UDC 004.652

OPTIMIZING STORAGE OF UNSTRUCTURED DATA USING NO-SQL DATABASES

Tulegenova Zh.
bachelor's degree, Kazakh-American University (Almaty, Kazakhstan)

ОПТИМИЗАЦИЯ ХРАНЕНИЯ НЕСТРУКТУРИРОВАННЫХ ДАННЫХ С ПОМОЩЬЮ NO-SQL БАЗ

Тулегенова Ж.С.
бакалавр, Казахско-Американский университет (Алматы, Казахстан)

Abstract

The article discusses the main approaches to storing unstructured data using NoSQL databases, including document-oriented databases (e.g., MongoDB) and key-value stores (e.g., Redis). Examples of practical applications of both technologies are provided, along with code snippets and explanations. The advantages of MongoDB, such as data structure flexibility and horizontal scaling, are highlighted, as well as the benefits of Redis, including high-speed access due to in-memory data storage and support for various data structures. Key differences between these databases and their suitability for different use cases are analyzed. The article emphasizes the importance of choosing the right database according to system requirements. Future research prospects involve integrating NoSQL solutions and creating hybrid architectures to enhance data storage and processing efficiency.

Keywords: NoSQL databases, MongoDB, Redis, data storage, data optimization.

Аннотация

В статье рассматриваются основные подходы к хранению неструктурированных данных с использованием NoSQL баз данных, включая документо-ориентированные базы данных (например, MongoDB) и базы данных типа «ключ-значение» (например, Redis). Описаны примеры применения обеих технологий с приведением кода и объяснением особенностей их использования. Показаны преимущества MongoDB, включая гибкость структуры данных и горизонтальное масштабирование, а также преимущества Redis, такие как высокая скорость доступа благодаря хранению данных в оперативной памяти и поддержка различных структур данных. Обсуждаются ключевые различия между этими базами данных и их применимость для различных сценариев. Статья акцентирует внимание на выборе подходящей базы данных в зависимости от потребностей системы. Перспективы дальнейших исследований связаны с интеграцией NoSQL решений и созданием гибридных архитектур для повышения эффективности хранения и обработки данных.

Ключевые слова: NoSQL базы данных, MongoDB, Redis, хранение данных, оптимизация данных.

Introduction

The modern increase in data volumes generated by various sources, including the Internet of Things (IoT), social networks, and industrial systems, calls for the development of effective solutions for data storage and management. Unlike traditional relational databases, which work well with structured data, processing and storing unstructured data require specialized technologies. One approach that enables efficient management of such data is the use of NoSQL databases, which offer flexibility and scalability when handling various formats and large data volumes. The aim of this

article is to explore optimal approaches for storing unstructured data using NoSQL technologies and to determine how they can contribute to improving the performance and reliability of systems.

NoSQL databases encompass a diverse range of technologies, including document-oriented, graph-based, and wide-column databases. These systems provide flexible data management and adaptation to changing business requirements. One of their key features is horizontal scaling, which makes them particularly useful for processing large volumes of data generated by modern applications and services.

Current research and practical use of NoSQL solutions demonstrate their ability to address challenges in handling unstructured data, such as text files, multimedia elements, and other formats. This article examines the main architectural approaches for optimizing data storage using NoSQL technologies, including their advantages and limitations compared to relational systems.

Main part

One popular approach to storing unstructured data is the use of document-oriented databases such as MongoDB. These databases store data in JSON-like documents, which allows for flexible data management [1]. The example below demonstrates the creation of a collection and the addition of a document to a database using Python.

```
from pymongo import MongoClient

# Connect to the local MongoDB database
client = MongoClient('mongodb://localhost:27017/')

# Create a database
db = client['example_database']

# Create a collection
collection = db['example_collection']

# Add a document to the collection
document = {
   "user_id": 1,
   "name": "Ivan Ivanov",
   "age": 29,
   "preferences": ["movies", "music", "sports"]
}

# Insert the document
collection.insert_one(document)

print("Document successfully added to the collection.")
```

In this code, the pymongo module is used to interact with the MongoDB database. The code includes connecting to the database, creating a new collection, and adding a document with unstructured information. The advantage of this approach is the ability to dynamically change the data structure without compromising system integrity, which is especially useful when dealing with diverse information [2].

The use of document-oriented databases allows for optimized storage by supporting hierarchical and semi-structured data. This approach facilitates the management of data with diverse structures and allows systems to adapt to changing business requirements without needing to modify the entire database [3].

Another example of an effective approach to storing unstructured data is the use of key-value databases like Redis. This database is known for its high performance and ease of use, making it suitable for storing frequently requested data or data requiring quick access. The example below demonstrates how to work with Redis using Python.

```
import redis

# Connect to the local Redis server
client = redis.StrictRedis(host='localhost', port=6379, db=0)

# Add data to Redis
client.set('user:1', '{"name": "Ivan Ivanov", "age": 29, "preferences": ["movies", "music", "sports"]}')

# Retrieve data from Redis
data = client.get('user:1')

# Decode the data
print("Retrieved data:", data.decode('utf-8'))
```

In this code, a connection is established with a local Redis server, and data is stored in JSON string format. The set method is used for data entry, where the key is user:1, and the value is a JSON string. The get method retrieves the data, which is then decoded for display.

Redis is ideal for caching, session management, and temporary data storage that requires quick access. Unlike document-oriented databases, Redis does not offer complex data structures but compensates for this with high operation speed and support for various structures like lists and sets [4].

Advantages and disadvantages of MongoDB

One of the main advantages of MongoDB is the flexibility of its data structure. This database allows for storing data in JSON-like documents, enabling the storage of unstructured information without a strict predefined schema. This approach simplifies data integration and management, especially when the structure requirements frequently change. Another key advantage of MongoDB is its support for horizontal scaling. The system enables sharding, which distributes data across multiple servers, allowing for efficient processing of large data volumes. MongoDB's high read and write performance, aided by indexing and optimization mechanisms, positively impacts data access speed [5].

However, MongoDB has its drawbacks. The absence of a strict data schema can lead to data inconsistencies, particularly in systems with highly complex relationships. This can complicate data management and analysis. Additionally, despite high performance with scaling, MongoDB may require significant resources to maintain performance when handling very large data volumes and complex queries. Effective use of MongoDB capabilities requires specific knowledge and skills for configuration and administration, which may involve additional time and training [6].

Comparison of MongoDB and Redis

MongoDB and Redis are two popular NoSQL databases, each with its strengths and weaknesses, making them suitable for different use cases. MongoDB is a document-oriented database that supports storing data in JSON-like documents. The main advantage of MongoDB is its flexible data structure, which allows unstructured information to be stored and managed without a strict schema [7]. This makes MongoDB an excellent choice for applications where data changes frequently or has a complex hierarchical structure. It also supports horizontal scaling and ensures high performance for read and write operations with the use of indexes.

Redis, on the other hand, is a key-value database known for its extremely high performance due to storing data in memory. This makes Redis ideal for tasks requiring ultra-fast data access, such as caching, session management, message queues, and temporary data. It supports various data structures, including strings, lists, sets, and hashes, which expand its data storage capabilities. However, Redis is less convenient for working with complex hierarchical information compared to MongoDB [8].

The main difference between MongoDB and Redis lies in their approaches to data storage. MongoDB handles larger data volumes better, where data must be persistently stored on disk, while

Redis, due to its in-memory architecture, is faster and more efficient in processing operations but requires additional mechanisms for long-term data storage. Additionally, MongoDB can process more complex queries thanks to its built-in indexing and advanced querying capabilities, whereas Redis is best suited for simple operations with instant response.

Redis has several advantages over MongoDB, particularly when it comes to tasks involving high performance and fast data access. First, the primary advantage of Redis is that it stores data in memory, which ensures extremely fast data access compared to systems using disk storage, including MongoDB [9]. This makes Redis an ideal choice for scenarios where minimal latency is crucial, such as data caching, session management, and message queue processing.

Another significant advantage of Redis is its support for various data types, including strings, lists, sets, and hashes, which allows for efficient handling of different use cases with minimal latency. Due to its architecture, Redis also supports atomic operations, making it suitable for applications where data integrity must be guaranteed during complex operations.

Redis is also known for its simplicity in installation and use. It has an intuitive command-line interface and relatively simple architecture, which facilitates deployment and management. Additionally, Redis can be used as a cache with an automatic data expiration feature (TTL), enabling convenient management of temporary data and ensuring memory is freed as needed [10, 11].

Thus, if the main priority is fast data access and support for simple data structures, Redis offers significant advantages over MongoDB. It is better suited for tasks that require immediate response and where data may not need long-term storage, while MongoDB is more applicable for long-term storage and managing complex data structures.

Conclusion

Optimizing unstructured data storage using NoSQL databases provides new opportunities for enhancing system performance and flexibility. The examples of MongoDB and Redis illustrate how choosing the right database depends on the tasks and data processing speed requirements. MongoDB is suitable for storing data with a changing structure and processing large volumes of information requiring long-term storage. In contrast, Redis, due to its in-memory data storage, ensures instant access, making it ideal for caching and temporary data storage.

The main differences between these systems lie in their data processing approaches: MongoDB offers storage and management of complex hierarchical information, while Redis focuses on high-speed operations and support for simple data structures. This allows system developers and architects to choose the most appropriate solutions based on the specific needs of their applications.

Future research in this field may focus on integrating various NoSQL databases to create hybrid solutions that combine the strengths of different systems. This will enable even greater efficiency and better system adaptation to growing data volumes and processing requirements.

References

1. Ilyin I.V., Ilyashenko V.M. Development of IT architecture in medical organizations based on the introduction of big data technologies // Technological Perspective within the Eurasian Space: New Markets and Points of Economic Growth. 2018. P. 376-382.
2. Kuznetsov S.D., Poskonin A.V. Distributed horizontally scalable solutions for data management // Proceedings of the Institute for System Programming of the RAS. 2013. Vol. 24. P. 327-358.
3. Klemenkov P.A., Kuznetsov S.D. Big data: modern approaches to storage and processing // Proceedings of the Institute for System Programming of the RAS. 2012. Vol. 23. P. 143-158.
4. Burlov V.G., Gryzunov V.V., Sipovich D.E. Adaptive availability management in a geo-information system using fog computing // International Journal of Open Information Technologies. 2021. Vol. 9. No.9. P. 74-87.
5. Seghier N.B., Kazar O. Performance benchmarking and comparison of NoSQL databases: Redis vs MongoDB vs Cassandra using YCSB tool // 2021 International Conference on Recent Advances in Mathematics and Informatics (ICRAMI). IEEE, 2021. P. 1-6.

6. Matallah H., Belalem G., Bouamrane K. Evaluation of NoSQL databases: MongoDB, Cassandra, HBase, Redis, Couchbase, OrientDB // International Journal of Software Science and Computational Intelligence (IJSSCI). 2020. Vol. 12. No.4. P. 71-91.
7. Punia Y., Aggarwal R. Implementing information system using MongoDB and Redis // International Journal of Advanced Trends in Computer Science and Engineering. 2014. Vol. 3. No.2. P. 16-20.
8. Abu Kausar M., Nasar M., Soosaimanickam A. A study of performance and comparison of NoSQL databases: MongoDB, Cassandra, and Redis using YCSB // Indian Journal of Science and Technology. 2022. Vol. 15. No.31. P. 1532-1540.
9. Sánchez R.A.G., Bernal D.J.M., Parada H.D.J. Security assessment of NoSQL MongoDB, Redis, and Cassandra database managers // 2021 Congreso Internacional de Innovación y Tendencias en Ingeniería (CONIITI). IEEE, 2021. P. 1-7.
10. Sen P.S., Mukherjee N. An ontology-based approach to designing a NoSQL database for semi-structured and unstructured health data // Cluster Computing. 2024. Vol. 27. No.1. P. 959-976.
11. Shantharajah S.P., Maruthavani E. A survey on challenges in transforming No-SQL data to SQL data and storing in cloud storage based on user requirement // International Journal of Performability Engineering. 2021. Vol. 17. No.8. P. 703.