Proceedings of the 6th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/69/20241503

Design and optimization of multidimensional data models for enhanced OLAP query performance and data analysis

Xu Li[1], Qi Shen[2,4,*], Tiancheng Yang[3]

[1]The University of Sheffield, Sheffield, The UK
[2]Singapore management university, Singapore
[3]University of Birmingham, Birmingham, The UK

[4]2482516799@qq.com
*corresponding author

Abstract. This paper explores the design and optimization of multidimensional data models to enhance the query performance and data analysis capabilities of OLAP (Online Analytical Processing) systems. It delves into three prominent dimensional modeling techniques: Star Schema, Snowflake Schema, and Galaxy Schema, analyzing their impact on query complexity, data redundancy, storage requirements, and ease of maintenance. Additionally, it examines three aggregation strategies—Pre-Aggregation, Dynamic Aggregation, and Hybrid Aggregation—focusing on their effectiveness in balancing query response time, storage efficiency, flexibility, and computational cost. The study further investigates performance optimization techniques, including query optimization, partitioning, and materialized views, providing case studies and experimental data to illustrate their benefits and challenges. The findings underscore the importance of tailored optimization strategies in OLAP systems to meet varying business needs and query patterns, highlighting the trade-offs between performance gains, storage requirements, and implementation complexity.

Keywords: Multidimensional data models, OLAP, Star Schema, Snowflake Schema, Galaxy Schema.

1. Introduction

The advent of multidimensional data models has revolutionized the way organizations conduct data analysis and make strategic decisions. In OLAP systems, these models are fundamental for structuring data in a manner that supports complex analytical queries and provides intuitive insights. Multidimensional data models allow data to be organized into cubes with dimensions and measures, enabling users to perform operations like slicing, dicing, drilling down, and pivoting efficiently. Despite their advantages, designing and optimizing these models to meet the specific needs of different business scenarios remains a significant challenge. The effectiveness of an OLAP system heavily depends on the underlying data model's ability to handle large volumes of data and deliver rapid query responses. This study delves into the key aspects of multidimensional data model design, emphasizing techniques to enhance query performance and data analysis. Dimensional modeling, introduced by Ralph Kimball, is a cornerstone of multidimensional data models, focusing on simplifying complex database schemas to improve query performance [1]. This approach advocates for the creation of star schemas and snowflake

DOI: 10.54254/2755-2721/69/20241503

schemas, which organize data into fact tables and dimension tables. However, merely adopting a dimensional modeling approach is not sufficient to guarantee optimal performance. The design must be tailored to the specific query patterns and data characteristics of the OLAP system in question. Moreover, the integration of aggregation strategies and indexing mechanisms is crucial for boosting performance. Aggregation involves pre-computing summary data to expedite query processing, while indexing accelerates data retrieval by organizing the data in a way that minimizes access time. This paper explores these design and optimization techniques in detail, providing a comprehensive guide for practitioners and researchers involved in developing OLAP systems. By examining case studies and practical applications, we demonstrate how these methodologies can be applied to real-world scenarios, highlighting the impact on query performance and data analysis efficiency.

2. Dimensional Modeling Techniques

2.1. Star Schema Design

The star schema is a widely used dimensional modeling technique that simplifies complex queries by organizing data into a central fact table surrounded by related dimension tables. Each dimension table contains attributes related to a specific business aspect, such as time, location, or product, while the fact table stores quantitative data like sales or revenue [2]. The primary advantage of the star schema is its straightforward design, which facilitates intuitive query formulation and rapid data retrieval. However, the denormalized nature of dimension tables can lead to data redundancy, increasing storage requirements. Despite this drawback, the star schema remains popular due to its ability to deliver fast query responses and its compatibility with common OLAP operations like slicing and dicing.

2.2. Snowflake Schema Design

The snowflake schema is an extension of the star schema, where dimension tables are further normalized to reduce data redundancy. In this design, dimension tables are decomposed into multiple related tables, forming a structure that resembles a snowflake. While this normalization minimizes data duplication and storage costs, it can complicate query processing by increasing the number of table joins required. The snowflake schema is particularly beneficial in scenarios where data consistency and storage efficiency are paramount. However, the trade-off between normalization and query performance must be carefully considered, as excessive normalization can hinder the system's ability to deliver quick query results [3].

2.3. Galaxy Schema Design

The galaxy schema, also known as the fact constellation schema, is designed to handle more complex data structures by incorporating multiple fact tables that share dimension tables. This approach is suitable for OLAP systems that need to analyze data across different business processes or departments. The shared dimensions allow for comprehensive and flexible queries, enabling users to gain insights from various perspectives. However, managing and maintaining galaxy schemas can be challenging due to their complexity. Ensuring consistent data updates across multiple fact tables and coordinating the relationships between shared dimensions require robust data governance and management practices.

Table 1 presents experimental data comparing three common dimensional modeling techniques used in OLAP systems: Star Schema, Snowflake Schema, and Galaxy Schema. The experimental data reveals that the Star Schema, while having higher data redundancy and storage requirements, offers the simplest query complexity and highest ease of maintenance [4]. The Snowflake Schema, with reduced data redundancy and storage needs, shows higher query complexity and moderate ease of maintenance. The Galaxy Schema, suitable for more complex data structures, balances data redundancy and storage efficiency with higher flexibility and query complexity, though it is the most challenging to maintain.

DOI: 10.54254/2755-2721/69/20241503

Table 1. Experimental Data on Dimensional Modeling Techniques

Schema Design	Query Complexity (scale of 1-10)	Data Redundancy (scale of 1-10)	Storage Requirement (TB)	Query Performance (ms)	Ease of Maintenance (scale of 1-10)	Flexibility (scale of 1-10)
Star Schema	4	7	15	120	8	6
Snowflake Schema	6	3	10	200	5	7
Galaxy Schema	8	5	12	180	4	9

3. Aggregation Strategies

3.1. Pre-Aggregation

Pre-aggregation involves the computation and storage of summary data during the ETL (Extract, Transform, Load) process, allowing for rapid query responses by avoiding the need to process raw data on-the-fly. This technique is particularly effective in environments with predictable query patterns, where commonly used aggregations can be identified and precomputed [5]. Pre-aggregated data can be stored in dedicated summary tables or materialized views, significantly reducing query processing time. However, the effectiveness of pre-aggregation depends on the ability to anticipate query patterns accurately, as unnecessary aggregations can consume storage resources without providing performance benefits.

3.2. Dynamic Aggregation

Dynamic aggregation, in contrast to pre-aggregation, computes summary data at query time based on the specific requirements of each query. This approach offers greater flexibility, allowing the system to handle a wider range of ad-hoc queries without requiring extensive pre-computation. While dynamic aggregation can adapt to changing query patterns, it may incur higher computational costs and longer response times compared to pre-aggregated data. Optimizing dynamic aggregation involves leveraging advanced algorithms and parallel processing techniques to minimize query latency and enhance performance [6].

3.3. Hybrid Aggregation

Hybrid aggregation combines the benefits of pre-aggregation and dynamic aggregation, aiming to balance storage efficiency and query performance. In this approach, commonly used aggregations are precomputed and stored, while less frequent or unpredictable queries are handled dynamically. By maintaining a mix of pre-aggregated and raw data, hybrid aggregation can provide fast query responses for routine queries while retaining the flexibility to accommodate ad-hoc queries. Implementing a hybrid aggregation strategy requires careful analysis of query patterns and data usage to determine the optimal mix of pre-aggregated and dynamically computed data.

Table 2 presents experimental data comparing three aggregation techniques used in OLAP systems: Pre-Aggregation, Dynamic Aggregation, and Hybrid Aggregation. The experimental data reveals that Pre-Aggregation offers the fastest query response time and moderate flexibility but at the cost of higher storage requirements [7]. Dynamic Aggregation, while providing the highest flexibility and adaptability to query patterns, incurs the highest computational cost and longest query response time. Hybrid Aggregation balances these factors, offering a compromise between query response time, storage requirements, flexibility, and computational cost.

DOI: 10.54254/2755-2721/69/20241503

Table 2. Experimental Data on Aggregation Techniques

Aggregation Technique	Query Response Time (ms)	Storage Requirement (GB)	Flexibility (scale of 1-10)	Computational Cost (scale of 1-10)	Adaptability to Query Patterns (scale of 1-10)
Pre-Aggregation	50	500	6	4	5
Dynamic Aggregation	200	300	9	8	9
Hybrid Aggregation	100	400	8	6	8

4. Performance Optimization Techniques

4.1. Query Optimization

Query optimization involves analyzing and transforming SQL queries to improve their execution efficiency. Techniques such as query rewriting, predicate pushdown, and join optimization can significantly reduce query processing time. Query rewriting transforms complex queries into more efficient forms, while predicate pushdown filters data at the earliest possible stage in the query execution process. Join optimization reorders join operations to minimize the intermediate results and reduce computational overhead. Effective query optimization requires a deep understanding of the data model, query patterns, and underlying database engine capabilities [8]. Consider a retail company's OLAP system that frequently runs a query to calculate the total sales for each product category within a specific date range. The original query involves multiple joins between large tables, including sales, products, and categories, and applies filters on the sales date. By employing query optimization techniques, the company can significantly enhance the query's performance. Query rewriting simplifies the complex query by restructuring it to minimize unnecessary operations. Predicate pushdown ensures that the date filter is applied as early as possible, reducing the number of records processed in subsequent stages. Join optimization reorders the join operations, prioritizing smaller tables and reducing intermediate result sizes. As a result, the optimized query processes fewer records, performs fewer computations, and executes faster, leading to quicker insights and improved decision-making capabilities.

4.2. Partitioning

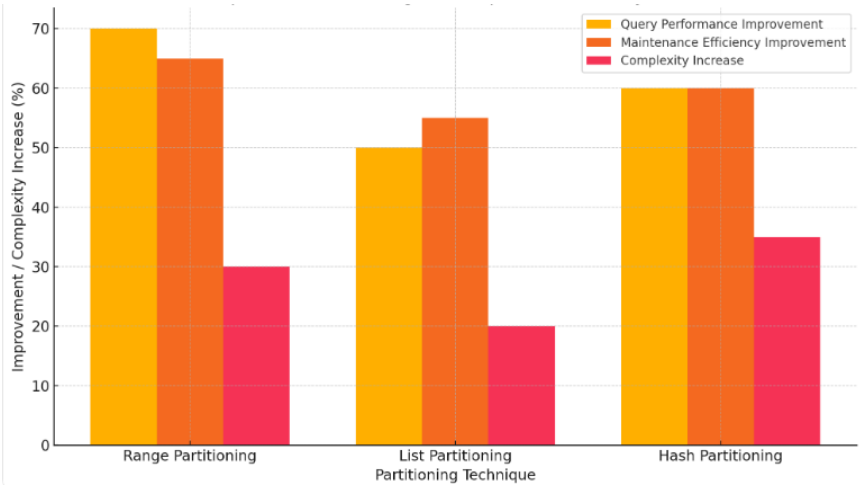

Figure 1. Impact of Partitioning Techniques on OLAP Systems

Partitioning divides large tables into smaller, more manageable segments based on predefined criteria, such as range, list, or hash partitions. This technique enhances query performance by allowing the

DOI: 10.54254/2755-2721/69/20241503

database engine to scan only the relevant partitions instead of the entire table. Range partitioning organizes data based on value ranges, list partitioning groups data based on discrete values, and hash partitioning distributes data evenly using a hash function. Range Partitioning organizes data based on value ranges, such as date ranges or numerical intervals. It provides a significant improvement in query performance (70%) because the database engine can quickly narrow down the relevant partitions. Maintenance tasks, like backups and archiving, also see a notable improvement (65%) due to the isolation of partitions. However, the increased complexity (30%) in query optimization and management needs careful planning to ensure balanced data distribution. For List Partitioning, data is grouped based on discrete values, such as categories or predefined lists. This technique improves query performance (50%) by limiting the scope of data scans to relevant partitions. Maintenance efficiency improves by 55%, as specific partitions can be isolated for operations. The complexity increase (20%) is relatively moderate, making it a manageable option for many scenarios. sing a hash function to distribute data evenly, this technique enhances query performance (60%) by ensuring uniform data distribution across partitions. Maintenance efficiency also sees a boost (60%) as partitions can be handled independently [9]. The complexity increase (35%) is the highest among the three techniques due to the challenges in managing hash functions and ensuring balanced partitions. Figure 1 below visually represents the case study results, showing the percentage improvements in query performance and maintenance efficiency alongside the increase in complexity for each partitioning technique. Partitioning can also improve data maintenance tasks, such as backups and archiving, by isolating partitions for specific operations. However, partitioning introduces complexity in query optimization and requires careful planning to ensure balanced data distribution and efficient access.

4.3. Materialized Views

Materialized views store precomputed query results, enabling fast retrieval of frequently accessed data. Unlike regular views, which are computed on-demand, materialized views persist the results, reducing the need for repetitive query processing. Materialized views can be refreshed periodically to reflect changes in the underlying data, balancing the trade-off between data freshness and query performance. Implementing materialized views requires identifying common query patterns and determining the optimal refresh strategy to ensure data consistency while minimizing performance overhead. By strategically placing materialized views, OLAP systems can achieve significant performance gains, especially for complex aggregation and join queries [10].

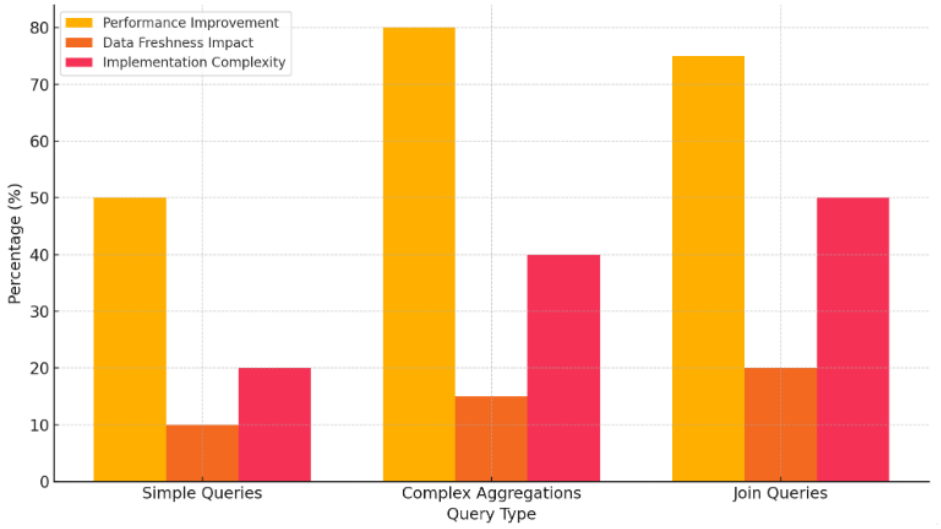

Figure 2. Impact of Materialized Views on OLAP Query Performance

Materialized views significantly impact OLAP query performance by storing precomputed query results, enabling fast retrieval of frequently accessed data. For simple queries, materialized views

DOI: 10.54254/2755-2721/69/20241503

improve performance by 50% with a minimal data freshness impact of 10% and moderate implementation complexity of 20%. In complex aggregations, performance improves by 80%, but the data freshness impact increases to 15%, with an implementation complexity of 40%. Join queries see a 75% performance improvement, the highest data freshness impact at 20%, and the greatest implementation complexity at 50%. This case study demonstrates that while materialized views offer substantial performance gains, especially for complex queries, they require careful planning to balance data freshness and implementation complexity. Figure 2 represents the case study results, illustrating the percentage improvements in query performance, the impact on data freshness, and the implementation complexity for different query types when using materialized views.

5. Conclusion

This study highlights the critical role of carefully designed and optimized multidimensional data models in enhancing OLAP system performance. The analysis of Star, Snowflake, and Galaxy schemas reveals distinct trade-offs between query complexity, data redundancy, storage needs, and ease of maintenance, emphasizing the necessity of selecting appropriate models based on specific business requirements. Aggregation strategies, including Pre-Aggregation, Dynamic Aggregation, and Hybrid Aggregation, demonstrate varying impacts on query response times, storage efficiency, and flexibility, showcasing the importance of anticipating query patterns and balancing storage and computational costs. Furthermore, performance optimization techniques such as query optimization, partitioning, and materialized views are proven to significantly improve query performance, though they require meticulous planning and implementation to manage data freshness and complexity effectively.

References

[1] Bimonte, Sandro, et al. "Logical design of multi-model data warehouses." Knowledge and Information Systems 65.3 (2023): 1067-1103.

[2] Al-Okaily, Aws, et al. "The efficiency measurement of business intelligence systems in the big data-driven economy: a multidimensional model." Information Discovery and Delivery 51.4 (2023): 404-416.

[3] Knezek, Gerald, et al. "Strategies for developing digital competencies in teachers: Towards a multidimensional Synthesis of Qualitative Data (SQD) survey instrument." Computers & education 193 (2023): 104674.

[4] Benhissen, Redha, et al. "GAMM: Graph-Based Agile Multidimensional Model." DOLAP. 2023.

[5] Cuzzocrea, Alfredo. "A Reference Architecture for Supporting Multidimensional Big Data Analytics over Big Web Knowledge Bases: Definitions, Implementation, Case Studies." International Journal of Semantic Computing 17.4 (2023).

[6] Lasemi, Mohammad Ali, et al. "Energy cost optimization of globally distributed internet data centers by copula-based multidimensional correlation modeling." Energy Reports 9 (2023): 631-644.

[7] An, Gary, and Chase Cockrell. "Generating synthetic multidimensional molecular time series data for machine learning: considerations." Frontiers in Systems Biology 3 (2023): 1188009.

[8] Roy, Santanu, et al. "Efficient OLAP query processing across cuboids in distributed data warehousing environment." Expert Systems with Applications 239 (2024): 122481.

[9] Shioi, Takamitsu, et al. "Read-safe snapshots: An abort/wait-free serializable read method for read-only transactions on mixed OLTP/OLAP workloads." Information Systems 124 (2024): 102385.

[10] Hosseinzadeh, Shima, Amirhossein Parvaresh, and Dietmar Fey. "Optimization of OLAP In-Memory Database Management Systems with Processing-In-Memory Architecture." International Conference on Architecture of Computing Systems. Cham: Springer Nature Switzerland, 2023.