The impact of Cloud computing on developing countries

FULL NAME: YUPENG WEN

STUDENT ID: s224212855

TEACHERS: Iynkaran Natgunanathan

DATE OF COMPLETION: 08/09/2024

WORD COUNT: 1018

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

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, this

same structure becomes a significant drawback when dealing with unstructured or semi-structured data, such as images, videos, or social media interactions. Relational databases rely on predefined schemas, meaning that any changes to the data structure require costly modifications. This inflexibility hinders their adaptability in scenarios where the data format frequently evolves or is not well-defined at the outset. Additionally, relational databases rely primarily on vertical scaling, meaning they require more powerful hardware to handle increased loads, which can be both expensive and inefficient when applied to large-scale, distributed systems.

As the volume and variety of data continue to grow, along with the need for real-time processing, the limitations of relational databases have become more pronounced. NoSQL databases emerged as a solution to these challenges, offering flexibility, scalability, and high performance, especially for handling large volumes of unstructured or semi-structured data. Unlike relational databases, NoSQL systems do not require a fixed schema, allowing them to easily accommodate changing data structures. In the study of Azad et al.(2020), This flexibility makes them particularly well-suited for applications like content management, social media, and Internet of Things (IoT) environments, where the data is often unstructured or semi-structured and may evolve over time.

NoSQL databases offer significant advantages when it comes to scalability. While relational databases primarily scale vertically by upgrading hardware, NoSQL databases are designed to scale horizontally, distributing data across multiple servers or nodes. This means that as the system grows, additional servers can be added, enabling the system to handle larger datasets and higher traffic volumes without significant performance degradation. This capability makes NoSQL databases an excellent choice for big data applications and real-time analytics, where massive amounts of data need to be processed and accessed quickly (Mishra et al. 2017).

Another key strength of NoSQL databases is their ability to optimize read and write operations, particularly in environments with high-velocity data. For applications like e-commerce platforms, social media networks, or real-time gaming, the speed at

which data can be read or written to the database is crucial for maintaining performance. NoSQL databases are often designed with this in mind, providing low-latency access to data and enabling fast, efficient data retrieval.

NoSQL databases also follow the CAP theorem, which posits that distributed systems can only guarantee two out of three properties: consistency, availability, and partition tolerance. Most NoSQL systems prioritize availability and partition tolerance over strict consistency, making them highly reliable in distributed environments where server failures or network issues may occur. This design ensures that data remains accessible even in the event of partial system failures, making NoSQL a robust choice for applications requiring high availability and fault tolerance.

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 (e.g., MongoDB, CouchDB): Store data as documents, typically in formats like JSON or BSON, allowing for flexible, nested structures.
Document stores, such as MongoDB and CouchDB, are a type of NoSQL database that store data as individual documents. These documents are typically in formats like JSON (JavaScript Object Notation) or BSON (Binary JSON), which provide a

flexible and schema-less structure. Unlike relational databases that require predefined table schemas, document stores allow for the storage of complex, nested data, making them ideal for applications where data structure may evolve over time or where the data is inherently unstructured. According to the study by Wang et al. (2020), a schema suggestion model for NoSQL document-store databases to address the challenges in modelling these flexible databases. Each document in a document store represents a self-contained unit of data and can vary in structure from other documents, offering significant flexibility. This makes document stores particularly useful for applications such as content management systems, catalogues, and user profiles, where each document may have unique attributes.

Moreover, document stores allow for easy scaling and high performance, especially for read-heavy or write-heavy workloads. MongoDB, for example, is optimized for handling large volumes of data and can distribute documents across multiple servers, ensuring both scalability and availability in distributed environments. This flexibility and adaptability to real-time data needs have made document stores a popular choice for modern applications.

Key-Value Stores

Key-value stores are a type of NoSQL database that use a simple key-value pair model to store data. In this model, each piece of data is associated with a unique key, allowing for quick lookups and retrieval. The simplicity of this structure makes key-value stores highly performant and scalable, particularly for applications that require fast read and write operations. Because there is no need for complex data relationships or joins, key-value stores are ideal for use cases like caching, session management, and real-time analytics, where speed and efficiency are critical.

Popular key-value databases, such as Redis and Amazon DynamoDB, can handle large volumes of data and transactions at high speed, distributing data across multiple nodes for improved scalability.

The design of key-value stores allows for straightforward horizontal scaling, meaning they can easily expand to accommodate growing data by adding more servers. This makes them especially useful in environments where fast access to data is a priority and the structure of the data is relatively simple.

Column-Family Stores

Column-family stores, a type of NoSQL database, organize data into columns rather than traditional rows, optimizing performance for large-scale distributed systems and real-time analytics. Instead of storing data in a row-oriented format like relational databases, column-family stores group data by columns, which allows for more efficient querying and data retrieval, especially when only certain columns need to be accessed (Counte E and Cuza C.m 2018). This structure is particularly useful for applications dealing with massive amounts of data, such as log analysis, recommendation engines, and data warehousing. By storing related data together in columns, these databases can quickly retrieve specific attributes without having to scan entire rows, significantly improving query performance.

Base on the study of Sheth V 2023, Comparative performance analysis has shown that column-family databases, like Cassandra and HBase, can offer better performance than traditional relational databases for certain types of workloads, particularly those involving large-scale data storage and retrieval. They can efficiently distribute data across multiple servers, ensuring high availability and fault tolerance. This makes them well-suited for systems that need to process and analyse real-time data streams or large datasets across multiple nodes. Additionally, the columnar format is beneficial for data compression, further enhancing 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 particularly well-suited for handling complex and interconnected data. Unlike relational databases that rely on tables and predefined relationships, graph databases use graph structures to represent and query relationships dynamically, allowing for more efficient processing of data with complex interconnections (Pokorný 2015). In a graph database, data is modelled as a collection of nodes, which represent entities such as people, products, or places, and edges, which represent relationships between these nodes. Each node and edge can hold properties, providing detailed information about the entities and their connections. This structure makes it easy to model real-world relationships and navigate through them quickly, offering a highly intuitive way to represent complex

data patterns. One of the key strengths of graph databases is their ability to efficiently handle queries involving relationships and connections, which are often cumbersome and inefficient in traditional relational databases. In applications like social networks, recommendation engines, fraud detection, and supply chain management, relationships between entities are central to the functionality. For example, in a social network, users are connected through various types of relationships such as friendships, likes, or follows. Graph databases allow for the quick traversal of these connections, making it easier to answer queries like "Who are the mutual friends of two users?" or "Which products are often bought together?"

The performance of graph databases shines in scenarios where relationships need to be explored deeply and in real-time. Since the database structure is optimized for relationship-based queries, traversing through connected data is faster and more efficient compared to relational databases, where such queries would involve multiple table joins and could become computationally expensive as the dataset grows. Popular graph databases like Neo4j, Amazon Neptune, and ArangoDB have become essential tools for industries that rely on complex data networks. In addition to social networks and recommendations, graph databases are widely used in areas like knowledge management, where intricate relationships between data points must be stored and queried. They are also useful in fraud detection, where entities such as financial transactions, users, and accounts can be connected in a graph to detect suspicious patterns of behaviours.

Graph databases provide flexibility in data modelling, as new nodes and relationships can be added without the need for schema changes or reconfiguration. This adaptability makes graph databases an excellent choice for systems that evolve over time and require continuous updates to their relationships.

In conclusion, graph databases excel in managing and querying complex, interconnected data. By representing data as nodes and relationships, they provide a powerful and intuitive way to model and explore connections. Their high performance in handling relationship-based queries makes them invaluable for applications ranging from social networking and recommendation systems to fraud

detection and knowledge management, making them a critical tool in modern data management.

Advantages of NoSQL Databases

NoSQL databases offer several significant advantages over traditional relational databases, particularly in modern applications managing large-scale data. A key benefit is scalability—NoSQL databases are designed for horizontal scaling, enabling them to handle massive datasets and high traffic across distributed systems, making them ideal for cloud environments. Their flexibility is another major asset; without the constraints of a rigid schema, NoSQL databases can easily accommodate unstructured and semi-structured data, adapting to evolving requirements and diverse data formats. High performance is also a notable advantage, as NoSQL systems are optimized for fast read and write operations, especially in scenarios where complex joins and transactions are unnecessary. Additionally, NoSQL databases tend to be more cost-effective, as they can scale horizontally using commodity hardware, avoiding the need for the expensive, high-performance servers that relational databases require for vertical scaling. These features collectively make NoSQL databases a compelling choice for modern, data-intensive applications.

Disadvantages of NoSQL Databases

Despite their advantages, NoSQL databases come with certain drawbacks that warrant careful consideration. One significant limitation is the lack of full ACID transaction support. Many NoSQL databases prioritize availability and partition tolerance over consistency, as per the CAP theorem, which can be problematic for applications requiring strong consistency guarantees. Additionally, NoSQL databases often have more limited query capabilities compared to relational databases. While SQL offers powerful, standardized query options, querying in NoSQL systems can be more complex, especially across multiple nodes or collections, often necessitating custom programming solutions.

Another challenge is data duplication, as NoSQL databases prioritize performance over strict normalization. This can lead to redundant data storage across various documents or collections, increasing storage requirements and causing potential

consistency issues during updates. Lastly, while NoSQL databases have grown in popularity, they are relatively newer compared to relational databases, often lacking the same level of maturity, robust tooling, and community support. This can be particularly limiting for applications with complex analytical needs or those relying on well-established ecosystems. Despite these challenges, understanding these limitations helps in making informed decisions when choosing the right database for specific use cases.

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

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.

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.

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).

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.

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.