International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3319-3325
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0426

SQL vs NoSQL vs NewSQL: A Comparative Study of Modern Database Technologies

Usha B A [1], Srivatsala V[2], Yashaswini E[3], Sahana Patil [4]
[1,3,4] PG- Department of Computer Applications, Dayananda Sagar College of Arts Science & Commerce, Bangalore, Karnataka
[2] Assistant Professor, Department of Computer Applications, Dayananda Sagar College of Arts Science & Commerce, Bangalore, Karnataka
Email ID: *ushaprabha8792@gmail.com[1], vsrivatsala123@yahoo.co.in[2], yashaswinie93@gmail.com[3], psahana900@gmail.com[4]*

Abstract
The rapid development of modern digital technologies, such as cloud computing, social networking platforms, and Internet of Things (IoT) systems, has significantly increased both the quantity and complexity of data generated by organizations. This growing demand has led to the need for more advanced and efficient database management approaches. As a result, three major database paradigms have evolved: SQL, NoSQL, and NewSQL. Relational databases based on SQL store data in structured tables and maintain strong data integrity by following ACID (Atomicity, Consistency, Isolation, Durability) principles. These characteristics make them highly suitable for applications that require accurate and reliable transactions, such as financial systems and enterprise applications. In contrast, NoSQL databases were developed to overcome the limitations of traditional relational systems, especially in handling large-scale and distributed data. They provide flexible data models and support horizontal scaling, allowing them to efficiently manage different types of data, including document-based, key-value, column-oriented, and graph-based formats. NewSQL databases represent a modern approach that combines the advantages of both SQL and NoSQL systems. They preserve the structured data model and strong consistency of relational databases while also offering improved scalability and performance similar to NoSQL solutions. This paper conducts a structured comparison of all three database paradigms, examining their architectural differences, scalability characteristics, consistency guarantees, and practical trade-offs, with the objective of enabling researchers, practitioners, and system architects to make well-informed technology decisions.

Keywords: *SQL Databases, NoSQL Databases, NewSQL, Distributed Databases, Big Data, Data Management*

1. Introduction
In today's data-driven world, the ability to efficiently store, retrieve, and process information has become a strategic imperative for enterprises, governments, and technology providers alike. The proliferation of mobile applications, cloud platforms, social networks, and interconnected IoT devices generates petabytes of data at an ever-increasing rate, placing enormous demands on the underlying infrastructure designed to manage it. For several decades, relational database management systems, commonly known as SQL databases, served as the dominant solution for structured data storage. By organizing data into rows and columns within tables and enforcing transactional rules via ACID properties (Atomicity, Consistency, Isolation, and Durability) [1], these systems delivered the reliability and predictability needed in sensitive domains such as financial services, healthcare record management, and enterprise software. However, the emergence of distributed computing and internet-scale applications exposed inherent challenges in the traditional relational model. Scaling SQL databases horizontally across commodity hardware clusters proved technically complex and financially costly. Furthermore, the rigid, predefined schema structures of relational databases were poorly suited to storing the fluid, semi-structured or unstructured data produced by many contemporary applications [2].

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3319-3325
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0426

These challenges catalyzed the development of NoSQL database systems. Designed with distribution in mind, NoSQL databases support flexible, schema-free data models and can efficiently spread workloads across many servers. The NoSQL category encompasses several distinct sub-paradigms, including key-value stores, document-oriented databases, column-family stores, and graph databases, each optimized for different data patterns and access requirements [3]. While NoSQL systems deliver advantages in scalability and throughput, they typically do so at the cost of strict consistency. Many adopt eventual consistency models, which may be inappropriate for workloads requiring precise transactional guarantees. NewSQL databases were created to address this gap, offering the distributed scalability of NoSQL alongside the relational structure and strong consistency of traditional SQL platforms [4]. This paper presents a structured comparative analysis of all three database families, evaluating their architectural foundations, scalability models, consistency characteristics, performance profiles, and real-world applicability [10][11][12].

2. Literature Review

The development of modern database systems can be traced back to the groundbreaking work of Edgar F. Codd in 1970, where he introduced the concept of the relational data model. In this approach, data is organized into tables consisting of rows and columns, providing a structured and logical way of storing information. This model laid the foundation for SQL-based database systems and significantly influenced how data is managed today. The relational model gained widespread acceptance, especially in enterprise environments, because it ensured data accuracy, maintained a clear structure, and supported reliable and efficient query processing. These features made it highly suitable for applications where consistency and performance were critical. Another important concept in distributed system design is the CAP theorem, proposed by Eric Brewer. This principle explains that a distributed database cannot simultaneously achieve consistency, availability, and partition tolerance. Instead, system designers must choose which aspects to prioritize. Many NoSQL databases are built with this understanding, often focusing on availability and partition tolerance while accepting weaker forms of consistency, such as eventual consistency. Further research by Abadi expanded on these ideas by examining the practical trade-offs involved. His work highlighted that although NoSQL databases perform well in terms of scalability and high data processing speed, they may not be suitable for applications that require strong consistency and accurate transaction handling. He stressed that selecting the right database system depends on the specific needs of the application, particularly in terms of consistency and performance requirements. Overall, these research contributions illustrate the evolution of database systems—from traditional relational databases to highly scalable distributed systems, and eventually to NewSQL solutions that aim to combine the benefits of both approaches while reducing their limitations.

3. Research Methodology

This study focuses on analyzing and comparing SQL, NoSQL, and NewSQL database systems based on various important factors such as their features, performance, scalability, and suitability for handling modern data management requirements. The research approach is grounded in systematic literature review and qualitative analysis of published technical work [10]. The initial phase of this research involved a comprehensive survey of academic and professional literature related to database management systems and distributed computing. Sources consulted include peer-reviewed journal articles, conference proceedings, and authoritative technical white papers, with particular attention to publications examining the historical evolution of relational databases, the emergence of NoSQL platforms, and the more recent development of NewSQL technologies [1]. The CAP theorem, as formalized by Eric Brewer, is treated as a fundamental analytical lens through which the trade-offs of distributed database systems are understood. Its implications for consistency, availability, and partition tolerance are considered in evaluating the suitability of each database paradigm for different application scenarios [3]. A structured set of evaluation criteria was selected to facilitate objective comparison across database families. These criteria include data model architecture, horizontal and vertical scalability, consistency and transaction support, schema

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3319-3325
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0426

flexibility, query language capabilities, runtime performance, and domain-specific suitability. SQL databases are assessed primarily on their relational structure and ACID compliance [1], NoSQL databases on their distributed storage capabilities and schema adaptability, and NewSQL databases on their capacity to integrate the transactional guarantees of SQL with the distributed performance of NoSQL systems [10][12]. The results of this analysis are synthesized into a comparative table and narrative discussion that highlight the distinguishing characteristics, strengths, and limitations of each database category. This methodology is intended to yield actionable insights that support developers, researchers, and organizations in selecting database technologies aligned with their specific requirements.

4. Evolution of SQL, NoSQL, and NewSQL

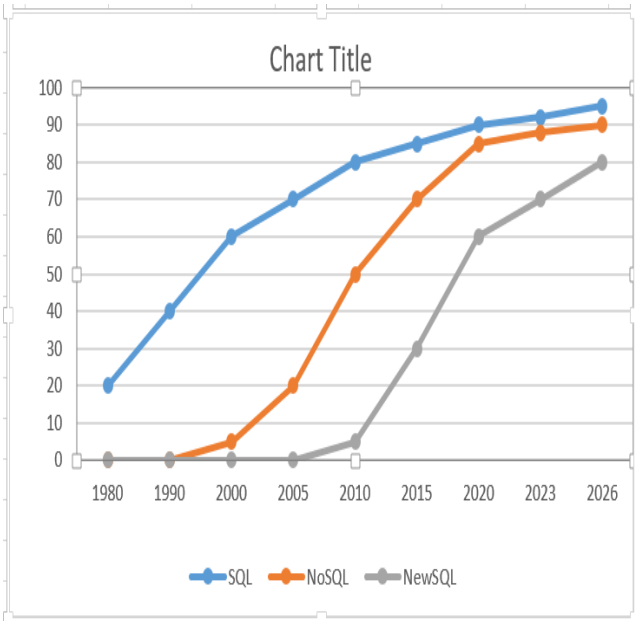

Figure 1 Evolution of SQL, NoSQL, and NewSQL Database Systems

Despite the growing diversity of database options, SQL-based systems continue to hold the dominant share of adoption, reflecting decades of proven reliability and developer familiarity. Research indicates that roughly 50 to 60 percent of developers actively work with SQL platforms such as MySQL and PostgreSQL [1]. These systems remain especially prevalent in industries where data accuracy and transactional integrity are non-negotiable, including banking, insurance, and enterprise applications. In terms of operational performance, SQL databases deliver consistent, stable throughput for transactional workloads, though their scaling characteristics can be less favorable in highly distributed or geographically dispersed deployments [10]. The expansion of big data and modern web-scale applications has driven significant growth in NoSQL adoption over the past two decades. Approximately 30 to 40 percent of developers now incorporate NoSQL technologies such as MongoDB and Apache Cassandra into their workflows [13][14]. These systems are particularly valued for their ability to sustain high read and write throughput, and their horizontal scalability makes them well-suited to data-intensive scenarios such as real-time analytics and social media platforms [3]. NewSQL databases, represented by platforms such as Google Spanner and CockroachDB, are newer entrants in the market and currently account for approximately 10 to 15 percent of usage. Their defining value proposition is the combination of distributed scalability with strong transactional consistency, positioning them as strong candidates for modern high-demand applications [10][11][12]. Figure 1.

5. Comparative Analysis of SQL, NoSQL, and NewSQL Databases

Table 1 Databases

Feature	SQL Databases	NoSQL Databases	NewSQL Databases
Data Model	Relational; data is organized in tables with defined rows and columns	Non-relational; supports key-value, document, column-family, and graph formats	Relational model, optimized for use in distributed computing environments
Schema Structure	Fixed and predefined schema structure	Flexible or dynamic schema design	Structured schema with greater adaptability

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3319-3325
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0426

Scalability	Primarily vertical scaling; performance is improved by upgrading server hardware	Horizontal scaling; data is distributed across multiple servers	Horizontal scalability enabled by a distributed system architecture
Consistency Model	Strong consistency enforced via ACID compliance	Often employs eventual consistency in line with CAP theorem trade-offs	Retains ACID compliance while also supporting distributed transactions
Transaction Support	Comprehensive support for multi-step complex transactions	Transaction capabilities are limited in many systems	Full ACID transaction support, comparable to traditional SQL
Query Language	Uses Structured Query Language (SQL)	Employs various query methods specific to each database type	SQL-compatible query interface
Performance	Reliable performance for structured data and complex relational queries	High throughput suited for large-scale distributed data environments	Strong performance across both transactional and distributed workloads
Data Type Support	Primarily handles structured data	Handles structured, semi-structured, and unstructured data	Focused on structured data, with architecture optimized for scale
Examples	MySQL, PostgreSQL, Oracle Database	MongoDB, Apache Cassandra	Google Spanner, CockroachDB
Best Use Cases	Banking, ERP, financial transaction systems	Big data analytics, social media platforms, IoT deployments	Large-scale cloud services, real-time financial transaction systems

6. Use Cases

6.1.SQL (Structured and Reliable Systems)

SQL databases are most at home in environments where data must be precisely organized and consistently accurate. Their table-based relational structure and support for inter-table relationships make them especially well-suited to transaction-intensive systems such as payroll processing, airline reservation platforms, banking applications, and enterprise resource planning tools [1]. In such contexts, even minor data discrepancies can have serious downstream consequences, making the strict transactional guarantees provided by ACID-compliant SQL engines indispensable. These systems also excel at executing complex, multi-table queries. Widely adopted examples include MySQL, PostgreSQL, and Oracle Database.

6.2.NoSQL (Flexible and Scalable Systems)

NoSQL databases are designed for modern, large-scale applications in which data does not conform to a fixed or predictable structure. They are commonly deployed in social networking platforms, e-commerce systems, real-time analytics pipelines, and IoT data collection architectures, all of which generate high volumes of varied data formats. A key advantage of NoSQL systems is their schema flexibility: data structures can evolve without requiring system-wide migrations or service interruptions [5][6][7]. Horizontal scaling allows these systems to grow incrementally by adding nodes, making them capable of supporting millions of concurrent users and extremely large datasets while maintaining acceptable response times. Widely used NoSQL platforms include MongoDB and Apache Cassandra [13][14].

NewSQL (Balanced and High-Performance Systems)

NewSQL databases occupy a strategic middle ground, designed for applications that demand both the performance of distributed systems and the transactional consistency of relational databases. Typical deployment scenarios include digital payment processing, fintech platforms, ride-hailing

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3319-3325
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0426

services, and online gaming infrastructure, all of which involve high transaction volumes where both speed and data accuracy are critical. NewSQL achieves this balance by supporting ACID compliance while also enabling horizontal scaling across distributed node clusters [10], making these systems suitable for large-scale, performance-sensitive applications that cannot tolerate either data inconsistency or processing bottlenecks. Representative NewSQL platforms include Google Spanner and CockroachDB [11][12].

7. Technologies

The SQL database landscape is anchored by a set of well-established and broadly adopted platforms. MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, and SQLite are among the most widely deployed relational systems. These technologies are the preferred choice in scenarios requiring high data reliability, complex query support, and structured data organization [1]. NoSQL technologies have proliferated in parallel with the growth of distributed data architectures. Prominent platforms in this space include MongoDB for document storage, Apache Cassandra [13][14] for wide-column workloads, Redis for in-memory key-value caching, HBase for Hadoop-integrated column storage, and Neo4j for graph data management. These tools are favored in applications where data formats evolve rapidly and elastic horizontal scaling is a core requirement [3][6][7]. NewSQL technologies are represented by a newer generation of platforms engineered to merge relational consistency with distributed performance. Google Spanner [12], CockroachDB, VoltDB [11], NuoDB, and SingleStore are notable examples. These systems are typically deployed in high-concurrency, performance-critical environments where strong consistency cannot be traded away for scalability gains [10][12]. The table below presents a chronological overview of notable database technology releases from 2015 through 2026, organized by paradigm. This timeline illustrates how each category has continued to grow and innovate in recent years. Table 2.

Table 2 Timeline of Notable Database Technology Releases (2015–2026)

Year	SQL	NoSQL	NewSQL
2015	Amazon Aurora	—	CockroachDB
2016	—	—	NuoDB
2017	—	—	Google Spanner
2018	Snowflake	Azure Cosmos DB	—
2019	—	—	SingleStore
2020	—	FaunaDB	TiDB
2021	PlanetScale	—	—
2022	Neon	—	—
2023	Supabase, AlloyDB	—	—
2024	—	—	EdgeDB
2025	DuckDB	SurrealDB	YugabyteDB
2026	MotherDuck	ArangoDB	CockroachDB

8. Results and Discussion

Examining the historical adoption trajectory of database technologies reveals a clear and instructive pattern of evolution driven by shifting application requirements [11]. SQL databases were among the earliest computerized data management tools, and their growth from the 1970s onward was both rapid and sustained [1]. The wide uptake of SQL platforms was driven by the combination of strong theoretical grounding, reliable consistency, and strong support for structured data. Platforms such as MySQL and

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3319-3325
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0426

PostgreSQL continue to represent a substantial share of production deployments, particularly in sectors where transactional accuracy and data integrity are paramount [2][4]. The emergence of big data and large-scale internet services beginning in the early 2000s created conditions in which NoSQL databases could thrive [7]. Adoption accelerated significantly after 2010, driven by the demand for systems capable of handling unstructured data at scale. MongoDB and Apache Cassandra gained particularly widespread use in real-time analytics and social platforms, where their throughput advantages and schema flexibility translated directly into operational and business value [8]. Their horizontal scaling model removed many of the infrastructure bottlenecks that had constrained relational systems in distributed environments. The NewSQL category has achieved a more gradual but steadily rising adoption curve since approximately 2010, reflecting the maturation of its underlying technologies and the growing recognition of its value proposition. By integrating distributed scalability with relational consistency and full ACID support, platforms such as Google Spanner and CockroachDB have demonstrated practical viability in high-concurrency enterprise environments. The overall picture that emerges from this analysis is one of complementarity: SQL retains its dominance in established enterprise domains, while NoSQL and NewSQL are redefining what is possible in distributed, high-throughput, and cloud-native application contexts [12].

Conclusion

This comparative study of SQL, NoSQL, and NewSQL database paradigms illustrates the remarkable adaptability of data management technology in response to evolving application demands [2]. Each paradigm occupies a distinct and valuable position within the broader database ecosystem. SQL databases, exemplified by MySQL and PostgreSQL, remain indispensable where data consistency and transactional reliability are essential, and they continue to underpin a vast range of critical enterprise systems [4]. NoSQL databases have expanded the boundaries of what data management systems can accommodate, enabling applications to handle diverse, large-scale data with the agility and throughput that modern digital services require [7]. Technologies such as MongoDB and Apache Cassandra have become foundational infrastructure for social media platforms, streaming analytics, and IoT systems where adaptability and scale are defining needs [8]. NewSQL databases represent the frontier of database engineering, synthesizing the proven reliability of the relational model with the distributed performance characteristics that cloud-era applications demand [9]. Systems like Google Spanner and CockroachDB demonstrate that it is possible to achieve both horizontal scalability and ACID compliance at scale, opening new possibilities for applications that previously faced difficult trade-offs. Ultimately, no single database paradigm is universally superior; the appropriate choice is determined by the specific consistency, scalability, and performance requirements of each application context. The trajectory of the field points toward continued innovation in flexible, scalable database solutions, with NewSQL likely to assume an increasingly prominent role as cloud-native architectures become the norm.

References

[1]. Edgar F. Codd, "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, vol. 13, no. 6, pp. 377-387, 1970.
[2]. Michael Stonebraker and U. Cetintemel, "One Size Fits All: An Idea Whose Time Has Come and Gone," Proceedings of the IEEE International Conference on Data Engineering, 2005.
[3]. Eric Brewer, "CAP Twelve Years Later: How the 'Rules' Have Changed," Computer, vol. 45, no. 2, pp. 23-29, 2012.
[4]. D. Abadi, "Consistency Tradeoffs in Modern Distributed Database System Design," IEEE Computer Society, 2012.
[5]. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Communications of the ACM, vol. 51, no. 1, pp. 107-113, 2008.
[6]. G. DeCandia et al., "Dynamo: Amazon's Highly Available Key-value Store," Proceedings of the ACM Symposium on Operating Systems Principles, 2007.
[7]. F. Chang et al., "Bigtable: A Distributed

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3319-3325
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0426

Storage System for Structured Data," ACM Transactions on Computer Systems, vol. 26, no. 2, 2008.

[8]. James Hamilton, "On Designing and Deploying Internet-Scale Services," Proceedings of the Large Installation System Administration Conference, 2007.

[9]. P. J. Sadalage and M. Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Addison-Wesley, 2013.

[10]. R. Cattell, "Scalable SQL and NoSQL Data Stores," ACM SIGMOD Record, vol. 39, no. 4, pp. 12-27, 2011.

[11]. M. Stonebraker et al., "VoltDB: A Main Memory OLTP Database System," Proceedings of the VLDB Endowment, 2010.

[12]. J. C. Corbett et al., "Spanner: Google's Globally Distributed Database," ACM Transactions on Computer Systems, vol. 31, no. 3, 2013.

[13]. MongoDB Inc., "MongoDB Architecture Guide," MongoDB Documentation, 2023.

[14]. Apache Cassandra Documentation, Apache Software Foundation, 2023.