TOPCOUNT(
            [Dim Project].[Project Name].[Project Name].ALLMEMBERS,
            10,
            [Measures].[Total quantity transferred]
        )
    }
    DIMENSION PROPERTIES MEMBER_CAPTION, MEMBER_UNIQUE_NAME ON ROWS
FROM [Developer Dw]
```

2. Query that shows the bottom 20 warehouses by "Total lost quantity," also displaying "Total transfer cost transfers."

```
SELECT
    NON EMPTY { [Measures].[Total lost quantity], [Measures].[Total transfer cost transfers] } ON COLUMNS,
    NON EMPTY {
        BOTTOMCOUNT(
            [Dim Warehouse].[Warehouse Name].[Warehouse Name].ALLMEMBERS,
            20,
            [Measures].[Total lost quantity]
        )
    }
    DIMENSION PROPERTIES MEMBER_CAPTION, MEMBER_UNIQUE_NAME ON ROWS
FROM [Developer Dw]
```

3. Query that shows the "Total transfer cost transfers" for all projects.

```
SELECT
    NON EMPTY { [Measures].[Total transfer cost transfers] } ON COLUMNS,
    NON EMPTY {
        [Dim Project].[Project Name].[Project Name].ALLMEMBERS
    }
    DIMENSION PROPERTIES MEMBER_CAPTION, MEMBER_UNIQUE_NAME ON ROWS
FROM [Developer Dw]
```

To achieve optimal results of the processing time of a cube we decided to take approximately 10 samples for each modification. The obtained results are presented in the following table.

Table 1 Processing time of cube and queries for MOLAP, ROLAP and HOLAP **without aggregations.**

Cube Processing			Query 1			Query 2			Query 3		
Molap	Rolap	Holap	Molap	Rolap	Holap	Molap	Rolap	Holap	Molap	Rolap	Holap
3395	2227	2303	69	148	101	72	72	75	99	93	88
3499	2312	2482	81	79	80	68	64	56	83	158	80
3645	2293	2344	81	81	78	58	71	74	80	83	80
3563	2193	2372	83	78	78	64	58	58	77	77	79
3464	2522	2430	83	77	79	59	58	61	81	79	77
3398	2386	2427	70	78	80	57	58	58	84	160	154
3711	2487	2363	86	78	77	59	64	60	77	82	80
3809	2336	2623	77	83	79	58	126	57	85	79	78
3495	2340	2468	79	78	76	57	58	59	84	78	78
3616	2452	2421	69	85	79	59	57	57	80	80	82

Afterwards, we decided to exclude outliers and calculate the mean and standard deviation for each column.

	Cube Processing			Query 1			Query 2			Query 3		
	Molap	Rolap	Holap	Molap	Rolap	Holap	Molap	Rolap	Holap	Molap	Rolap	Holap
Mean	3559,5	2354,8	2423,3	77,8	86,5	80,7	61,1	68,6	61,5	83	96,9	87,6
SD	135,73	107,8	89,7	6,32	21,76	7,24	5,17	20,92	7,01	6,3	33,04	23,53

Table 2 Processing time of cube and queries for MOLAP, ROLAP and HOLAP **with aggregations.**

Cube Processing			Query 1			Query 2			Query 3		
Molap	Rolap	Holap	Molap	Rolap	Holap	Molap	Rolap	Holap	Molap	Rolap	Holap
4624	5665	6319	18	17	21	4	4	2	20	19	19
4606	6101	5567	20	24	21	3	3	4	19	19	20
4818	6187	6210	21	28	23	4	3	3	27	19	22
4500	6241	5737	25	29	22	4	2	3	21	30	19
4512	5479	6366	19	19	21	4	4	3	18	21	20
4754	6123	6271	19	18	21	3	3	2	20	18	27
4661	6168	6468	17	20	21	4	2	5	18	24	20
4627	6231	5953	16	27	18	2	10	3	20	20	20
4471	6525	6147	22	24	25	5	3	3	19	20	20
4595	6416	5992	18	23	16	3	4	3	21	19	18

	Cube Processing			Query 1			Query 2			Query 3		
	Molap	Rolap	Holap	Molap	Rolap	Holap	Molap	Rolap	Holap	Molap	Rolap	Holap
Mean	4616,8	6113,6	6103	19,5	22,9	20,9	3,6	3,8	3,1	20	20,9	20,5
SD	996,46	2076,75	2043,33	30,9	35,69	31,76	29,55	35,35	30,07	33,17	44,14	37,55

Average time of processing cube and queries using MOLAP, ROLAP and HOLAP with and without aggregations.

	MOLAP		ROLAP		HOLAP	
	Aggr.	No aggr.	Aggr.	No aggr.	Aggr.	No aggr.
Querying speed (for 3 different queries)	19,5	77,8	22,9	86,5	20,9	80,7
	3,6	61,1	3,8	68,6	3,1	61,5
	20	83	20,9	96,9	20,5	87,6
Processing time	4616,8	3559,5	6113,6	2354,8	6103	2423,3

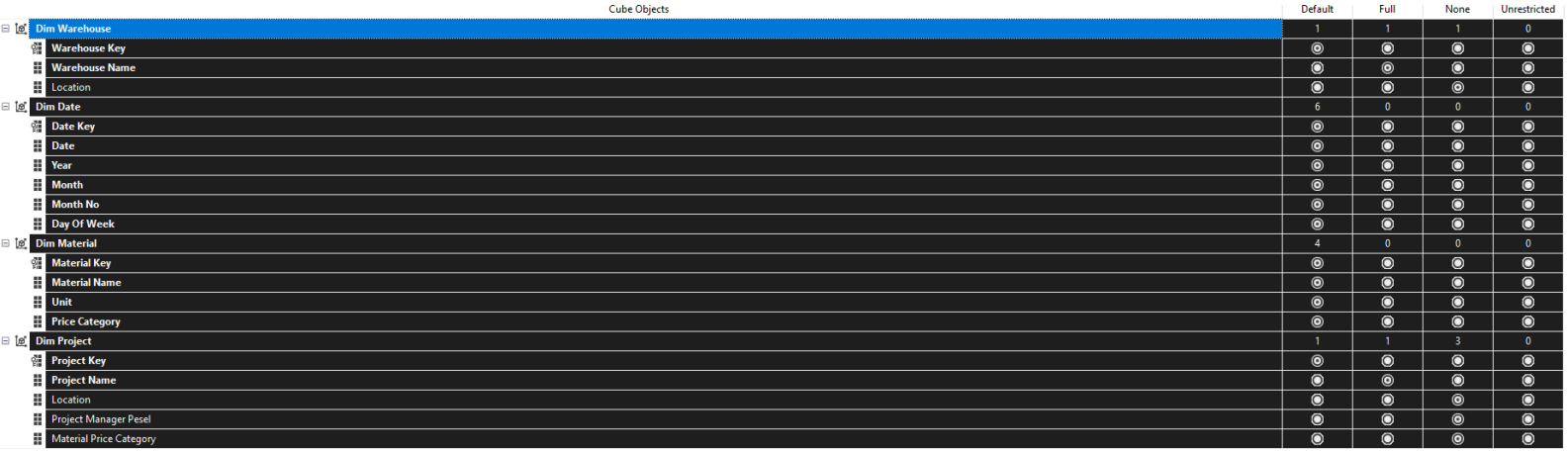
4. Cache and aggregation settings

Cache:

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

Aggregations were made on the monitoring table:

- Dim Warehouse: Warehouse Key, Warehouse Name, Location
- Dim Date: Date Key, Date, Year, Month, Month No, Day of Week
- Dim Material: Material Key, Material Name, Unit, Price Category
- Dim Project: Project Key, Project Name, Location, Project Manager Pesel, Material Price Category



5. **Discussion** (comparison of the theory with the obtained results)

Processing the cube

MOLAP		ROLAP		HOLAP	
Aggr.	No aggr.	Aggr.	No aggr.	Aggr.	No aggr.
4616,8	3559,5	6113,6	2354,8	6103	2423,3

For aggregated data, MOLAP performs the best, with significantly faster processing times compared to HOLAP and ROLAP, which show similar results. However, when processing non-aggregated data, ROLAP is the fastest, followed by HOLAP, while MOLAP is slower. These results confirm that MOLAP is most efficient when working with pre-aggregated data, while ROLAP excels with non-aggregated data, and HOLAP provides a balanced solution.

Query execution

MOLAP		ROLAP		HOLAP	
Aggr.	No aggr.	Aggr.	No aggr.	Aggr.	No aggr.
19,5	77,8	22,9	86,5	20,9	80,7
3,6	61,1	3,8	68,6	3,1	61,5
20	83	20,9	96,9	20,5	87,6

MOLAP is the fastest for aggregated data, outperforming both ROLAP and HOLAP. For non-aggregated data, HOLAP performs similarly to MOLAP, while ROLAP is the slowest in both cases, mainly due to its need to constantly access the database for data retrieval. This confirms that MOLAP is best for aggregated data, while HOLAP offers a good balance for both types of data.

Aggregations

At first, we noticed that queries with aggregations were faster than those without for all storage modes (MOLAP, ROLAP, and HOLAP). This is because aggregations pre-group the data during cube processing, so the system doesn't need to do it again when running the query. MOLAP had the best performance with aggregations because it stores all the pre-aggregated data in the cube. ROLAP and HOLAP also benefited from aggregations, but the difference in performance was smaller since they still rely on retrieving some data in real time. In general, more complex queries with many groups benefit the most from aggregations, while simpler queries show less improvement.

Conclusion

The tests showed how different storage methods (MOLAP, ROLAP, and HOLAP) perform and how aggregations affect them. The results matched expectations:

MOLAP: Takes more time to process data at first, but queries are faster because the data is pre-aggregated in the cube.
ROLAP: Processes data faster initially, but queries are slower because it needs to access the database in real-time.
HOLAP: A mix of MOLAP and ROLAP – faster queries with aggregations but slower without them.
Aggregations: Increase processing time but make queries faster by reducing the need to access data in real-time.