NoSQL Databases

FULL NAME: YUPENG WEN

STUDENT ID: s224212855

TEACHERS: Iynkaran Natgunanathan

DATE OF COMPLETION: 08/09/2024

WORD COUNT: 2614

Abstract

Relational databases have long been the dominant technology for managing structured data, primarily due to their reliability, consistency, and efficiency in transactional operations. They are based on the principles of the relational model developed by Edgar F. Codd in 1970, which revolutionized the way data was stored, accessed, and managed. The model organizes data into tables (or relations) with predefined schemas, ensuring that relationships between different data points are clearly defined and that the data remains consistent and accurate. This approach quickly became indispensable in industries where data accuracy and consistency are paramount, such as banking, finance, and large enterprise systems. In these environments, the ability to maintain strict control over data integrity, enforce complex relationships, and execute transactions reliably made relational databases the go-to choice.

However, the rise of big data, cloud computing, and the need for scalable distributed systems exposed limitations in relational databases. Chen et al. (2018) found that processing times on blockchain networks were significantly slower than MySQL, and relational databases struggle with unstructured or semi-structured data. Moreover, their vertical scaling architecture requires more powerful hardware, making them less efficient for large-scale, real-time data processing.

In response, NoSQL databases emerged as a viable alternative, addressing the challenges faced by traditional systems. With flexible data models, horizontal scalability, and the ability to handle diverse data types, NoSQL databases cater to the needs of modern applications. This essay will explore the limitations of relational databases, introduce NoSQL, and discuss how NoSQL addresses these issues, along with its pros and cons.

Introduction

NoSQL databases are non-relational databases designed to handle large volumes of structured, semi-structured, or unstructured data. They offer

flexibility in data storage and retrieval, making them well-suited for scenarios where traditional relational databases (like MySQL or PostgreSQL) might struggle. NoSQL databases are particularly useful for big data, real-time web applications, and distributed systems.

There are four primary types of NoSQL databases, each suited to different types of data and use cases. Document stores, such as MongoDB and CouchDB, store data in the form of documents, typically in formats like JSON or BSON. This allows for flexible, nested data structures that are easy to modify and scale. Key-value stores, such as Redis and Amazon DynamoDB, use a simple key-value pair model, excelling in performance and scalability for applications that require fast read and write operations. Column-family stores, like Apache Cassandra and HBase, organize data into columns rather than rows, optimizing performance for large-scale distributed systems and real-time analytics. Lastly, graph databases, such as Neo4j, store data in a graph structure, making them ideal for applications that involve complex relationships between data points, such as social networks or recommendation engines.

NoSQL databases' types

Document stores

Document stores, like MongoDB and CouchDB, are a type of NoSQL database that stores data as individual documents, typically in formats like JSON or BSON, providing a flexible, schema-less structure. Unlike relational databases that require predefined schemas, document stores allow for the storage of complex, nested data, making them ideal for applications where data evolves or is inherently unstructured. Each document in a document store is a self-contained unit of data that can vary in structure from other documents, offering significant flexibility. This makes document stores especially useful for applications such as content management systems, catalogues, and user profiles. Additionally, document stores offer easy scaling and high performance, particularly for read- or write-heavy workloads.

MongoDB, for instance, handles large data volumes and distributes documents across multiple servers, ensuring scalability and availability in distributed environments, making it a popular choice for modern applications.

Key-Value Stores

Key-value stores, a type of NoSQL database which stores data using a simple key-value pair model, where each piece of data is associated with a unique key, allowing for fast lookups and retrieval. This simplicity makes key-value stores highly performant and scalable, particularly for applications requiring rapid read and write operations. Without the need for complex data relationships or joins, key-value stores excel in use cases like caching, session management, and real-time analytics, where speed and efficiency are crucial. Databases like Redis and Amazon DynamoDB are popular key-value stores, capable of handling large volumes of data and transactions at high speed, while distributing data across multiple nodes for scalability. Their design supports horizontal scaling, allowing them to expand easily by adding servers, making them ideal for environments that prioritize fast data access with relatively simple data structures.

Column-Family Stores

Column-family stores, a type of NoSQL database, organize data by columns rather than rows, optimizing performance for large-scale distributed systems and real-time analytics. Unlike relational databases that store data in rows, column-family stores group data by columns, allowing for more efficient querying and retrieval, especially when only specific columns are needed (Counte & Cuza, 2018). This structure is ideal for applications handling large datasets, such as log analysis, recommendation engines, and data warehousing, as it enables quick retrieval of specific attributes without scanning entire rows, significantly improving performance. According to Sheth (2023), comparative performance analysis shows that column-family databases like Cassandra and HBase outperform traditional relational databases for large-scale data workloads. These databases efficiently distribute data across multiple servers, ensuring high availability and fault

tolerance, making them ideal for systems processing real-time data streams or large datasets. Additionally, the columnar format supports data compression, further improving performance and storage efficiency.

Graph Databases

Graph databases are a type of NoSQL database designed to store and manage data as nodes (entities) and edges (relationships), making them ideal for handling complex, interconnected data. Unlike relational databases, which rely on tables and predefined relationships, graph databases use dynamic graph structures to represent and query relationships, enabling efficient processing of data with intricate interconnections (Pokorný, 2015). In a graph database, nodes represent entities such as people, products, or places, while edges represent relationships between these entities, with both holding properties to provide detailed information. This model allows for the quick and intuitive representation of real-world relationships, significantly improving the performance of relationship-based queries, which are often inefficient in relational databases. Graph databases excel in applications like social networks, recommendation engines, fraud detection, and supply chain management, where relationships between entities are central. For example, they can efficiently answer queries like "Who are the mutual friends of two users?" or "Which products are often bought together?" thanks to their optimized traversal capabilities. Popular graph databases like Neo4j, Amazon Neptune, and ArangoDB are essential in industries that rely on complex data networks, including knowledge management and fraud detection, where relationships between entities such as financial transactions can be analysed to detect suspicious patterns. Moreover, graph databases offer flexibility in data modelling, allowing new nodes and relationships to be added without schema changes. This adaptability makes them a powerful tool for systems that evolve over time. Overall, graph databases excel in managing and querying complex relationships, providing an efficient and intuitive way to explore connections in applications where data interconnections are critical.

Limitation of Relational Databases

Struggle with unstructured data

Relational databases, while long considered the standard for managing structured data, face growing limitations in the modern data landscape. Built around a rigid schema, relational databases store data in rows and columns, which makes them highly effective for structured data with clearly defined relationships. However, according to Nishtha et.al(2012), relational databases make use of SQL, which is featured to work on structured data, but SQL can be highly complex when working with unstructured data. This same structure becomes a significant drawback when dealing with unstructured or semi-structured data, such as images, videos, or social media interactions.

Poor scalability

Relational databases, built on the traditional RDBMS model, are well-suited for structured data but struggle when dealing with large-scale, unstructured datasets. Base on the research done by WONDWESSEN HAILE ADDAL, scaling relational databases has been done vertically, by adding more resources like CPU and memory to a single machine. However, this approach has its limits and becomes costly as data demands grow. To achieve scalability, relational databases often require redundant copies of data, which introduces challenges in maintaining consistency and managing the system (Kolonko Kamil 2018). Horizontal scaling, which involves adding more nodes to the system, is even more complex in relational databases, as it can create difficulties in handling transaction processing and maintaining data consistency across distributed nodes. These limitations make it harder for traditional relational databases to adapt to modern, large-scale, and highly dynamic data environments.

Poor Performance

Relational databases often struggle with performance when compared to NoSQL databases, particularly in handling large-scale, unstructured, or highly dynamic data. The rigid schema and high level of abstraction in relational databases can slow down query processing, especially for complex queries or operations requiring frequent schema changes. According to the

study of Kotiranta et.al in 2022, the performance of MySQL 5.1.41 was too poor for complex queries. Additionally, relational databases face challenges when distributed across multiple nodes, often leading to operational inefficiencies and performance bottlenecks in cloud environments (Litchfield et.al 2017).

Advantages of NoSQL Databases

NoSQL databases effectively address the limitations of traditional relational databases by offering a more flexible and scalable solution for modern data needs. Unlike relational databases, which rely on rigid schemas, NoSQL databases handle unstructured and semi-structured data without predefined structures, making them ideal for managing diverse data types such as images, videos, and social media content. Additionally, NoSQL supports horizontal scaling, allowing data to be distributed across multiple nodes, which enhances scalability and reduces the costs associated with vertical scaling in relational systems. By avoiding rigid schemas, NoSQL also improves query processing and performance, particularly for large and dynamic datasets. Its cloud-friendly architecture makes NoSQL databases well-suited for distributed environments, enabling efficient data management at scale while minimizing performance bottlenecks. Furthermore, NoSQL is optimized for dynamic, evolving data structures, offering flexibility that relational databases struggle to provide in rapidly changing data landscapes.

Flexible Schema: NoSQL databases handle unstructured and semi-structured data effectively, without the need for rigid schemas.

Unstructured Data Support: NoSQL can store and manage various data types such as images, videos, and social media interactions.

Horizontal Scaling: NoSQL databases scale horizontally, distributing data across multiple nodes, allowing for better scalability compared to relational databases' vertical scaling.

Performance Improvement: NoSQL avoids rigid schemas, enabling faster query processing and better performance, particularly for large, dynamic data.

Cloud-Friendly: NoSQL databases work well in distributed cloud environments, handling large datasets efficiently while avoiding performance bottlenecks.

Dynamic Data Handling: NoSQL is optimized for highly dynamic, evolving data structures, unlike relational databases, which struggle with frequent schema changes.

Feature	Relational Databases	NoSQL Databases
Data Model	Table-based with rows and columns	Flexible models: key-value, document, column-family or graph
Schema	Rigid, predefined schema (needs to be defined upfront)	Flexible, dynamic schema (can change as needed)
Scalability	Vertical (add more resources to one server)	Horizontal (add more servers to distribute load)
Data Integrity	Strong consistency (ACID: Atomicity, Consistency, Isolation, Durability)	Eventual consistency or tunable (CAP theorem)
Query Language	SQL (Structured Query Language)	Varies (JSON, Key-Value lookups, etc.)
Performance	Optimized for complex joins and transactions	Optimized for high-volume reads/writes
Best Use Cases	Financial systems, ERP, CRM, apps with complex relationships	Real-time apps, distributed systems, big data

Disadvantages of NoSQL Databases

NoSQL databases have drawbacks, including limited support for full ACID transactions, as they often prioritize availability and partition tolerance over consistency. They also offer fewer standardized query capabilities compared

to SQL, requiring custom solutions for complex queries. Data duplication is another issue, as NoSQL favours performance over strict normalization, leading to increased storage needs and potential consistency problems during updates. Additionally, NoSQL databases are relatively new, lacking the maturity, tooling, and community support of relational databases, which can be a challenge for complex analytical tasks. Understanding these limitations is key to selecting the appropriate database for specific use cases.

No transaction support

Many non-relational databases implement BASE (Basically Available, Soft state, eventually consistent) properties, prioritizing performance over the traditional ACID guarantees. As a result, this often leads to a compromise on data consistency, allowing eventual consistency rather than immediate consistency across the database.

By not strictly adhering to ACID properties, non-relational databases offer a lower level of reliability compared to relational databases. To ensure ACID constraints, developers must implement custom programming solutions, which are inherently available in relational databases. (Nishtha et.al 2012).

In 2015, Ramesh Dharavath and Chiranjeev Kumar propose a scalable three-tier architecture and a distributed middleware protocol to support atomic transactions across heterogeneous NoSQL databases. The proposed framework is entirely distributed, making it highly scalable, available, flexible, and cost-effective. It addresses the lack of standard and interoperability issues in the current NoSQL database landscape, enabling complex applications to benefit from the use of NoSQL systems. The framework preserves atomicity and isolation of transactions in a heterogeneous distributed column-oriented database environment.

Data Duplication

According to the study of NICOLETA C. BRAD in 2012, NoSQL databases typically use a simple data model with flexible control over data layout, often resulting in highly de-normalized structures and excessive duplicate data.

This duplication extends to backups, significantly increasing storage requirements. The flexibility of the key-value model, where values can be schema-less and complex, further contributes to the generation of duplicate data.

In this research, NOCOLETA also proposes a data de-duplication approach for NoSQL databases, called DDNSDB, that operates at the database level rather than the file or sub-file level. DDNSDB leverages the structural information (metadata) of NoSQL data models to identify and remove duplicate data, leading to significant storage savings. The main goals are to maximize the reduction of duplicates in key-value NoSQL databases, improve the performance of the de-duplication process, and design the solution to be scalable in a cloud environment.

Imperfect Technology Ecosystem

The technology ecosystem surrounding NoSQL databases is still evolving and lacks the maturity and robustness of traditional relational databases. Unlike SQL, which benefits from decades of standardization, well-established tools, and a large community of support, NoSQL databases often have fragmented ecosystems with fewer standardized frameworks. This can lead to challenges in finding comprehensive tools for database management, monitoring, and querying. Moreover, the relative novelty of many NoSQL systems means that they may not offer the same level of documentation, community resources, or integration capabilities as relational databases. This incomplete ecosystem can be a limiting factor for developers, particularly for complex applications requiring advanced analytics or enterprise-grade stability and support.

However, the future of the NoSQL technology ecosystem is promising, as the demand for scalable, flexible, and high-performance databases continues to grow. With increasing adoption in various industries, NoSQL databases are likely to see further advancements in tooling, standardization, and community support. As more companies and developers contribute to open-source NoSQL projects, improvements in database management, querying

capabilities, and performance optimization will emerge. Additionally, the integration of NoSQL with cloud platforms and modern development frameworks is expected to enhance ease of use and scalability, making it more accessible for a wider range of applications. Over time, this evolving ecosystem will close the gap with relational databases, providing developers with a richer set of tools and resources for building and maintaining robust, large-scale systems.

conclusion

In conclusion, while relational databases have long dominated the field of structured data management due to their reliability and well-defined schemas, they face increasing challenges in today's data landscape. The emergence of NoSQL databases offers a flexible, scalable, and high-performance alternative for handling unstructured and semi-structured data, particularly in applications requiring real-time processing and distributed systems. With various types, including document stores, key-value stores, column-family stores, and graph databases, NoSQL provides tailored solutions for different data needs. However, despite their advantages in scalability and adaptability, NoSQL systems come with trade-offs, such as limited ACID transaction support and potential data redundancy. Therefore, the choice between relational and NoSQL databases should be based on specific application requirements, balancing the need for flexibility, performance, and consistency.

Reference List

ADDAL, W.H., 2019. A COMPARATIVE ANALYSIS OF RELATIONAL AND NON-RELATIONAL DATABASES FOR WEB APPLICATION (Doctoral dissertation, NEAR EAST UNIVERSITY).

Azad, P., Navimipour, N.J., Rahmani, A.M. and Sharifi, A., 2020. The role of structured and unstructured data managing mechanisms in the Internet of things. Cluster computing, 23, pp.1185-1198.

Batra, S. and Tyagi, C., 2012. Comparative analysis of relational and graph databases. International Journal of Soft Computing and Engineering (IJSCE), 2(2), pp.509-512.

Codd, E.F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), pp.377–387. doi:https://doi.org/10.1145/362384.362685.

Conte, E. and Cuza, C.M., 2018. Column-based Databases and HBase.
Sheth, V., 2023. Comparative Performance Analysis of Column Family Databases:

Cassandra and HBase (Doctoral dissertation, Dhirubhai Ambani Institute of Information and Communication Technology).

Chen, S., Zhang, J., Shi, R., Yan, J. and Ke, Q. (2018). A Comparative Testing on Performance of Blockchain and Relational Database: Foundation for Applying Smart Technology into Current Business Systems. Distributed, Ambient and Pervasive Interactions: Understanding Humans, pp.21–34. doi:https://doi.org/10.1007/978-3-319-91125-0_2.

Dharavath, R. and Kumar, C., 2015. A scalable generic transaction model scenario for distributed NoSQL databases. *Journal of Systems and Software*, *101*, pp.43-58.

Imam, A.A., Basri, S., Ahmad, R., Watada, J. and González-Aparicio, M.T. (2018). Automatic schema suggestion model for NoSQL document-stores databases. Journal of Big Data, 5(1). doi:https://doi.org/10.1186/s40537-018-0156-1.

Jatana, N., Puri, S., Ahuja, M., Kathuria, I. and Gosain, D., 2012. A survey and comparison of relational and non-relational database. International Journal of Engineering Research & Technology, 1(6), pp.1-5.

Kotiranta, P., Junkkari, M. and Nummenmaa, J., 2022. Performance of graph and relational databases in complex queries. *Applied sciences*, *12*(13), p.6490.

Kolonko, K., 2018. Performance comparison of the most popular relational and non-relational database management systems.

Lim, H., Fan, B., Andersen, D.G. and Kaminsky, M., 2011, October. SILT: A memory-efficient, high-performance key-value store. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (pp. 1-13).

Litchfield, A., Althwab, A. and Sharma, C., 2017. Distributed relational database performance in cloud computing: An investigative study. AIS.

Mishra, S. and Misra, A., 2017, September. Structured and unstructured big data analytics. In 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC) (pp. 740-746). IEEE.

Brad, N., 2012. Data De-Duplication in NoSQL Databases (Doctoral dissertation, University of Saskatchewan).

Pokorný, J., 2015. Graph databases: their power and limitations. In *Computer Information Systems and Industrial Management: 14th IFIP TC 8 International Conference, CISIM 2015, Warsaw, Poland, September 24-26, 2015, Proceedings 14* (pp. 58-69). Springer International Publishing.

Seeger, M. (2009). Key-Value stores: a practical overview. [online] Available at: https://blog.marc-seeger.de/assets/papers/Ultra_Large_Sites_SS09-Seeger_Key_Value_Stores.pdf.

Wikipedia (n.d.) JSON, Wikipedia, accessed 08 September 2024. https://en.wikipedia.org/wiki/JSON.
Wikipedia (n.d.) BSON, Wikipedia, accessed 08 September 2024. https://en.wikipedia.org/wiki/BSON.

NoSQL Databases to Burwood Library

NoSQL databases can significantly improve the efficiency and service delivery of Burwood Library, especially in the face of evolving technology needs and user demands. Libraries like Burwood handle vast amounts of data that span across different formats, ranging from books and multimedia collections to user records, borrowing history, and event management. As modern libraries seek to offer personalized, scalable, and flexible services, NoSQL databases provide an ideal solution for accommodating the growing data variety and volume that traditional relational databases may struggle with.

1. Scalability for Large Data Sets

Burwood Library deals with a continuously expanding collection of digital and physical assets, as well as growing patron data. Traditional relational databases are typically vertically scaled, meaning they need increasingly powerful hardware to handle more data. However, NoSQL databases offer horizontal scalability, allowing the library to efficiently distribute data across multiple servers. This can improve performance when managing large datasets, such as digital book archives, multimedia content, and user-generated data, which can span terabytes or even petabytes. By using

NoSQL, Burwood Library can scale its database system without costly hardware upgrades, ensuring smooth and fast access to resources as the collection grows.

2. Flexible Data Management

NoSQL databases excel at handling unstructured, semi-structured, and structured data, offering much-needed flexibility for modern libraries. Burwood Library likely manages diverse data types beyond traditional text, such as video files, e-books, scanned documents, and metadata. NoSQL databases such as MongoDB or Couchbase allow for the storage of various formats without requiring a rigid, predefined schema, which relational databases would demand. This flexibility is crucial for adapting to evolving data storage needs, as libraries continuously add new types of digital content and services, such as streaming platforms or interactive media.

3. Personalized User Experience

Burwood Library can greatly enhance its user experience by leveraging the capabilities of NoSQL databases. Modern users expect personalized recommendations and easy access to relevant resources. For example, a library could use a NoSQL database to power a recommendation system that suggests books, articles, or multimedia based on a user's borrowing history, search behaviour, or interests. By using graph databases, like Neo4j, the library can build complex relationships between users, books, authors, and genres, allowing for personalized recommendations similar to what is seen on platforms like Netflix or Amazon. This would improve engagement by making it easier for patrons to discover new materials tailored to their preferences.

4. Enhanced Real-Time Services

Libraries increasingly offer real-time services such as live catalogue searches, instant book reservations, and real-time updates on event registrations. NoSQL databases, which are optimized for fast read and write operations, can enhance the responsiveness of such systems. Document-based NoSQL

databases like MongoDB allow for rapid retrieval of catalogue information, enabling real-time search results for users, even during peak usage times. Additionally, NoSQL databases can handle large numbers of concurrent users efficiently, making them suitable for online library portals that serve multiple patrons simultaneously.

5. Event Management and Community Engagement

Libraries, including Burwood Library, often host events, workshops, and community programs. Managing event registrations, user participation, and feedback can generate a wealth of data that NoSQL databases can efficiently handle. NoSQL's ability to manage varied data types allows the library to track event attendees, capture feedback, and analyse trends over time, helping them to better understand community engagement and optimize future event planning. For example, a key-value store NoSQL system like Redis can help manage event ticketing and registration processes in real-time, while a document-based database can store detailed feedback and participant information for future analysis.

6. Cost Efficiency and Resource Optimization

By adopting NoSQL databases, Burwood Library can also benefit from cost efficiency. Many NoSQL solutions are designed to run on commodity hardware, avoiding the need for expensive servers that relational databases often require. This means the library can operate its digital infrastructure at a lower cost while still benefiting from high-performance data management. Moreover, with NoSQL databases' support for distributed systems, the library can ensure uptime and redundancy by distributing data across multiple locations, minimizing the risk of data loss or service interruptions due to server failures.

7. Future-Proofing with Cloud Integration

As cloud computing becomes increasingly important, NoSQL databases are particularly well-suited for integration with cloud services. Libraries, including Burwood Library, are embracing cloud platforms for digital services, and

NoSQL databases offer the scalability and flexibility needed to thrive in such environments. Cloud-native NoSQL solutions, such as Amazon DynamoDB or Google Bigtable, allow libraries to expand their services seamlessly and access a global infrastructure, ensuring that digital resources are available to patrons from anywhere at any time. This opens up opportunities for Burwood Library to expand its reach and serve a broader, potentially international audience with digital content.

Conclusion

By implementing NoSQL databases, Burwood Library can significantly improve its scalability, flexibility, and ability to provide personalized services. NoSQL's capability to handle diverse data types and support real-time operations can enhance user engagement, improve operational efficiency, and reduce costs. Whether managing large multimedia collections, delivering personalized recommendations, or streamlining event management, NoSQL provides the tools necessary for Burwood Library to adapt to the demands of modern digital library services, ultimately ensuring a better experience for patrons and a more efficient system for staff.