ISSN 2278-3091
Volume 14, No.4, July - August 2025
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse021442025.pdf
https://doi.org/10.30534/ijatcse/2025/021442025

Fundamentals of working with Big Data in Databases

Eduard Shaikhulov
[1]Bachelor's Degree, Kazan innovative university named after V.G. Timiryasov, Russia, shaihulove@gmail.com

Received Date: June 20, 2025 Accepted Date: July 26, 2025 Published Date: August 06, 2025

ABSTRACT

This article examines the key aspects of working with big data in databases, including their characteristics, architectures, and processing technologies. It analyzes modern solutions such as distributed computing (Hadoop, Spark), NoSQL databases (MongoDB, Cassandra, Redis), and relational DBMS (SQL Server, Oracle, PostgreSQL) that are adapted for handling big data. The significance of the optimization methods, such as indexing, partitioning, and compression, is emphasized, particularly the role played by security controls, such as encryption, access control, and monitoring. The emphasis is on integration of the big data with analytical tools with a scalability and high performance orientation.

Key words : Big data, databases, database management systems (DBMS), distributed computing, NoSQL databases.

1. INTRODUCTION

The contemporary world has entered the age of digital transformation, with data volumes increasing at an accelerated rate. Big data is an enormous, unavoidable essential component of contemporary information systems defined by its size, rapid speed, and variety in formats. It has extended to multiple sectors: including healthcare, finance, scientific research, commerce, and production.

Traditional database management systems (DBMS), designed to handle relatively small volumes of structured information, face significant limitations when working with big data. There are many more in relation to related limitations with velocity and volume issues, reliability and availability, and handling multi-data format variability. The present interest covers methodologies and technologies specific to big data processing: distributed computing, NoSQL systems, and hybrid database architectures. The aim of this article is to explore the fundamental concepts of working with big data in databases. It examines technical characteristics, architectural solutions, technological tools, and optimization methods used to ensure the efficient management of large volumes of information.

2. TECHNICAL CHARACTERISTICS OF BIG DATA

Big data refers to a collection of information arrays characterized by enormous volume, high velocity of incoming data and variety of formats [1]. These three aspects define the requirements for DBMS, which must be adapted to process such information effectively.

The volume of data in today's systems is measured in terabytes, petabytes, and even exabytes because of the constantly increasing number of information sources such as Internet of Things (IoT) sensors, social media, e-commerce platforms, and other online spaces. To be able to handle this much, a system has to be highly scalable, either by vertically scaling-increasing one server's capabilities-or horizontally scaling-distributing the load among numerous servers.

The second critical characteristic is the velocity of data generation. A lot of data in modern information systems should be processed in real time or near-real time; this becomes crucially important in applications like financial transactions, logistics management, and industrial process monitoring.

Data variety is associated with the fact that information comes in different formats: structured, semi-structured, and unstructured. For instance, data can be represented in tables (relational databases), JSON or XML files (semi-structured data), as well as images, videos, or text (unstructured data).

Other important features of big data are veracity, associated with trust and quality of the information. In practice, big data could be incorrect, incomplete, and outdated because the sources feeding a central application vary, therefore including mechanisms of data cleaning, validation, and renewal is required.

That involves significant restrictions with regard to the architectures of database systems, which will surely affect: thus, they shall be distributed given a large volume, featuring high levels of fault tolerance for instant data access. Another no less important side of the story is supportability in scalability relevance to analytics with the integration into machine learning technologies; databases thus will be substantial members of recent big data ecosystems.

3. TECHNOLOGIES FOR BIG DATA

Big data necessitates the utilization of specialized technologies that can keep pace with such volume, velocity, and variety challenges while processing and managing it [2]. The main enabling technologies in this respect are distributed computing, NoSQL databases, and adapted relational DBMS. Distributed computing is the backbone of big data processing systems nowadays. Different platforms, such as Apache Hadoop and Apache Spark, enable the allocation of computational tasks across several nodes to manage large amounts of data effectively [3]. Hadoop utilizes the MapReduce framework, where a task is split into smaller subtasks that are executed simultaneously and subsequently merged to produce the final outcome (figure 1).

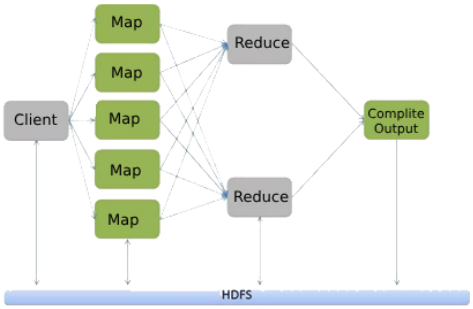

Figure 1: Hadoop workflow diagram

Equally, Hadoop contains the distributed file system HDFS (Hadoop Distributed File System) for reliable data storage characterized by high levels of fault tolerance. It is particularly effective in the long-term storage and analysis of huge amounts of data in a distributed environment.

Spark, on the other hand, offers in-memory data processing capabilities, significantly accelerating task execution, especially for real-time data analysis or iterative computations (figure 2).

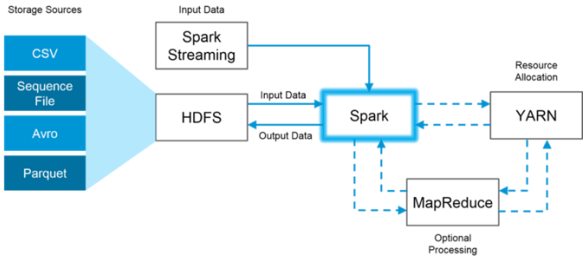

Figure 2: Spark data processing diagram

Apache Spark is a high-performance big data processing platform that works with CSV, Sequence File, Avro, and Parquet file formats while ensuring the loading and transformation are carried out for their future processing. This diagram provides an overview of how Spark is going to intake data in various formats, perform processing based on distributed computation mechanisms, and move the outcome to other systems or storage devices for further analysis. It gives Spark an architecture to work with huge volumes of data in memory with no latency, granting computational performance to increase in real time.

NoSQL databases are specifically designed to handle unstructured and semi-structured data, which are often unsuitable for traditional relational models [4]. MongoDB, Cassandra, and Redis are among the most commonly used NoSQL solutions. MongoDB is a document-oriented database that stores data in JSON-like documents, allowing for efficient handling of flexible structures (figure 3).

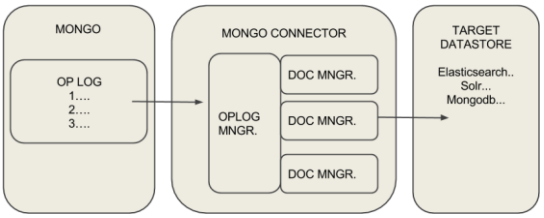

Figure 3: MongoDB architecture diagram

This architecture makes MongoDB particularly valuable in scenarios where data structures can be changed dynamically and horizontally scaled. MongoDB, in the diagram, enables integration with multiple data stores through Mongo Connector, which tracks changes in the operation log (OP LOG) and writes them to target systems such as Elasticsearch, Solr, or other MongoDB instances. This feature enables MongoDB to be effectively employed in distributed systems to ensure data synchronization, facilitate full-text search capabilities, and support analytics integration.

Cassandra, in contrast, is a column-based database featuring high scalability and a decentralized architecture. It provides high performance and low latency, making it ideal for handling substantial data quantities in worldwide distributed networks (figure 4).

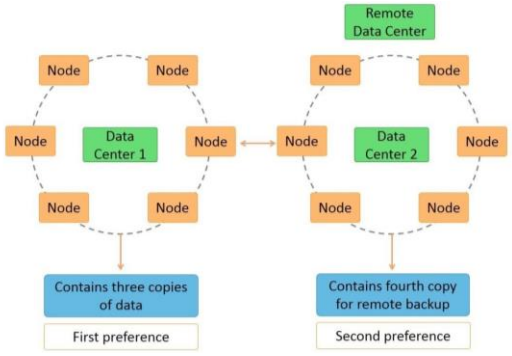

Figure 4: Cassandra architecture diagram

The architecture of Cassandra is based on a cluster model where data is stored across multiple nodes organized into data centers. This model ensures high fault tolerance as data is replicated across different nodes and geographically distributed data centers.

Redis is a key-value database known for its high operation speed and is used for caching, real-time data processing, and other tasks requiring minimal latency (figure 5).

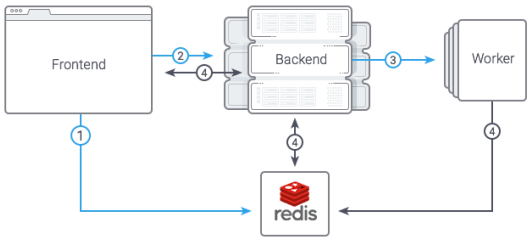

Figure 5: Redis architecture diagram

Redis will play nicely in all server application architectures, providing a frontend-backend-worker nodes caching layer to reduce the latency of the primary database, thus speeding up the data processing and system performance in general. Another distinctive feature of Redis is the in-memory approach to data processing; it enables the possibility of an immediate response in really highly loaded systems.

Relational DBMS, such as SQL Server, Oracle, and PostgreSQL, are also adapted to handle big data through the implementation of specialized technologies (table 1).

Table 1: Relational DBMS and their work with big data [5]

DBMS	Big data processing methods	Usage features
SQL Server	Integration with Hadoop and Spark; support for analytics and machine learning.	Used in corporate BI systems and analytics.
Oracle	Exadata – an optimized storage system for high-performance computing.	Designed for high-load financial and cloud solutions.
PostgreSQL	JSON support, parallel query execution, integration with external data sources.	Popular among open-source developers and analytical systems.

Each of these technologies has its specific tasks to perform; collectively, they make the ecosystem of big data. Their correct application provides the necessary efficiencies in processing data, extracting insights, and building analytic models applied in sectors such as finance, healthcare, manufacturing, and science.

4. OPTIMIZATION OF DATABASES FOR BIG DATA PROCESSING

As the amount of information handled by today's databases is growing significantly, efficient optimization methods must be found to increase their performance and minimize query processing delay times. The optimization techniques may range from indexing, partitioning, and data load management to compression and deduplication methods.

One of the most important optimization mechanisms is indexing, which accelerates query execution by structuring data in advance. In relational DBMS, indexing is based on B-trees, hash tables, or bitmap indexes that reduce the number of read operations during record searches [6]. In NoSQL databases like MongoDB, the creation of indexes on document fields significantly accelerates data retrieval. However, excessive indexing increases the cost of updating data, and therefore requires a balance between search speed and indexing overhead.

Another optimization technique is data partitioning, which includes the division of tables or collections into logical segments called partitions. This, in turn, enables parallel data processing, hence increasing query execution speed. As such, there exist a variety of partitioning techniques: horizontal, vertical, and by range. In the context of NoSQL databases, such as Cassandra, for instance, partitioning forms the basis for scalability, since this will ensure even distribution of load across nodes.

Handling big data also means efficient management of data loading. In real-time systems, stream processing is applied where new records are loaded upon arrival without overloading the primary database. Traditional relational DBMS often use batch processing, where data is written to the system in large blocks, reducing overhead costs. Asynchronous processing and intermediate result caching further contribute to reducing system response time.

Data compression, in this regard, reduces the volume of stored information considerably, reduces the actual need for storage, and improves the overall system throughput accordingly [7]. Various compression algorithms, such as LZ4, Snappy, and Zstandard, are applied in the storage and transportation of data with minimal loss of performance. For example, columnar compression is followed in systems dealing with textual and semi-structured data to reduce redundancy and speed up analytics.

Moreover, deduplication in highly informed databases helps in the elimination of duplicate records to save storage; this is applicable on both application and database levels, which again would find relevance with high data redundancy systems, logging, monitoring, and analytics platforms.

5. SECURITY AND DATA MANAGEMENT

With the unparalleled increase in volumes of data and their huge usage in nearly every other industry, the security and management of information become a highly crucial task for the developers and administrators of the database management system (table 2).

Table 2: Methods for ensuring security and managing big data [8]

Method	Description	Application
Authentication and authorization	Mechanisms for verifying user identity and restricting access rights.	Used in corporate and cloud databases.
Data encryption	Encoding information using cryptographic algorithms.	Applied for data protection in storage and transmission.
Access control	Restricting user rights to view, modify, and delete data.	Implemented through role-based access control (RBAC, ABAC).
Monitoring and auditing	Tracking user actions and detecting anomalies in system operations.	Used for identifying suspicious activity.
Backup and recovery	Creating copies of data for restoration in case of failure.	Applied in high-load and critical systems.
Data masking	Concealing parts of information to protect sensitive data.	Used in testing and analytical environments.

Big data security and its efficient management cannot be ensured without considering information protection at all levels, starting from user authentication to activity monitoring and data leak prevention. In this case, the consideration is not limited to the technological aspects of security; the regulatory requirements around personal data processing, such as GDPR (General Data Protection Regulation), CCPA (California Consumer Privacy Act), and HIPAA (Health Insurance Portability and Accountability Act), should also be taken into consideration. It is relevant that a data management system be adapted to the distributed storage architecture, ensuring fault tolerance with minimal losses in the case of any failure.

6. CONCLUSION

Big data and the related database technologies are instruments of the fundamental class of a modern digital world that ranges from real-time analyses to machine learning. Working with big data requires special architectures, such as implementations of distributed computing, NoSQL solutions, or adapted relational DBMS. Among the effective ways for database optimization, a distinction is made for indexing, partitioning, compressing, and managing data load. These methods ensure the high performance, scalability, and reliability of storage and processing systems and are a core in developing digital platforms. Security and data management also remain crucial issues.

The modern concept of database protection involves encryption, access control, monitoring, and auditing in order to minimize risks related to data disclosure and unauthorized access. Big data management is to be provided in conformity with legal requirements and to ensure the fault tolerance of systems. Future development will include enhancing data processing methodologies, integration of artificial intelligence tools, and further development of analytical and visualization capabilities. These solutions let organizations unlock the complete power of big data, thus making big data a prime asset in every sector.

REFERENCES

1. A. O. Adewusi, O. F. Asuzu. **Business intelligence in the era of big data: a review of analytical tools and competitive advantage**, *Computer Science & IT Research Journal,* Vol 5, no. 2, pp. 415-431, 2024.
2. A. Malikov. **Digital transformation and its impact on the structure and efficiency of modern business**, *Annali d'Italia*, no. 62, pp. 112-115, 2024.
3. L. G. Ahmad. **Spark, Hadoop, and Beyond: Exploring Distributed Frameworks for Metaheuristic Optimization in Big Data Analytics.** 2024.
4. V. H. Olivera, G. Ruizhe, R. C. Huacarpuma, P. B. Silva, A. M. Mariano, M. Holanda. **Data modeling and NoSQL databases-a systematic mapping review**, *ACM Computing Surveys (CSUR),* Vol. 54, no. 6, pp. 1-26, 2021.
5. B. El Idrissi, S. Baïna, A. Mamouny, M. Elmaallam. **RDF/OWL storage and management in relational database management systems: A comparative study**, *Journal of King Saud University-Computer and Information Sciences,* Vol. 34, no. 9, pp. 7604-7620, 2022.
6. M. M. Rahman, I. Siful, Md Kamruzzaman, H. J. Zihad. **Advanced query optimization in SQL databases for real-time big data analytics,** *Academic Journal on Business Administration, Innovation & Sustainability*, Vol. 4, no. 3, pp. 1-14, 2024.
7. L. Dinesh, K. G. Devi. **An efficient hybrid optimization of ETL process in data warehouse of cloud architecture**, *Journal of Cloud Computing*, Vol. 13, no. 1, pp. 122024.
8. A. Israfilov. **Stages of development and implementation of information security policies in organizations**, *Danish scientific journal*, no. 90, pp. 131-134, 2024.