International Journal of Scientific Engineering and Science
Volume 8, Issue 4, pp. 46-48, 2024. ISSN (Online): 2456-7361

Optimizing Database Performance in Large Projects

Mykyta Machekhin
Senior Software Engineer, Curbside Technologies Inc., US, New York City
machekhinn@gmail.com

Abstract— *With the growing volume of data and increasing complexity of information systems, database optimization is becoming a key factor in the success of large projects. High database performance directly impacts query processing speed, application load times, and ultimately user satisfaction. This article examines modern methods and practices that help maximize database efficiency in large projects. The article focuses on aspects such as data structuring and normalization, indexing, sharding and replication. Various approaches to query optimization and transaction management are considered. It also covers scaling and load balancing, as well as using caching to reduce load on the underlying database. Real-life examples are provided to demonstrate the application of these methods in large projects. Another important aspect is the selection and configuration of a suitable caching mechanism, which helps reduce the load on the database. Considering horizontal scaling and data replication also plays an important role in maintaining high availability and performance in large projects. The purpose of the work is to consider the possibilities of optimizing database performance in large projects. Scientific works served as the theoretical basis.*

Keywords— *IT, digitalization, modern technologies, databases, performance optimization.*

I. INTRODUCTION

Tuning the performance of SQL code in a live environment becomes relevant only after the main database development has been completed and logic errors have been eliminated. Users are generally satisfied with the functionality of the application, but strive to improve its performance.

Optimizing database performance includes the following steps:

1. Identify and analyze queries that are impacting performance.
2. Drawing up an optimal execution plan for identified queries.
3. Making changes to queries or database structure to achieve an optimal execution plan [1].

In this regard, the improvement of databases significantly reduces the response time to requests, providing instant access to data. This plays a key role in user experience and meeting business requirements. Secondly, optimization reduces the load on servers, facilitating more efficient use of computing resources. This opens up additional opportunities to scale the system and meet growing needs. In addition, database optimization has an impact on the overall performance of applications, increasing their responsiveness - an aspect that is extremely relevant in an environment of intense competition and high service standards.

An integral part of effective management is the pursuit of optimizing database performance to reduce operational costs and improve user experience. The speed of operation of the control system, its stability and reliability of information storage, as well as economic efficiency depend on optimization [2].

1. Key design strategies

For maximum efficiency in data management, ensure that stores, partitions, and indexes are in optimal condition for intended and actual use in the production environment. Optimizing data usage improves query performance, optimizes resources, and improves overall system efficiency.

- Data profiling: Explore the data, ensuring that the data model matches the requirements of your own workload. Consider data normalization, indexing strategies, and partitioning techniques. For optimal data access, select appropriate data types, define relationships between entities, and select optimal indexing strategies.

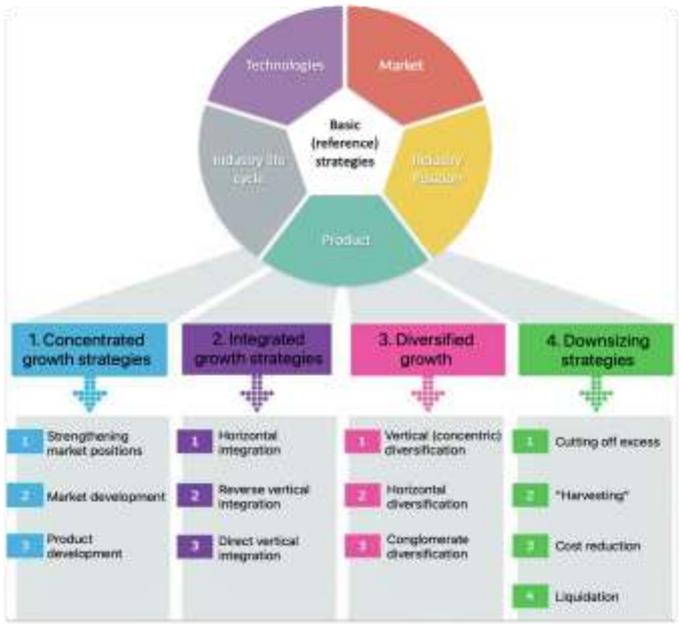
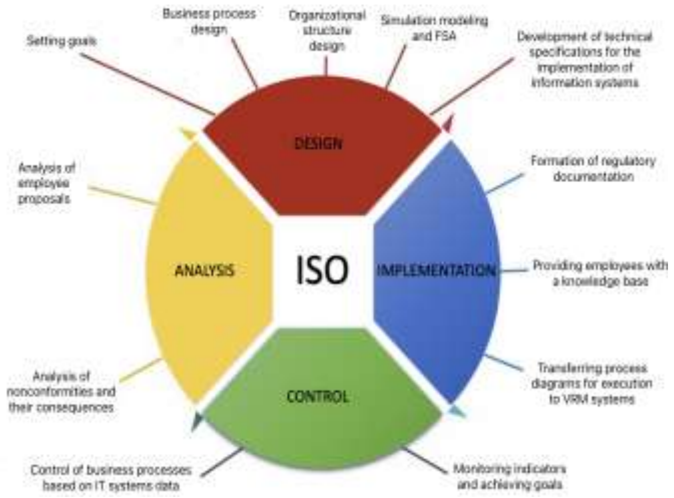

Technologies
Market
Industry life cycle
Basic (reference) strategies
Industry Position
Product
1. Concentrated growth strategies
2. Integrated growth strategies
3. Diversified growth
4. Downsizing strategies
1 Strengthening market positions
2 Market development
3 Product development
1 Horizontal integration
2 Reverse vertical integration
3 Direct vertical integration
1 Vertical (concentric) diversification
2 Horizontal diversification
3 Conglomerate diversification
1 Cutting off excess
2 "Harvesting"
3 Cost reduction
4 Liquidation

Fig.1. Design strategies

- Data Storage Configuration: Match your storage infrastructure to your production environment. Choose the right storage technology, such as relational databases, NoSQL databases, and data warehouses. Optimize storage parameters including compression, caching, and buffer size.
- Query Performance Optimization: Analyze and optimize queries running in production. Take advantage of techniques such as query optimization, indexing, and caching. Use query plans and performance monitoring tools to identify bottlenecks, then make the necessary improvements.

- Regular system monitoring and tuning: Continuously monitor production environment performance and iterate on data warehouse configuration and query optimization. Based on performance tuning recommendations, analyze system metrics, identify areas for improvement, and implement necessary changes.
- Data profiling: Data profiling involves analyzing data from a source and gathering insights about it. This allows you to understand the quality, structure and characteristics of the production environment data. Assessing data structure helps you understand how data is organized and how it is interconnected. It is also important to estimate the volume of data to determine storage requirements and identify scalability issues.
- Define data relationships: Analyze relationships between data elements, such as primary and foreign keys. Learn how data interacts to understand how changes to one table can affect related data.
- Assessing Data Quality: Assess data quality by examining completeness, accuracy, consistency, and uniqueness. Identify data anomalies, such as missing values or duplicate records, that can impact data integrity and query performance. This will help identify areas for data improvement.
- Collect distribution data: Analyze the distribution of values in each column to identify patterns in the data. This will help you select appropriate indexing strategies and query optimization techniques, taking into account the data distribution [3].
- Data Performance Monitoring: Continuous data performance monitoring is a practice aimed at monitoring the performance of data warehouses, partitions, and indexes in real time. This process involves collecting and analyzing key performance metrics associated with data operations. Effective monitoring allows potential bottlenecks to be identified and addressed, ensuring data efficiency.


Setting goals
Business process design
Organizational structure design
Simulation modeling and FSA
Development of technical specifications for the implementation of information systems
DESIGN
Analysis of employee proposals
Formation of regulatory documentation
ANALYSIS
ISO
IMPLEMENTATION
Providing employees with a knowledge base
Analysis of nonconformities and their consequences
CONTROL
Transferring process diagrams for execution to VRM systems
Control of business processes based on IT systems data
Monitoring indicators and achieving goals
Fig.2. Data monitoring strategies

Let's look at data performance monitoring strategies:

- Collecting Data Metrics: Monitoring key metrics that impact data performance. These measurements include data throughput, disk I/O related to data access, query response time, and loading times for certain data sections.
- Alert Configuration: Configure alerts specific to data metrics. Use predefined thresholds or anomaly detection to trigger alerts. These alerts alert you when metrics exceed acceptable limits or exhibit abnormal behavior.
- Diagnose data performance issues: Regularly review collected data metrics to identify potential bottlenecks or performance issues in data operations. Visualization tools or dashboards can be helpful in this process by identifying trends and anomalies in data performance.
- Data partitioning: Partitioning is the process of breaking up huge datasets or tasks into smaller, more manageable groups. This practice improves data performance by distributing the workload and optimizing parallel processing. Partitioning also enables efficient data access based on specific needs and query patterns. Partitioning can be implemented both vertically and horizontally (or segmentation) [4].

TABLE 1. Data partitioning

Strategy	Definition	Example	Use Cases
Vertical partitioning	Divide the table into smaller tables by selecting specific columns or fields for each section. Each partition represents a subset of the complete data.	If you have a table with columns A, B, C, and D, you can create one table with columns A and B and another table with columns C and D.	— The table contains many columns, but queries do not access all columns together. - Some columns are larger than others, and separating them can improve I/O performance. - Different pieces of data have different access patterns.
Horizontal partitioning	Dividing data into rows or ranges of values (also called sharding). Each section contains a subset of rows with similar characteristics.	If you have a table with rows 1 through 1000, you can create one partition with rows 1 through 500 and another partition with rows 501 through 1000.	— The data set is too large for one location or server. — Data is accessed based on certain ranges or filters. — to improve performance, it is necessary to distribute the workload between physical nodes or servers.

Let's look at a few strategies to optimize index performance:

• Examine query structure: Analyze database query patterns to identify frequently performed and potentially slowing operations. Understanding query structure can help you determine which indexes may be most useful for optimizing performance.

• Evaluate existing indexes: Review the current indexes in the database and evaluate their effectiveness, performance impact, and compliance with query patterns. Removing unused or redundant indexes can free up resources and improve performance.

• Selecting Columns for Indexing: Identify columns that are heavily used in WHERE, JOIN, and ORDER BY clauses in personal queries as potential candidates for indexing. This will provide quick access to data [5].

• Index Type Selection: Select the best index type according to the database system characteristics and query requirements. Consider balancing requirements for equality, ranges, exact match, and text search.

• Index column ordering: When creating composite indexes, consider column ordering to provide more efficient access to data. Placing the most frequently used columns at the beginning of the index will improve its efficiency.

• Balancing index size: Avoid creating indexes on columns with low selectivity as this can increase the size of the database. Index columns with high selectivity for optimal performance.

• Maintain index usage: Regularly monitor index usage and performance. Create new indexes or modify existing ones based on changes in queries. Remove or update indexes that are no longer useful.

• Testing and Validation: Before making changes to your production environment, conduct thorough testing to evaluate the performance impact of the changes using real-world workloads.

• Tradeoffs: Choose index types carefully, taking into account their storage costs and specific query considerations. Balance performance and storage requirements.

Effective index management helps speed up operations and reduce database load, resulting in improved overall system performance.

Recommendations for optimizing caching:

• Tune settings such as Time to Live (TTL) for optimal performance.

• Use in-memory caching for resource-intensive data that changes infrequently.

• Apply database query caching to avoid repeated execution of the same queries.

• Consider content delivery network caching, especially for static content, to improve delivery and reduce latency.

Introducing read cues:

• Use multiple read replicas to distribute load and improve overall system performance.

• Adjust the use of read replicas based on traffic, which can improve system responsiveness to requests.

Choosing the best strategies for caching and using read replicas will help you effectively manage the load on your data storage, ensuring fast access and high system performance.

When optimizing data consistency in a distributed workload, where data is distributed across multiple nodes or locations, it is important to balance the level of consistency with compute resources to maintain performance. Eventual consistency provides a trade-off between data accuracy and performance, where changes are propagated gradually to ensure consistent data across all nodes. This choice improves workload performance and availability.

To optimize data updates, efficient use of optimistic concurrency offers a solution for concurrent updates. A versioned or timestamp-based approach minimizes conflicts and allows multiple users or processes to work concurrently, improving performance and reducing resource latency.

In the context of optimizing data movement and processing, strategies such as extract, transform, and load (ETL) optimization, parallel processing, and batch processing play a key role in improving the efficiency of data operations. Using these methods reduces processing time and provides high throughput.

Optimizing the data warehouse design based on the proximity of data to users or services provides faster access and improves responsiveness. Designing for data proximity involves assessing access patterns and selecting solutions that support data movement and synchronization [6].

II. Conclusion

Thus, it can be noted that optimizing database performance is a key aspect of the successful implementation of large projects. All of these strategies represent a comprehensive approach to data optimization, balancing accuracy, performance, and availability, which are key tradeoffs in distributed systems.

Given the constant development of technology, it is also necessary to note the need to constantly update and improve database optimization methods in order to adapt to changing requirements and operating conditions. Optimized databases not only improve project efficiency, but also help improve the overall stability and reliability of information systems.

References

1. Performance optimization in Microsoft SQL server DBMS. [Electronic resource] Access mode: https://novainfo.ru/article/11469 .– (accessed 25.01.2024).
2. How to optimize work with databases: basic methods and recommendations. [Electronic resource] Access mode: https://eurobyte.ru/articles/kak-optimizirovat-rabotu-s-bazami-dannykh-osnovnye-metody-i-rekomendacii/. – (accessed 25.01.2024).
3. Recommendations for optimizing data performance . [Electronic resource] Access mode: https://learn.microsoft.com/ru-ru/azure/well-architected/performance-efficiency/optimize-data-performance. – (accessed 25.01.2024).
4. How to optimize work with databases: best practices and tips. [Electronic resource] Access mode: https://serverspace.ru/about/blog/kak-optimizirovat-rabotu-s-bd/?utm_source=yandex.ru&utm_medium=organic&utm_campaign=yandex.ru&utm_referrer=yandex.ru. – (accessed 25.01.2024).
5. A little bit about improving database performance: Practical tips. [Electronic resource] Access mode: https://habr.com/ru/companies/1cloud/articles/304642 /.– (accessed 01/25/2024).
6. Best Practices for Optimizing Database Performance: Tips and Techniques for Developers. [Electronic resource] Access mode: https://www.code-sample.com/2023/06/optimizing-database-performance-tips.html .– (accessed 25.01.2024).