Khaula Molapo

Database Week 10

1. MDX Query Practice

MDX (Multidimensional Expressions) is a query language used for OLAP databases, mainly to analyze data in cubes. It helps to slice, dice, and drill into multidimensional data. Below are three sample MDX queries on a sales cube:

a. Slicing Query:

```
SELECT [Measures].[Sales Amount] ON COLUMNS,
[Date].[2025] ON ROWS
FROM [Sales]
WHERE ([Product].[Electronics])
```

This shows total sales for Electronics in 2025.

b. Drilling Query:

```
SELECT [Measures].[Sales Amount] ON COLUMNS,
[Region].[Country].[City].Members ON ROWS
FROM [Sales]
WHERE ([Product].[Furniture])
```

This drills down to see Furniture sales by city.

c. Aggregation Query:

```
SELECT [Measures].[Total Profit] ON COLUMNS,
[Time].[Year].Members ON ROWS
FROM [Sales]
```

This aggregates and shows yearly profit. MDX makes analyzing big datasets easy and fast by summarizing complex cube data.

2. Architecture Comparison

Below is a comparison between MOLAP, ROLAP, and HOLAP architectures:

Commercial Tools and Their Architectures:
- Microsoft Analysis Services (SSAS): Supports all, mainly MOLAP.
- Oracle OLAP: Primarily MOLAP, with some ROLAP support.
- IBM Cognos: Mainly ROLAP, can integrate HOLAP models.

MOLAP is best for speed, ROLAP for handling large data, and HOLAP balances both performance and scalability.

Feature	MOLAP	ROLAP	HOLAP
Data Storage	Multidimensional cube	Relational database	Both cube and relational
Query Speed	Very fast	Slower	Medium
Data Volume	Smaller datasets	Large datasets	Balanced
Storage Space	High	Low	Moderate
Real-Time Updates	Hard	Easy	Moderate
Example Use	Pre-calculated reports	Live transactional data	Mixed analysis

3. Performance Analysis

A real-world OLAP case is Microsoft's AdventureWorks Sales Cube used by many organizations to analyze sales performance. The company faced performance issues when users ran complex queries involving large historical data. Reports took minutes to load because the cube had millions of records across products, regions, and time dimensions.

To fix this, the team focused on three main areas: cube design, aggregation strategy, and hardware optimization. First, they redesigned the cube by splitting large dimensions into smaller ones. Instead of having one huge product dimension, they grouped products into categories and subcategories. This reduced query complexity and improved performance.

Second, they created pre-aggregated measures for common queries. For example, monthly and yearly summaries were calculated during cube processing rather than during runtime. This change reduced query response time from minutes to seconds.

Third, they optimized hardware by moving from traditional HDDs to SSD storage, improving read/write speeds. They also increased memory allocation for caching frequently accessed cube data.

To further improve speed, they implemented partitioning. Sales data was divided by year so only relevant partitions were scanned during analysis. This approach worked well for time-based queries.

Lastly, they fine-tuned indexes and processed cubes overnight to ensure daytime queries used pre-built aggregations.

After these improvements, the OLAP system achieved nearly 80% faster query performance and allowed real-time dashboards to run smoothly. Managers could now view sales trends, profit margins, and regional growth instantly.

This case shows that good cube design, smart aggregations, and optimized hardware are key to solving OLAP performance problems.