Proceedings of CONF-MLA 2026 Symposium: Learning and Decision Making in Multi Agent Software Systems
DOI: 10.54254/2755-2721/2026.BA38542

Traditional Relational Databases vs. NoSQL Databases: A Review of Differences and Application Scenarios

Qihao Li

College of Light Industry Science and Engineering, Sichuan University, Chengdu, China
Liqh822@gmail.com

Abstract. Since Codd proposed the relational model in 1970, relational databases have been widely adopted in enterprise data management because of their structured model, standardized SQL, and ACID transactions. With the growth of the Internet, cloud computing, and the Internet of Things, data volume, types, and business requirements have become increasingly diverse. Traditional relational databases therefore face new technical requirements in large-scale distributed, high-concurrency, and frequently changing schema scenarios. NoSQL databases have emerged with document, graph, key-value, and column-family models and distributed architectures. Using MongoDB and Neo4j as examples, this paper compares relational and NoSQL databases in technical characteristics and application scenarios, and examines e-commerce, social networks, financial risk control, and enterprise production management. It argues that the two paradigms are complementary rather than substitutive, and that database selection should balance data structure, transactional requirements, access patterns, and scalability. This study provides references for database selection, system architecture design, and data management optimization.

Keywords: Relational databases, NoSQL databases, MongoDB, Neo4j, Database selection

1. Introduction

Since Codd proposed the relational model in 1970, relational databases have long dominated enterprise information management and data processing due to their mature relational model, standardized SQL query language, and ACID transaction mechanisms [1]. However, with the rapid development of the Internet, cloud computing, and the Internet of Things, data volume has continued to grow and data types have become increasingly diverse. As a result, traditional relational databases have revealed limitations in massive data storage, high-concurrency access, and distributed scaling [2].

To satisfy the data management needs of the big data era, NoSQL databases have been regarded as an effective method [3]. Different from relational databases, NoSQL databases employ flexible data models and distributed structures which are beneficial for dealing with semi-structured and unstructured data and for handling high-concurrency accesses [4]. Among these, MongoDB is appropriate for saving flexible data; Neo4j is good at studying complex relationships and carrying out graph computations and is extensively used in e-commerce recommendation systems, social networks and financial risk controls [5]. This paper investigates the development of relational and

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.BA38542

NoSQL databases and contrasts their data models, transaction procedures, scalability and general application fields. By taking MongoDB and Neo4j as examples, it analyzes the technical advantages and practical applications of NoSQL databases. At last, it reviews the trend of integrating relational and NoSQL databases, offering a theoretical foundation for enterprises to choose databases, design system architectures and optimize data storage according to the different performance requirements, data features and scalability demands in various business situations.

2. Overview of relational and NoSQL database fundamentals

2.1. The core characteristics and inherent limitations of traditional relational databases

Relational databases are founded on the relational model, store data in two-dimensional tables, define structures by means of predetermined schemas and employ SQL for retrieval and modification [1]. Their main characteristics comprise a definite structure, well-developed integrity rules and ACID transactions. The benefits of relational databases are their data normalization, transaction processing and the language environment. By joining tables, columns and constraints, they express logical relations and maintain the accuracy of data through entity and referential integrity. ACID transactions carry out several operations as a whole unit, which is indispensable in updating and accounting for multiple tables. SQL also facilitates complicated queries, summary analysis and multi-table joins. These advantages have maintained the dominant position of relational databases in enterprise management, financial transactions and important business systems for a long time.

Moreover, in some new situations, relational databases have some restrictions. If the requirements alter quickly or the schema changes frequently, the predefined schemas need to be modified, leading to data migration and compatibility problems and thus increasing the development and maintenance expenses [2]. Relational databases were mainly developed for ACID transactions and structured data; therefore, their scalability horizontally in distributed systems is relatively small. In large clusters, cross-node transactions or distributed joins usually require sharding, separation of reads and writes or distributed SQL middleware, which makes the system structure more complex and increases the maintenance costs [6]. Besides, multi-table joins are used in relational model to represent the relationships. When dealing with multi-level relationships, many joins are required for the query; as the number of join layers increases, the SQL complexity and the execution overhead become greater, which may not satisfy the online requirements with low latency [6].

2.2. The evolution and overall technical characteristics of NoSQL databases

The advancement of NoSQL databases is associated with the increase of Internet applications. Relational databases are especially suitable for structured data and transaction processing, whereas the Internet applications produce large volumes of diverse data like logs, user behaviors, social connections and product characteristics which vary in structure, size and accessing ways. Therefore, NoSQL databases developed for particular data types and distributed needs have appeared [2, 3].

NoSQL is not a kind of technology, but a group name for non-relational data handling techniques. According to the data model, they are mainly divided into key-value (Redis, DynamoDB), column-family (Cassandra, HBase), document (MongoDB) and graph (Neo4j) databases [2]. The objectives of their designs are different: key-value databases are suitable for high-speed access to simple information; column-family databases are suitable for large-scale distributed processing; document

DOI: 10.54254/2755-2721/2026.BA38542

databases are suitable for complex and hierarchical data; and graph databases are suitable for the complex relationships between entities requiring frequent traversal.

NoSQL databases have flexible data models and employ sharding and replication for scalability [2, 3]. However, they do not entirely neglect transactions, consistency or availability; hence, it is incorrect to believe that NoSQL databases discard ACID or merely focus on performance. At present, some advanced NoSQL databases are improving their transactional capabilities, consistency and data management.

The main advantage of NoSQL databases is not to substitute relational databases entirely, but to provide additional data management techniques for various data structures and access modes. While choosing a database, enterprises should compare data structure, transactional needs, access methods, scalability and maintenance costs.

2.3. Typical NoSQL databases: an introductory overview of MongoDB and Neo4j

MongoDB is a document-oriented database using BSON documents. Unlike fixed relational tables, it allows flexible field structures, making it suitable for applications with varying attributes or frequent schema changes. It also offers indexing, aggregation, replication, and sharding for large-scale storage and retrieval [7]. Its use cases include product information, content, and some user behavior data.

Neo4j is a graph database using a property graph model, with nodes, relationships, and attributes, and uses Cypher for graph manipulation [5]. Unlike relational databases that express relationships through joins, graph databases treat relationships as part of the data model, making them suitable for relationship-intensive applications such as social networks, recommendation systems, and financial risk control [8, 9]. Almabdy's experiments on Twitter show that for social relationship retrieval within two hops, Neo4j's response time is significantly shorter than MySQL's [9]. This indicates a performance advantage in medium-depth traversal, though it may not apply to ultra-large-scale graphs or extremely deep path queries.

In summary, MongoDB addresses flexible data structures, while Neo4j addresses complex relationships. Database selection should not rely solely on the "NoSQL" label but on the data model and query patterns.

3. Comparison of typical application scenarios for relational databases versus NoSQL databases

3.1. Analysis of database selection for two different e-commerce business scenarios

E-commerce platforms include both transactional and non-transactional data. Orders, payments, and inventory are structured and require high consistency; product information, reviews, and some user behavior data are more flexible [10]. For core transactions such as order processing, payment, and inventory, relational databases offer advantages because order creation, inventory deduction, and payment updates require transactions for consistency [11].

MongoDB suits product information, shopping carts, and content data. Product information has many unstructured fields and varies greatly across categories. Traditional relational databases often need complex "entity-attribute-value" designs or frequent schema changes, whereas MongoDB's dynamic document model maps hierarchical product data directly [12]. Research shows that in e-commerce big data, MongoDB significantly outperforms MySQL in retrieval and insertion [13].

DOI: 10.54254/2755-2721/2026.BA38542

Neo4j offers unique value in e-commerce recommendation. By building a "User-Product-Browse-Purchase-Review" graph, it enables real-time path queries and community discovery for collaborative filtering. Traditional relational databases require multiple joins for multi-hop queries, and performance declines exponentially with depth, whereas Neo4j's traversal performance remains nearly constant [5, 14]. Therefore, large e-commerce platforms often adopt a multi-database architecture of "relational + document + graph" databases, each handling suitable workloads [10, 11].

3.2. Analysis of differences in adaptation between social networks and financial risk control scenarios

Social networks such as Facebook and Twitter are relationship-intensive. Users, friends, followers, comments, and interactions create many relationships. Multi-level queries require multiple joins in relational databases, making query logic complex. Strong consistency and complex joins become burdens on availability and scalability. Graph databases represent social networks directly with nodes and relationships, making them suitable for traversal and association analysis [9, 14]. MongoDB can manage flexible data such as user profiles and dynamic content [15]. Thus, graph and document databases are complementary, handling relationship analysis and content management respectively.

Financial risk control requires both transaction processing and complex correlation analysis. Account balances and transaction records need high consistency, while fraud detection and anti-money laundering require analyzing relationships among accounts, individuals, devices, and transactions [14, 16]. Relational databases can manage core transaction data, while graph databases can analyze relationship networks and identify risks, supporting a hybrid architecture. Database selection should therefore be based on specific tasks, not on database type.

3.3. Analysis of typical industry application cases for two types of databases

Practical cases illustrate technical applicability boundaries. In e-commerce, Ramesh et al. developed a MongoDB-based solution organizing products, shopping carts, and inventory as documents [11]. This shows MongoDB can represent hierarchical data and offers a viable approach for frequently changing e-commerce data. However, lacking high-concurrency and large-scale test data, its production performance and stability require further validation.

In social networking, Facebook's TAO system combines relational and graph data management. Facebook uses sharded MySQL for persistence and builds TAO for social relationship access, using objects and associations as core abstractions [2]. Facebook does not replace MySQL with a graph database but uses different components for different tasks, creating a collaborative architecture.

In financial risk control, China UnionPay's graph database case shows practical value in complex correlation analysis. When using a relational database, UnionPay faced challenges in fraud detection in large-scale transactions, so it transitioned to graph database technology [16]. The system uses historical transaction data to generate risk rules and loads patterns into the production database for matching; for suspicious transactions, it implements automatic blocking or manual investigation based on risk assessment [16]. This shows graph databases can support fraud detection but cannot fully replace relational databases.

In enterprise production management, Kudo et al. compared MongoDB and MySQL for production management systems involving inventory, images, and video [17]. Results show performance varies by operation type; MongoDB has advantages in some high-volume operations,

DOI: 10.54254/2755-2721/2026.BA38542

though migration may increase program complexity [17]. Thus, MongoDB is viable for systems with images and video, but database choice should be based on data types and operations.

Overall, MongoDB adapts to flexible data structures and large data volumes, while Neo4j handles complex relationships. The Facebook TAO and China UnionPay cases show that in large systems, relational and NoSQL databases are more often synergistic than substitutive.

4. The convergence trend between relational and NoSQL databases

4.1. Mainstream implementation approaches for hybrid database architectures

As business operations become more complex, a single database type cannot meet all data management needs, so multi-database collaboration and hybrid architectures have gained attention [4, 18]. The core idea is to store data in appropriate databases based on data characteristics and access requirements, and to enable collaboration through an intermediate layer, data synchronization, or a unified access mechanism. Figure 1 shows a typical hybrid SQL/NoSQL architecture. The application layer includes e-commerce, social networking, financial risk control, and enterprise management; through a unified access middleware layer providing query parsing, request routing, data synchronization, and a unified API, it connects to relational databases, MongoDB, and Neo4j, forming a three-tier architecture: business applications, middleware, and database layer.

Bjeladinović et al. proposed an SQL/NoSQL integration architecture that enables unified access and management of database components through a Wrapper Layer [18]. It includes statement parsing, statement mapping, constraint management, and integrated control, allowing applications to access different databases through a unified interface, reducing heterogeneous database access complexity.

The Facebook TAO case also illustrates hybrid architecture practice. It uses MySQL for persistent storage and a dedicated data access layer for high-frequency social relationship data [2]. This shows that hybrid architectures do not require different databases to share the same data model; they can handle different tasks at different levels. Therefore, the core of hybrid architecture is not simply adding databases, but assigning responsibilities based on data types and workloads and establishing data exchange and access mechanisms.

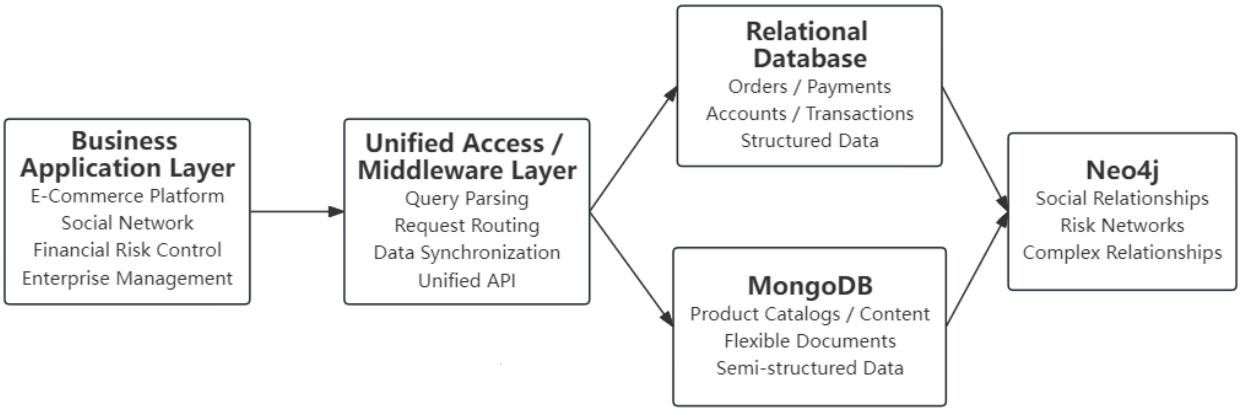

Figure 1. Hybrid SQL/NoSQL database architecture

Note. Different databases assume distinct business responsibilities and are coordinated through the middleware layer.

DOI: 10.54254/2755-2721/2026.BA38542

4.2. Hybrid architecture: collaborative advantages and implementation challenges

The greatest advantage of hybrid architecture is leveraging different databases' strengths. Relational databases handle orders, payments, and accounts; MongoDB stores flexible product or content data; Neo4j handles complex relationship analysis. This avoids forcing one database to meet all needs.

However, hybrid architecture increases complexity. Data synchronization may cause consistency issues, especially when the same data exists in multiple systems; update order and synchronization mechanisms must be defined. Different data models and query languages require an additional intermediate layer or integration mechanism for cross-database queries [18]. Multi-database environments also increase maintenance costs, including backup, monitoring, fault recovery, and performance optimization. Thus, hybrid architecture is not a simple combination but a data partitioning and architecture design based on business requirements. Future research can focus on cross-database query optimization, heterogeneous consistency, data synchronization, and unified access interfaces.

5. Conclusion

This paper reviews the technical characteristics and application scenarios of relational and NoSQL databases. Relational databases, with structured models, SQL, and ACID transactions, remain valuable for core business with high consistency requirements. NoSQL databases offer tailored solutions for flexible, large-scale, or relationship-intensive data through document and graph models. MongoDB and Neo4j represent document and graph databases, respectively. MongoDB suits flexible structures and diverse patterns, with advantages in e-commerce product information and content management; Neo4j suits complex entity relationships, performing well in social networks and financial risk control. Cases show that database suitability depends on business requirements, not type alone.

Relational and NoSQL databases are not merely competitive; they are more likely to complement each other through hybrid architectures. Enterprises can select databases based on data structures, transaction requirements, query patterns, and scalability, achieving collaboration through middleware and data synchronization. Database selection should balance consistency, flexibility, scalability, query efficiency, and system complexity. This paper focuses on MongoDB and Neo4j, with limited discussion of key-value and column-family databases; case conclusions are influenced by hardware and data scale, limiting representativeness; and cloud-native and distributed databases are not deeply explored. Future research can focus on multi-database collaboration, cross-database consistency, automated selection, and workload-specific performance optimization.

References

[1] Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM, 13*(6), 377-387.

[2] Davoudian, A., Chen, L., & Liu, M. (2018). A survey on NoSQL stores. *ACM Computing Surveys (CSUR), 51*(2), 1-43.

[3] Khan, W., Kumar, T., Zhang, C., Raj, K., Roy, A. M., & Luo, B. (2023). SQL and NoSQL database software architecture performance analysis and assessments—a systematic literature review. *Big Data and Cognitive Computing, 7*(2), 97.

[4] Ana-Gabriela, B. (2024). Choosing the appropriate database type: A comparison of SQL, NoSQL, and NewSQL. *Annals of "Constantin Brâncuși" University of Târgu-Jiu. Economy Series, 6*, 2024-06.

[5] Angles, R., & Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys (CSUR), 40*(1), 1-39.

DOI: 10.54254/2755-2721/2026.BA38542

[6] Khasawneh, T. N., AL-Sahlee, M. H., & Safia, A. A. (2020, April). SQL, NewSQL, and NoSQL databases: A comparative survey. In *2020 11th International Conference on Information and Communication Systems (ICICS)* (pp. 013-021). IEEE.
[7] Rathore, M., & Bagui, S. S. (2024). MongoDB: Meeting the dynamic needs of modern applications. *Encyclopedia, 4*(4), 1433-1453.
[8] Guia, J., Soares, V. G., & Bernardino, J. (2017, April). Graph databases: Neo4j analysis. In *ICEIS (1)* (pp. 351-356).
[9] Almabdy, S. (2018, April). Comparative analysis of relational and graph databases for social networks. In *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)* (pp. 1-4). IEEE.
[10] Liu, X. (2015). An analysis of relational database and NoSQL database on an ecommerce platform. Trinity College.
[11] Ramesh, D., Khosla, E., & Bhukya, S. N. (2016, December). Inclusion of e-commerce workflow with NoSQL DBMS: MongoDB document store. In *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (pp. 1-5). IEEE.
[12] Aboutorabi[a], S. H., Rezapour, M., Moradi, M., & Ghadiri, N. (2015, August). Performance evaluation of SQL and MongoDB databases for big e-commerce data. In *2015 International Symposium on Computer Science and Software Engineering (CSSE)* (pp. 1-7). IEEE.
[13] Jose, B., & Abraham, S. (2020). Performance analysis of NoSQL and relational databases with MongoDB and MySQL. *Materials Today: Proceedings, 24*, 2036-2043.
[14] Sadowski, G., & Rathle, P. (2014). Fraud detection: Discovering connections with graph databases. *White Paper-Neo Technology-Graphs are Everywhere, 13*, 1-13.
[15] Cattuto, C., Quaggiotto, M., Panisson, A., & Averbuch, A. (2013, June). Time-varying social networks in a graph database: A Neo4j use case. In *First International Workshop on Graph Data Management Experiences and Systems* (pp. 1-6).
[16] Henderson, R. (2020). Using graph databases to detect financial fraud. *Computer Fraud & Security, 2020*(7), 6-10.
[17] Kudo, T., Ito, Y., & Serizawa, Y. (2017). An application of MongoDB to enterprise system manipulating enormous data. *Int, Journal of Informatics Society, 9*(3), 97-108.
[18] Bjeladinovic, S., Marjanovic, Z., & Babarogic, S. (2020). A proposal of architecture for integration and uniform use of hybrid SQL/NoSQL database components. *Journal of Systems and Software, 168*, 110633.