A Survey of NoSQL Database Systems for Flexible and Scalable Data Management

www.surveyx.cn

Abstract

NoSQL databases have emerged as pivotal solutions in modern data management, addressing the limitations of traditional relational database systems through their ability to handle large volumes of unstructured and semi-structured data. This survey explores the significance, evolution, and application of NoSQL databases, categorizing them into document-oriented, key-value, column-family, and graph databases. Each type is analyzed for its data models and specific use cases, highlighting their applicability in diverse scenarios. The survey further examines the advantages of NoSQL systems, such as scalability and flexibility, while also addressing challenges related to consistency, security, and schema evolution. In distributed environments, NoSQL databases facilitate efficient data management and integration, overcoming the constraints of relational systems. The paper also presents real-world applications across various industries, showcasing the practical benefits of NoSQL solutions. Future trends and research directions are identified, emphasizing the need for continued innovation in database optimization, hybrid solutions, and schema management. This comprehensive exploration underscores the transformative role of NoSQL databases in contemporary data management, providing robust, scalable, and flexible solutions for complex, high-volume datasets.

1 Introduction

The rapid evolution of technology and the exponential growth of data have transformed the landscape of data management in recent years. Traditional relational databases, once the cornerstone of data storage and retrieval, face significant challenges in accommodating the diverse and dynamic nature of modern data. This review paper focuses on NoSQL databases, which have emerged as crucial solutions to address these challenges. By offering flexibility, scalability, and the ability to manage unstructured and semi-structured data, NoSQL databases are increasingly adopted across various sectors, including finance, healthcare, and e-commerce. This paper aims to provide a comprehensive overview of NoSQL databases, examining their importance, motivations for adoption, and the structure of the survey that follows.

1.1 Importance of NoSQL Databases in Modern Data Management

NoSQL databases have become indispensable in the contemporary landscape of data management, primarily due to their capacity to handle large volumes of unstructured and semi-structured data, which are prevalent in today's data-centric environments. These databases excel in managing the diverse and heterogeneous datasets produced by modern applications, such as social media platforms, where vast amounts of semi-structured data are generated [1]. The rapid expansion of data, often referred to as 'Big Data', presents significant challenges in storage, processing, and analysis, necessitating innovative solutions like NoSQL databases that integrate various aspects of data management, processing, analysis, and visualization [2].

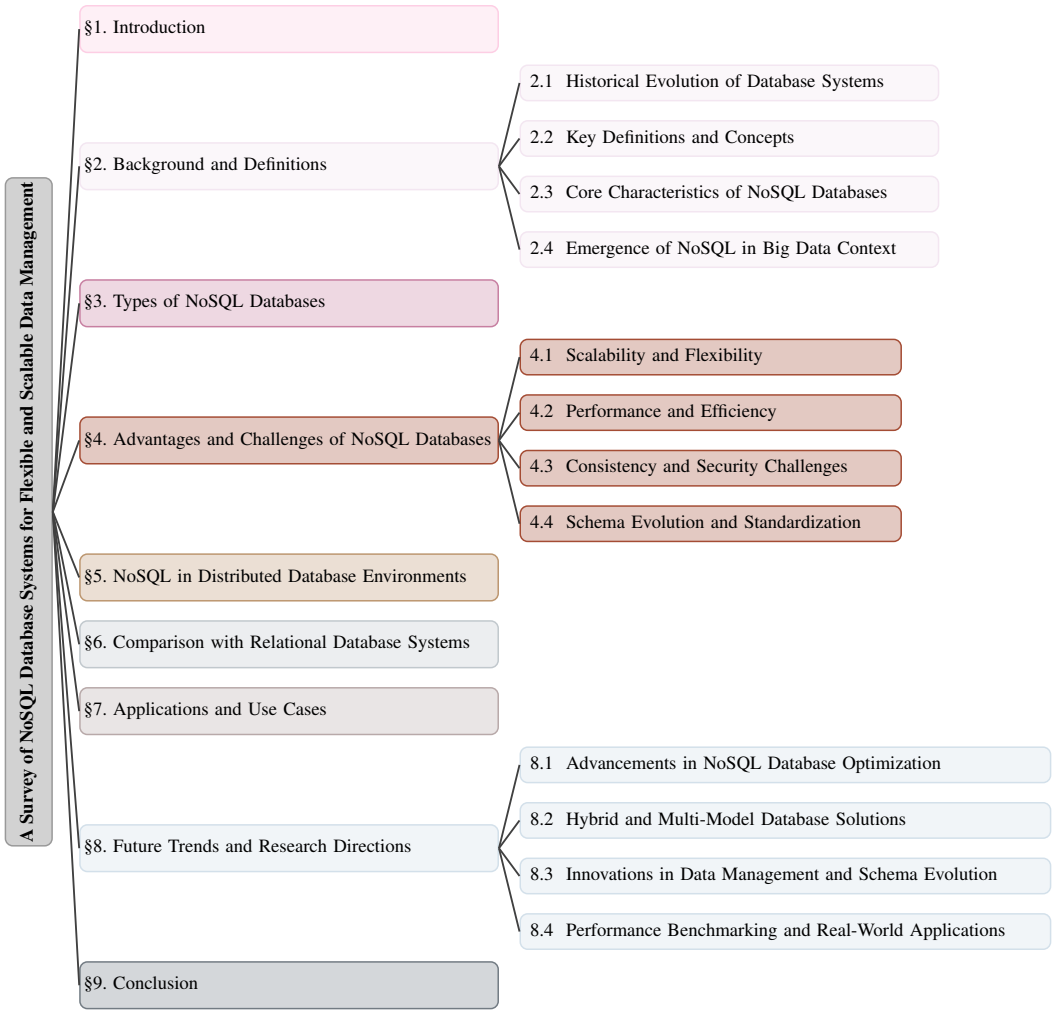


Figure 1: chapter structure

The scalability and flexibility of NoSQL databases are crucial in addressing the challenges posed by the influx of high-throughput technologies, such as those used in scientific research, which generate massive data volumes daily [3]. These capabilities are vital for fields that require efficient methods for data management and analysis. Furthermore, NoSQL databases are significant in handling large-scale, distributed systems that must adapt to changing requirements and coexist with legacy systems [4]. This adaptability is essential in dynamic environments where modern software systems are built to respond to changes and external uncertainties [5].

In the context of Internet of Things (IoT) data, which is characterized by large scale, volume, noise, and continuous nature, NoSQL databases provide significant advantages in managing such unstructured data [6]. The emergence of new NoSQL solutions tailored for Big Data applications has been instrumental in storing, managing, and extracting information from semi-structured data [7]. These solutions are crucial in sectors where traditional relational databases struggle to offer robust data management capabilities.

NoSQL systems have become the de facto back-end for modern applications, enabling unprecedented performance at large scale [8]. This performance is particularly important in global supply chains and logistics, where heterogeneous Logistics Information Systems face significant challenges that NoSQL databases are well-suited to address [9]. As such, NoSQL databases are essential tools for organizations seeking to leverage the full potential of their data assets in today's rapidly evolving data landscape.

AI-generated, for reference only.

The significance of NoSQL databases extends beyond mere performance metrics; they embody a paradigm shift in how data is conceptualized, stored, and utilized. Their ability to seamlessly integrate with various data sources and formats encourages innovation and enhances the agility of organizations in responding to market demands. This transformative potential positions NoSQL databases at the forefront of modern data management strategies, making them an essential focus for researchers and practitioners alike.

1.2 Motivation for Using NoSQL over Relational Databases

The shift from traditional relational database management systems (RDBMS) to NoSQL databases is propelled by the necessity to overcome the inherent limitations of RDBMS, particularly when addressing the complex demands of modern data environments characterized by high volume, variety, and velocity [10]. Relational databases often struggle with the rigid schema structures that are insufficient for the dynamic and heterogeneous data needs of contemporary applications, especially in the Big Data context [11]. NoSQL databases, with their inherent schema flexibility, provide a solution that accommodates evolving data structures, making them highly suitable for environments where data models must rapidly adapt to changing requirements [12].

In distributed systems, the limitations of SQL databases in managing large-scale data have led to the adoption of NoSQL solutions, which offer horizontal scaling and improved performance, albeit sometimes at the expense of stronger consistency models [13]. The ability of NoSQL databases to manage diverse application requirements and integrate multiple data models further underscores their preference over traditional systems [14]. The lack of interoperability among various data models and query languages in traditional systems emphasizes the motivation for using NoSQL databases [15].

Moreover, the demand for real-time data storage and retrieval in dynamic environments, such as the Internet of Things (IoT), further underscores the motivation for employing NoSQL databases [16]. These systems benefit from the capability of NoSQL databases to efficiently handle large datasets and persist data without a central authority, crucial in decentralized systems like peer-to-peer networks and Distributed Ledger Technologies (DLTs) [17]. The volatile and dynamic nature of IoT environments necessitates the use of NoSQL databases, which are better equipped to manage such data than traditional relational databases [6].

The challenges of scalability, availability, integration, and security in data storage solutions, compounded by the need for real-time data processing, further highlight the motivation for adopting NoSQL databases [18]. Effectively managing and analyzing large volumes of heterogeneous data generated from various sources in real-time, which conventional methods cannot handle, is a significant factor driving the adoption of NoSQL databases [2]. Traditional centralized security architectures in the data storage space face limitations in reliability, scalability, and operating costs, necessitating a new approach [19]. NoSQL databases offer solutions that address these challenges, providing flexible, efficient, and scalable data management in complex and large-scale environments.

Furthermore, the integration of SQL's rich features with the scalability of NoSQL systems is a critical consideration for Web applications requiring both functionalities. The development of systems like Yesquel aims to bridge this gap, offering scalable SQL storage solutions that combine the benefits of both paradigms [20]. This integration reflects the ongoing evolution of data management practices, where the strengths of both relational and NoSQL databases are leveraged to meet the diverse needs of modern applications.

The motivations for adopting NoSQL databases are not solely based on technical advantages; they also encompass strategic business considerations. Organizations that embrace NoSQL solutions position themselves to adapt swiftly to changing market dynamics, thereby enhancing their competitive edge. By leveraging the unique strengths of NoSQL databases, businesses can harness the full potential of their data assets while ensuring that their infrastructure remains agile and responsive to future challenges.

1.3 Structure of the Survey

This survey systematically explores the landscape of NoSQL database systems, addressing their significance, evolution, and application in modern data management. The paper is organized into several sections, each focusing on distinct aspects of NoSQL databases. The introduction sets the

stage by highlighting the importance of NoSQL databases and the motivation for their use over traditional relational databases. Following this, the background and definitions section provides a historical overview of database systems, leading to the emergence of NoSQL databases. It includes key definitions and core characteristics that distinguish NoSQL from relational databases.

The survey progresses to examine the various types of NoSQL databases, such as document-oriented, key-value stores, column-family stores, and graph databases. Each type is explored in terms of data models and specific use cases, providing a comprehensive understanding of their applicability in different scenarios [21]. This is followed by an analysis of the advantages and challenges associated with NoSQL databases, including scalability, flexibility, performance, and issues related to consistency and security.

Subsequent sections delve into the utilization of NoSQL databases in distributed environments, comparing them with relational database systems and highlighting integration, security, and privacy considerations. The survey also presents real-world applications and use cases across various industries, showcasing the practical benefits of NoSQL solutions. Finally, the paper discusses future trends and research directions in NoSQL databases, identifying areas for innovation and development. This structured approach ensures a thorough exploration of NoSQL databases, facilitating a deeper understanding of their role in contemporary data management.

Through this comprehensive survey, readers will gain insights into the current state of NoSQL databases, the rationale behind their adoption, and the challenges that remain. By synthesizing existing literature and highlighting key trends, this paper aims to contribute to the ongoing discourse surrounding data management technologies and their implications for future research and practice.The following sections are organized as shown in Figure 1.

2 Background and Definitions

2.1 Historical Evolution of Database Systems

The evolution of database systems reflects the changing demands of data management, transitioning from simple record-keeping to sophisticated, scalable solutions that accommodate diverse data types. Initially, relational databases dominated due to their efficacy in managing structured, tabular data. However, as data complexity grew, particularly with the advent of High-Performance Computing (HPC) systems, the limitations of rigid relational schemas became apparent, necessitating more adaptable solutions [22]. The increasing prevalence of heterogeneous data generated by modern applications exposed the inadequacies of traditional relational databases in handling unstructured and semi-structured data, prompting the development of NoSQL databases [23]. These databases introduced flexible data models that dynamically adapt to changing data structures, providing scalable solutions for high-volume, high-variety, and high-velocity data environments.

The rise of cloud computing further accelerated NoSQL adoption, offering scalable, efficient management of large datasets across distributed environments, adeptly handling geo-localized data and high concurrency demands typical of cloud applications. Traditional relational DBMS energy efficiency concerns under varying workloads highlighted the need for adaptable systems optimizing query performance [22]. As cloud infrastructure migration increased, robust data management solutions capable of seamless platform integration became crucial. The emergence of NewSQL databases, combining NoSQL scalability with the ACID properties of relational databases, reflects ongoing efforts to meet contemporary applications' diverse data management needs [23]. This interplay among database technologies continues to shape data management's future, as organizations optimize data strategies in complex digital environments.

The historical evolution of database systems underscores a continuous adaptation to the changing landscape of data management, with each technological advancement contributing to a nuanced understanding of data types and structures. This progression emphasizes the importance of ongoing research and development, as the demand for efficient, scalable data management solutions grows in response to the expanding volume and variety of data generated by modern applications.

2.2 Key Definitions and Concepts

NoSQL databases represent a significant shift from traditional relational systems, offering flexible, scalable solutions for modern data management needs. Categorized into key-value stores, document databases, wide-column stores, and graph databases, each type provides unique characteristics and applications. Key-value stores optimize rapid data retrieval environments [16], while document databases accommodate semi-structured formats like JSON, facilitating complex data representation [24]. Wide-column stores, suited for read-heavy operations, organize data into columns for enhanced performance [12]. Graph databases manage interconnected data, essential for querying intricate relationships.

The CAP theorem, a pivotal concept in NoSQL systems, posits that a distributed database can ensure only two of three properties: Consistency, Availability, and Partition tolerance [25]. This influences NoSQL architecture, favoring BASE (Basically Available, Soft state, Eventually consistent) properties over traditional ACID guarantees [26]. This focus on eventual consistency allows high availability and partition tolerance, crucial for distributed systems. NoSQL databases also address schema management and data discovery challenges, particularly with semi-structured data [27]. Their flexible schema evolution suits applications like IoT systems, where data formats constantly change [16].

In extensive networks, NoSQL databases enhance data management, providing insights into network structures and neighborhood information for efficient processing [28]. Integrating natural language processing capabilities improves user accessibility, enabling intuitive data interactions [29]. The adaptability of NoSQL systems, allowing real-time data processing and analysis, underscores their importance in dynamic environments [4]. These definitions and concepts highlight NoSQL databases' transformative role in data management, offering robust solutions for managing complex, high-volume datasets across various domains.

2.3 Core Characteristics of NoSQL Databases

NoSQL databases excel in managing large volumes of unstructured and semi-structured data through scalable, flexible, and efficient mechanisms. Supporting diverse data models like document-oriented, key-value, column-family, and graph databases, they offer unique operational benefits and adaptability to varied data structures. This flexibility is vital in dynamic environments, facilitating efficient data management and rapid query responses [16]. Document-oriented databases, such as MongoDB, utilize JSON or XML formats for enhanced data representation and manipulation [7], with JSON Schema managing subschema relationships [30]. Key-value stores optimize rapid data access, while column-family stores improve performance in read-heavy operations. Graph databases manage relationships using graph theory, enabling efficient querying of interconnected data, though interoperability challenges remain [31].

Adhering to the BASE model, NoSQL databases prioritize availability and partition tolerance over strict consistency, advantageous in distributed systems requiring high availability [9]. Their adaptability to various data models and operational mechanisms supports efficient query processing, enhancing utility in complex data environments [14]. The UDBMS framework exemplifies multi-model integration, enabling unified query processing and cross-model consistency [32], while innovations like Warp demonstrate NoSQL's capability for one-copy serializable ACID transactions using acyclic transactions [8]. Advanced indexing techniques, such as Wormhole, optimize data retrieval processes using trie, hash table, and B+ tree structures [33], while initiatives like ZipCache enhance performance through integrated data compression [34].

These core characteristics underscore NoSQL databases' transformative capability in effectively managing complex, high-volume datasets across various domains. Unlike traditional SQL databases optimized for structured data, NoSQL databases excel in handling unstructured data and provide horizontal scalability, making them well-suited for big data analytics [10, 35, 36, 37]. Their modularity, high availability, and performance optimization facilitate efficient workflows and interdisciplinary collaboration in modern applications.

2.4 Emergence of NoSQL in Big Data Context

NoSQL databases have become essential in big data management, driven by the need to efficiently handle the vast scale, diversity, and velocity of modern data demands. Traditional relational databases

often fall short in this context, unable to cope with dynamic, heterogeneous data requirements typical of big data environments [2]. The rise of NoSQL is attributed to their ability to manage diverse data types and models across various systems, offering a flexible alternative to rigid relational schemas [38]. This flexibility is crucial as data is voluminous and varied in format, necessitating innovative storage and retrieval approaches.

The deployment of advanced database management systems on cloud infrastructures highlights NoSQL's emergence, providing scalable, efficient data management capabilities for large datasets [39]. NoSQL databases balance consistency and availability, crucial for distributed environments where data spans multiple systems [6]. This capability is vital for real-time data processing, enabling organizations to leverage data-driven insights for enhanced decision-making. Graph data management, a subset of NoSQL technologies, has grown substantially, though widespread adoption is hindered by the lack of standardized query languages and inconsistent performance [40]. The LDBC Social Network Benchmark offers a framework for evaluating graph-like data management technologies, underscoring their performance and functionality in simulating social network operations [41].

The integration of data-centric techniques in the IoT domain parallels NoSQL's role in big data, emphasizing real-time processing and diverse data type handling [6]. Unified frameworks, such as datar, address challenges in big data management systems, integrating functionalities to enhance data management effectiveness [42]. These advancements reflect NoSQL technologies' ongoing evolution, adapting to an increasingly data-driven world.

The emergence of NoSQL in the big data context signifies a fundamental shift in data management approaches. By leveraging NoSQL's unique characteristics, organizations navigate big data complexities, unlocking new innovation and growth opportunities. This evolution enhances data management systems' capabilities and sets the stage for further advancements, as researchers and practitioners explore NoSQL technologies' potential in addressing modern data landscape challenges.

3 Types of NoSQL Databases

The diverse landscape of NoSQL databases reflects their evolution to address specific data management challenges, offering unique features tailored to various applications. This section explores document-oriented databases, highlighting their flexibility and adaptability, which are central to NoSQL systems. As illustrated in Figure 2, the hierarchical classification of NoSQL databases encompasses Document-Oriented, Key-Value, Column-Family, and Graph Databases, each characterized by distinct features and applications in various data-driven environments. Understanding these types elucidates their operational mechanisms and underscores their relevance in contemporary data-driven contexts, thereby aiding researchers and practitioners in selecting suitable database solutions.

3.1 Document-Oriented Databases

Document-oriented databases store data in semi-structured formats such as JSON, BSON, or XML, allowing for flexible schemas compared to traditional relational databases. This flexibility enables the storage of complex data structures within a single document, accommodating nested structures and various data types [43]. Such adaptability is crucial for applications requiring rapid iteration and schema evolution, as it allows developers to adjust data models without extensive migrations [43]. This capability is particularly advantageous in dynamic environments where data models must evolve with application needs.

A significant application of document-oriented databases is in e-commerce, where they manage product catalogs and customer data, facilitating personalized recommendations through complex data representations. For instance, a system leveraging NoSQL databases for C2C website recommendations uses machine learning to rank items based on descriptions, categories, and pricing [44]. Additionally, these databases are vital in content management systems, real-time analytics, and IoT applications, where handling diverse and semi-structured data is essential. Their ability to scale horizontally ensures efficient data management across distributed systems, making them integral to modern data solutions [45, 46, 7].

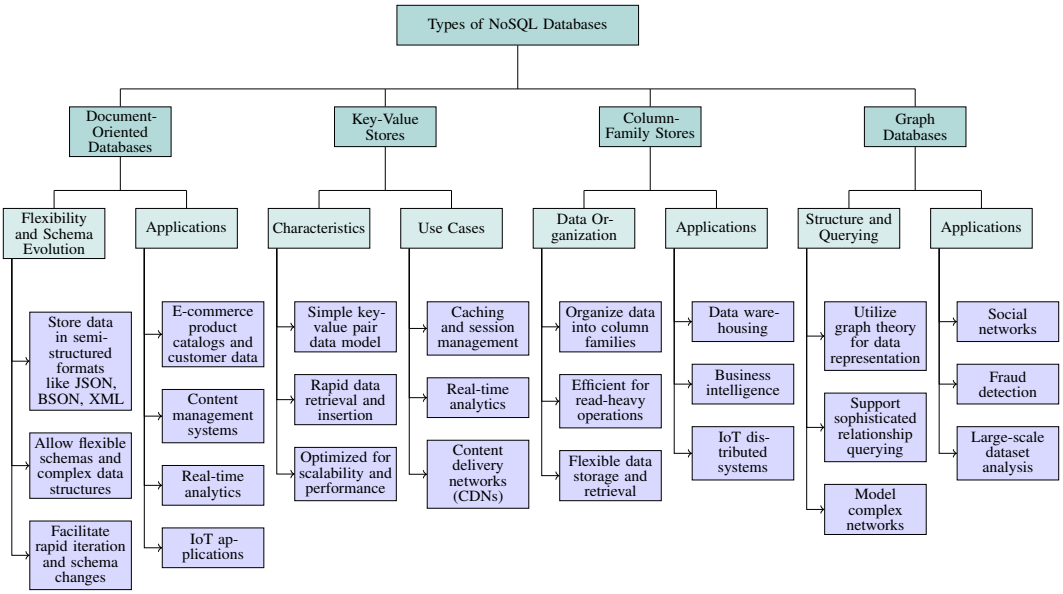


Figure 2: This figure illustrates the hierarchical classification of NoSQL databases, including Document-Oriented, Key-Value, Column-Family, and Graph Databases, highlighting their unique characteristics and applications in various data-driven environments.

3.2 Key-Value Stores

Key-value stores, characterized by their simple data model of key-value pairs, enable rapid data retrieval and insertion, ideal for applications demanding high-speed access and minimal latency [16]. This model supports efficient storage and retrieval, aligning with applications prioritizing performance over complex data relationships [14]. Optimized for scalability and performance, key-value stores employ distributed hashing to distribute data across nodes, making them suitable for caching, session management, and real-time analytics [9].

Their use is prevalent in environments requiring high-performance data processing, such as CDNs, where they cache web content to improve user experience. They also manage user sessions in web applications, where fast data access is critical. While key-value stores can support complex data types by storing serialized objects, their lack of advanced querying capabilities may limit their use in scenarios requiring intricate data interactions [18, 47, 15].

3.3 Column-Family Stores

Column-family stores, or column-oriented databases, organize data into column families, enhancing efficiency for read-heavy operations and large-scale data management. This model allows for flexible data storage and retrieval, focusing on specific data segments to enhance performance, particularly in analytics [14, 12]. Ideal for data warehousing and business intelligence, they enable efficient compression and data retrieval by accessing only relevant columns during queries [25].

Their architecture supports horizontal scaling, making them suitable for distributed systems like IoT, where data generation is continuous. This scalability ensures high availability and fault tolerance, vital for mission-critical applications [16]. Apache Cassandra exemplifies a column-family store, renowned for its scalability and performance in distributed environments [25, 12].

3.4 Graph Databases

Graph databases efficiently handle interconnected data by utilizing graph theory to represent entities as nodes and their relationships as edges, offering a natural way to model complex networks [48]. This structure supports sophisticated querying of relationships, crucial for applications like social networks and fraud detection. As data becomes more interconnected, graph databases' ability to traverse relationships quickly is essential, providing insights into large-scale datasets [49].

Frameworks such as Gradoop, built on the Hadoop ecosystem, demonstrate the scalability and analytical capabilities of graph databases, managing and analyzing graph data at scale [50]. As organizations increasingly rely on interconnected data for strategic insights, graph databases continue to drive innovation in data analysis and management.

4 Advantages and Challenges of NoSQL Databases

In contemporary data management, NoSQL databases are pivotal in addressing scalability and flexibility demands. These systems accommodate rapid data growth and versatile structures, facilitating adaptive storage and retrieval. The following subsection explores NoSQL databases' scalability and flexibility, highlighting architectural advantages and implications for modern applications.

4.1 Scalability and Flexibility

NoSQL databases are renowned for scalability and flexibility, essential for modern data environments. Architected for horizontal scalability, they distribute data across nodes, balancing loads and enhancing fault tolerance, thus supporting high performance even with increasing data volumes [38]. Their distributed nature suits real-time processing, especially in IoT environments requiring efficient storage and retrieval [16]. This scalability allows cost-effective infrastructure growth, accommodating workloads without downtime.

The schema-less architecture enhances flexibility, storing heterogeneous data without predefined schemas [7]. This adaptability is crucial for dynamic data models, minimizing overhead during schema evolution and enhancing interoperability [15]. Handling diverse formats like JSON or XML promotes innovation and rapid development, enabling swift responses to changing business needs.

In distributed settings, NoSQL databases reduce communication overhead and improve query execution, especially for complex queries [3]. Their flexibility in data analysis manages large datasets efficiently [41]. BigDAWG exemplifies performance improvements through efficient complex query management across models, showcasing NoSQL systems' scalability and flexibility [38]. Integration with data processing frameworks broadens applicability across sectors.

NoSQL databases enforce data integrity in complex structures through robust frameworks [40]. Warp's architecture maintains high throughput and concurrency without centralized transaction managers, highlighting scalability advantages [8]. Wormhole's indexing offers efficient lookup costs, handling large datasets effectively [33]. These features enhance performance, providing a foundation for reliable applications handling large data volumes.

Systems like Yesquel integrate SQL features with NoSQL scalability, avoiding vendor lock-in and offering flexibility [20]. Innovations like ZipCache improve cache hit ratios through compression and indexing, enhancing retrieval performance [34]. Frameworks like Biggy illustrate intelligent big data management, underscoring NoSQL databases' evolution as essential organizational tools for competitive advantage.

NoSQL databases offer robust, scalable, and flexible data management solutions, enabling efficient handling of complex datasets across applications. Their adaptability to evolving requirements and seamless system integration ensures continued relevance in modern data environments [39].

4.2 Performance and Efficiency

NoSQL databases excel in performance and efficiency, crucial for high-throughput, low-latency environments. Couchbase and MongoDB demonstrate superior write and read performance, essential for applications where response time is critical [51]. Wormhole indexing significantly improves lookup throughput, vital for rapid data access and processing [33]. Quick data retrieval enhances user experience and enables timely decisions based on real-time analytics, improving operational agility.

In graph data management, NoSQL databases address challenges in validating Graph Generating Dependencies, enabling efficient query and analysis of interconnected data [52]. This capability is vital for applications like social networks and recommendation systems, allowing complex graph traversals and analyses, driving innovation across fields.

NoSQL databases offer robust performance and efficiency, supporting various applications from graph management to real-time analytics and complex query processing. Their adaptability to workloads and data processing optimization positions them as attractive options for organizations seeking enhanced data management capabilities, aligning with evolving systems leveraging AI and machine learning for efficiency and accuracy [53, 54].

4.3 Consistency and Security Challenges

NoSQL databases face challenges in data consistency and security. The CAP theorem limits distributed systems to two of Consistency, Availability, and Partition tolerance, often leading to eventual consistency models unsuitable for all applications [55]. This trade-off can affect data integrity and user trust, crucial in finance and healthcare sectors [14].

Security vulnerabilities, especially in cloud-based environments, arise from lacking built-in features like encryption and access control [9]. The distributed nature adds complexity, risking performance and security breaches. Transitioning from SQL to NoSQL often lacks standards, complicating model selection and interoperability, impacting performance and security [56]. Addressing these vulnerabilities is essential for fostering trust in NoSQL systems.

Performance issues also arise with large datasets and complex queries, such as in graph databases. Frameworks like IGAG face limitations in Gremlin language coverage, focusing only on pattern matching without mutating operations [31]. This highlights the need for efficient query processing to maintain performance and consistency [2]. Inaccuracies in relational databases can propagate to NoSQL systems, affecting data quality [4]. Robust validation and error-handling mechanisms are crucial to mitigate data discrepancies.

Addressing consistency and security challenges requires standardized protocols, enhanced security measures, and improved resource management. Such improvements optimize NoSQL databases' performance and reliability, ensuring effectiveness in modern data environments [8]. Evolving security frameworks and best practices will mitigate vulnerabilities, enhancing NoSQL systems' robustness.

NoSQL systems also face challenges in data quality and uncertainty, especially in IoT data management, where continuous data influx can lead to inconsistencies and vulnerabilities [6]. Systems like Yesquel, limited to single data centers, may struggle in geo-distributed environments [20]. Approaches like ZipCache introduce complexity in managing B+ tree structures and compression overheads [34]. Frameworks like Biggy, being standalone, may not fully utilize distributed system benefits, impacting consistency and security effectiveness [42].

4.4 Schema Evolution and Standardization

Schema evolution and standardization pose challenges due to NoSQL databases' flexible, diverse data models. Effective schema change management is vital, especially in agile contexts where models evolve with application code. Despite flexibility advantages, lack of standardized practices can lead to inconsistencies, particularly when integrating diverse models and query languages [57]. Custom migration scripts are error-prone and costly, complicating maintenance and highlighting the need for principled tool support. Establishing best practices for schema evolution is crucial for data quality and efficiency.

SkiQL addresses query language limitations by supporting entity variations and relationships, facilitating schema evolution while maintaining consistency [27]. However, dependency on existing tools can restrict adaptability to new requirements. Inconsistent protection across NoSQL systems can compromise data integrity, particularly in high-frequency update environments [58].

The Shapley value's computational complexity poses challenges, often requiring approximations that may not yield precise results [59]. Innovative approaches are needed for efficient schema evolution management. The UDBMS framework exemplifies challenges in cross-model query optimization and ensuring global consistency [32]. Advancing such frameworks is critical for leveraging NoSQL databases' potential while minimizing schema evolution risks.

Addressing these challenges requires adaptable frameworks integrating security and supporting schema evolution across diverse NoSQL implementations. Lack of comprehensive benchmarking and

exploration of hybrid models complicates efforts [18]. Advancing schema evolution and standardization enhances data management capabilities, ensuring reliability and consistency across applications. Standardized practices and tools empower organizations to navigate NoSQL databases' complexities, fostering a robust, secure data management landscape.

5 NoSQL in Distributed Database Environments

Category	Feature	Method
Utilization of NoSQL Databases in Distributed Systems	Data Visualization and Management	ICDM[19], QCSA[16]
	Storage and Resource Optimization	TSC[60]
	Transaction Efficiency	WARP[8]
	Query and Integration Techniques	N/A[20]
Data Consistency and Availability Challenges	Performance Impact Assessment	DCST[61]
	Consistency Evaluation	QoD[62]
Performance Optimization Techniques	Resource and Workload Management	ARM-MN[63]
	Query Execution Enhancement	BD[38]
	Data Retrieval Optimization	WH[33]

Table 1: This table presents a comprehensive overview of various methodologies employed in distributed systems utilizing NoSQL databases, focusing on their application categories, specific features, and implementation methods. It highlights innovative solutions for data visualization, storage optimization, transaction efficiency, and performance enhancement, alongside addressing challenges in data consistency and availability. The table serves as a valuable resource for understanding the diverse functionalities and integrations of NoSQL technologies in modern distributed environments.

The adoption of NoSQL technologies in distributed database environments has transformed large-scale data management, offering distinct advantages over traditional relational databases. As we explore the applications of NoSQL databases within these settings, their versatility and scalability emerge as key factors in addressing data management complexities. Table 1 provides a detailed summary of the methodologies applied in distributed systems using NoSQL databases, showcasing their diverse capabilities and addressing key challenges in data management and performance optimization. Additionally, Table 3 offers a detailed comparison of methodologies applied in distributed systems utilizing NoSQL databases, focusing on scalability, data model flexibility, and performance optimization strategies. The following subsection delves into the utilization of NoSQL databases in distributed systems, emphasizing their capabilities and innovative solutions for modern data challenges.

5.1 Utilization of NoSQL Databases in Distributed Systems

Method Name	Scalability Features	Data Model Flexibility	Integration Capabilities
QCSA[16]	Real-time Data	Flexible Data Structure	Amazon Web Services
TSC[60]	Data Migration Strategies	Hierarchical Storage Scheduler	Workload Forecasting Integration
BD[38]	Multiple Nodes	Various Data Models	Integrates Various Systems
ICDM[19]	Enhanced Data Reliability	Decentralized Data Management	Integrating Blockchain
ARM-MN[63]	9-node Cassandra	Various Data Models	Other Nosql Systems
WARP[8]	High Concurrency	Multi-key Transactions	Language Bindings
N/A[20]	Distributed Balanced Tree	Transactional Key-value Storage	Embedded Query Processor

Table 2: Comparison of NoSQL Database Methods in Distributed Systems: This table provides an overview of various NoSQL database methods, highlighting their scalability features, data model flexibility, and integration capabilities. The methods listed include QCSA, TSC, BD, ICDM, ARM-MN, WARP, and an unnamed method, each demonstrating unique strengths in handling large-scale data across distributed environments.

NoSQL databases are essential to distributed computing environments, providing scalable and efficient solutions for managing large-scale data across multiple nodes. Their implementation is characterized by the ability to facilitate parallel computation and concurrent communication, crucial for handling substantial data volumes. This is achieved through strategies such as leveraging multiple processing cores to parallelize computation and communication, enabling simultaneous data transfers and minimizing synchronization barriers. Frameworks like Node Scala exemplify efficient request handling in horizontally-scalable architectures, demonstrating performance improvements up to 74% over traditional methods and enhancing overall throughput [64, 65]. NoSQL databases' flexibility allows adaptation to various data models, including key-value, column-oriented, document, and graph databases, each offering unique advantages and security challenges in distributed settings. Table 2

AI-generated, for reference only.

presents a comparative analysis of different NoSQL database methods, emphasizing their scalability, flexibility, and integration within distributed systems.

In distributed environments, NoSQL databases enable real-time data collection and visualization, as seen in systems like the Quality Check System Architecture (QCSA), which manages time-series data using a NoSQL database [16]. Hierarchical storage capabilities in systems like TS-Cabinet enhance efficiency by optimizing data storage locations across cloud, edge, and end devices, crucial for applications requiring dynamic resource allocation and efficient resource utilization [60].

The integration of NoSQL databases with other technologies is evident in frameworks like BigDAWG, enabling efficient querying and analytics on heterogeneous datasets in distributed environments [38]. This integration is vital for applications requiring complex data processing and analytics, allowing seamless interoperability with existing frameworks. Additionally, blockchain technology in the ICDM model provides a decentralized framework for storing encrypted communication data and logs, ensuring secure data retrieval in distributed systems [19].

NoSQL databases excel in handling dynamic resource allocation in multi-tenant environments. Techniques like resource sharing and tenant workload-based scheduling ensure efficient resource utilization, maintaining high performance and scalability in distributed settings, essential for managing diverse and evolving data requirements [63].

Transactional capabilities are exemplified by systems like Warp, enabling efficient multi-key transactions with high concurrency on distributed, sharded data stores [8]. Similarly, Yesquel maps SQL queries to operations on a distributed balanced tree, highlighting the integration of SQL features with NoSQL scalability [20]. The diverse functionalities and integrations of NoSQL databases underscore their critical role in modern distributed systems, paving the way for innovative solutions to complex data challenges.

5.2 Data Consistency and Availability Challenges

Maintaining data consistency and availability in distributed NoSQL systems presents significant challenges, particularly in high concurrency and network partition environments. The CAP theorem asserts that a distributed database can only ensure two of the three properties: Consistency, Availability, and Partition tolerance. Consequently, NoSQL systems often prioritize availability and partition tolerance, adopting eventual consistency models that may not suit all applications [62]. This trade-off can lead to discrepancies between expected and observed consistency levels, especially in geo-replicated cloud data stores where network latency and partitions complicate consistency maintenance [62].

Experiments using the Data Consistency Simulation Tool (DCST) on systems like Cassandra and MongoDB demonstrate variability in consistency and performance across different workloads and network configurations [61]. These experiments highlight the challenges in achieving consistent data states while maintaining high availability, particularly in distributed environments with data replicated across multiple nodes. The trade-offs between consistency and performance necessitate careful consideration of application requirements and network conditions to optimize NoSQL configurations [61].

Moreover, the lack of standardized protocols for ensuring data consistency across diverse NoSQL implementations complicates these challenges. Integrating different data models and query languages can lead to inconsistencies, particularly when synchronizing data across distributed systems. Addressing these challenges requires innovative resource management approaches, including dynamic resource allocation and efficient workload scheduling, to maintain high availability while minimizing consistency discrepancies [62]. Ongoing research and development are needed to establish robust frameworks enhancing consistency and availability in NoSQL environments.

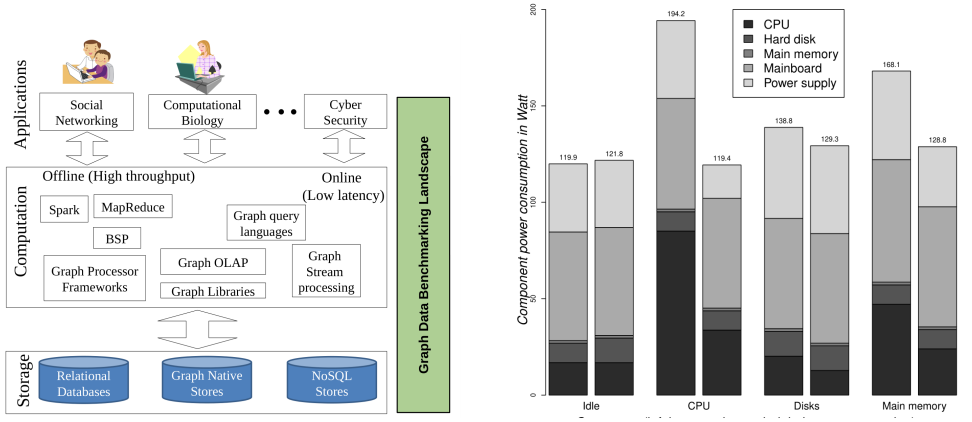
5.3 Performance Optimization Techniques

Optimizing performance in distributed NoSQL databases is crucial for efficient large-scale data management and ensuring rapid data access and processing. Advanced indexing methods, such as Wormhole, integrate trie, hash table, and B+ tree structures to enhance lookup, insertion, deletion, and range query operations, significantly improving data retrieval processes and offering faster throughput compared to traditional indexing methods [33].

Dynamic resource allocation and workload scheduling strategies are critical for maintaining high performance in multi-tenant environments, where efficient resource utilization is necessary to handle varying workloads and ensure consistent data access across distributed systems [63]. By dynamically adjusting resources based on tenant demands and workload characteristics, NoSQL databases achieve better scalability and performance.

Caching mechanisms, such as ZipCache, play a vital role in performance optimization. ZipCache employs a hybrid DRAM-SSD architecture with data compression within the cache design, enhancing cache hit ratios and reducing latency in data retrieval [34]. This technique is particularly effective in high-frequency data access environments, where minimizing latency is critical for maintaining system responsiveness.

Leveraging distributed computing frameworks like BigDAWG facilitates efficient querying and analytics on heterogeneous datasets, enabling seamless interoperability with existing data processing frameworks and supporting complex data processing and analytics in distributed environments [38]. Optimizing query execution across different data models and systems is essential for enhancing NoSQL databases' overall performance. The diverse strategies employed in performance optimization reflect the dynamic nature of data management in distributed systems, underscoring the importance of continuous innovation in this field.



(a) Graph Data Benchmarking Landscape[66] (b) Component Power Consumption in Watt[67]

Figure 3: Examples of Performance Optimization Techniques

As shown in Figure 3, optimizing performance is crucial in distributed database environments, particularly with NoSQL systems handling vast amounts of unstructured data across multiple nodes. The example provided delves into performance optimization techniques, illustrated through two distinct images. The first image, "Graph Data Benchmarking Landscape," offers a comprehensive overview of tools and technologies in graph data processing, categorizing applications into areas like Social Networking, Computational Biology, and Cyber Security, and distinguishing between offline and online computational modes, each utilizing specific tools like Spark and graph query. This landscape provides insights into how different components interact to enhance data processing efficiency. The second image, "Component Power Consumption in Watt," presents a bar chart highlighting energy usage across computer system components, underscoring energy efficiency's importance in optimizing system performance. Together, these examples underscore the multifaceted approach required for performance optimization in NoSQL distributed databases, balancing computational efficiency with energy considerations.

6 Comparison with Relational Database Systems

The evolution of database systems has introduced NoSQL databases as a viable alternative to traditional relational databases, offering distinct advantages in scalability and flexibility. This section examines the integration of NoSQL databases with other technologies, highlighting their ability to interact seamlessly with existing infrastructures. The rise of big data and diverse data types necessitates a shift towards solutions that accommodate these complexities. A comparative analysis

AI-generated, for reference only.

Feature	Utilization of NoSQL Databases in Distributed Systems	Data Consistency and Availability Challenges	Performance Optimization Techniques
Scalability	Horizontally-scalable Architectures	Eventual Consistency Model	Efficient Resource Utilization
Data Model Flexibility	Key-value, Column-oriented	Not Specified	Heterogeneous Datasets
Performance Optimization	Parallel Computation	Dynamic Resource Allocation	Advanced Indexing Methods

Table 3: This table provides a comparative analysis of various methodologies employed in distributed systems using NoSQL databases. It highlights key features such as scalability, data model flexibility, and performance optimization techniques, emphasizing the challenges of data consistency and availability. The table serves as a comprehensive overview of the diverse capabilities and strategies for optimizing performance in distributed NoSQL environments.

of NoSQL and relational databases elucidates their respective strengths and weaknesses in modern data environments.

6.1 Integration with Other Technologies

The integration of NoSQL databases with existing systems is pivotal for modern data management, enabling organizations to leverage both NoSQL and relational database strengths. Frameworks like the RAL Framework illustrate this by supporting SQL and NoSQL databases, facilitating interoperability across diverse data models and storage solutions [68]. This dual support is essential for applications requiring NoSQL's flexibility alongside SQL's structured querying capabilities, allowing organizations to tailor data strategies to specific use cases, thereby enhancing efficiency and performance.

The ingestion of relational databases into unified NoSQL Data Warehouses underscores integration's role in improving data accessibility and management [69]. By merging relational data into NoSQL systems, organizations gain a comprehensive view of their data assets, enabling advanced analytics and informed decision-making. This process often involves data transformation and migration tools to ensure consistency and integrity across database environments, streamlining workflows and enhancing real-time analytics capabilities vital in today's fast-paced business landscape.

NoSQL databases also integrate with data analytics frameworks, such as Hadoop and Spark, to process and analyze large-scale datasets efficiently. This interoperability is crucial for applications requiring real-time data analysis, facilitating seamless data exchange across ecosystem components and enhancing overall data management and analysis scalability [70, 2, 54, 15]. The synergy between NoSQL and big data technologies not only optimizes performance but also fosters innovation by enabling insights from vast unstructured data.

The integration of NoSQL databases with existing technologies is a strategic advantage for organizations aiming to remain competitive in a data-driven world. By leveraging both NoSQL and relational databases, businesses can create agile data architectures aligning with operational goals. As organizations navigate modern data ecosystems' complexities, effective integration strategies will become increasingly important, paving the way for sophisticated data management solutions.

6.2 Security and Privacy Considerations

Security and privacy in NoSQL databases pose unique challenges due to architectural differences and the lack of standardized security protocols. Research shows NoSQL databases, like MongoDB, have higher vulnerability compared to SQL databases, with significant breaches reported [71]. This vulnerability is often due to the absence of built-in security features, such as data encryption and access control, more common in relational databases. These vulnerabilities can lead to financial losses and reputational damage for organizations failing to protect their data assets adequately.

The distributed nature of NoSQL systems complicates security, as resource contention among tenants can lead to unpredictable performance and security breaches. Effective hashing algorithms, such as AnchorHash and Maglev, are critical for optimizing performance and security in distributed databases [72]. These algorithms enhance NoSQL systems' resilience by efficiently distributing workloads across nodes, minimizing data exposure and unauthorized access risks. Organizations must prioritize robust load balancing techniques to strengthen their NoSQL environments' security posture.

Furthermore, integrating microservices architectures with NoSQL databases requires robust testing tools to evaluate database client resilience under unreliable conditions, essential for maintaining secure applications [73]. These tools allow developers to simulate potential security threats and

assess impacts on database performance and integrity, facilitating proactive data privacy measures. By adopting a proactive security approach, organizations can better prepare for emerging threats and ensure NoSQL implementations remain resilient against evolving cybersecurity challenges.

Addressing security and privacy in NoSQL databases requires a multifaceted approach, including advanced load balancing, rigorous testing frameworks, and developing standardized security protocols. Implementing robust security measures is essential for NoSQL systems across various domains, safeguarding against cybersecurity threats and enhancing data asset management and protection, particularly in high-volume, heterogeneous information environments like IoT and cloud-based applications [74, 71, 36, 1, 75]. As data management evolves, addressing these security challenges remains paramount, ensuring organizations can confidently leverage NoSQL technologies while protecting sensitive information.

7 Applications and Use Cases

NoSQL databases have become indispensable in modern data management, offering versatile solutions across diverse domains. This section explores specific use cases demonstrating NoSQL's adaptability in addressing industry-specific challenges. Initially, the focus is on managing Internet of Things (IoT) and time-series data, emphasizing NoSQL's capacity to handle the complexities and volume of data from connected devices.

7.1 IoT and Time-Series Data Management

The exponential growth of IoT and time-series data necessitates scalable solutions, with NoSQL databases providing the required flexibility. The emergence of 5G technology enhances IoT applications, demanding robust systems for managing increased data flows [76]. NoSQL databases efficiently support spatial and temporal queries, crucial for real-time analysis in IoT applications [77]. This capability allows organizations to extract actionable insights, improving operational efficiency and decision-making [78]. Additionally, NoSQL facilitates user interaction in scientific research, exemplified by BioSmart's natural language querying [79]. Practical applications include managing user-generated content and enhancing data retrieval efficiency through languages like Mongolog [80, 81]. Systems like HBase-QoD demonstrate NoSQL's potential in cloud environments, offering tailored consistency levels for IoT applications [62].

7.2 E-commerce and Personalized Recommendations

NoSQL databases are integral to e-commerce, enabling personalized recommendations and enhancing user experience. Their flexibility and scalability support vast datasets typical of e-commerce environments, facilitating efficient storage and retrieval of user profiles and transaction histories. By integrating machine learning, platforms can enhance recommendation accuracy [82, 83]. NoSQL's scalability aids in processing large user data volumes, improving performance in C2C recommendation systems [44]. Graph-based systems further enhance recommendations by modeling complex user-product relationships [84]. Real-time data analysis optimizes inventory management and marketing strategies, ensuring competitiveness in dynamic markets.

7.3 Industrial and Asset Management

In industrial and asset management, NoSQL databases offer robust solutions for handling complex data environments. They integrate data from sensors and logs, supporting predictive maintenance and real-time analytics, thus reducing downtime and extending asset lifecycles [16]. NoSQL's scalability aids supply chain optimization by managing logistics data, enhancing decision-making and operational agility [9]. Their ability to track asset utilization and performance supports strategic planning and asset optimization [14].

7.4 Social Media and Sentiment Analysis

NoSQL databases are pivotal in social media and sentiment analysis, efficiently managing unstructured data from platforms like Twitter. They support real-time processing, enabling insights into user sentiment and engagement patterns, crucial for marketing strategies [1]. Integration with machine

learning algorithms enhances sentiment analysis, allowing for nuanced sentiment classification and prediction [44]. NoSQL's scalability supports large dataset processing, crucial for responsive marketing strategies. Visualization tools facilitate data-driven decision-making by presenting insights in accessible formats [2].

7.5 Healthcare and Smart City Management

NoSQL databases revolutionize healthcare and smart city management by handling complex, large datasets. In healthcare, they integrate heterogeneous data sources for personalized medicine and population health management, supporting schema changes without significant overhead [85]. In smart cities, NoSQL supports data collection and analysis for urban planning and service delivery, enhancing resource allocation and public safety [86]. Integration with IoT technologies facilitates adaptive urban infrastructure, addressing challenges like energy management and environmental monitoring [36, 77].

7.6 High-Performance Computing and Data Visualization

In high-performance computing (HPC) and data visualization, NoSQL databases manage vast datasets efficiently. Their scalability supports complex data requirements in scientific research, enabling large-scale data analysis [18, 87]. NoSQL systems facilitate real-time data visualization, enhancing decision-making through interactive tools [88]. Integration with vector similarity search methods underscores their relevance in supporting advanced data analysis [83]. Experiments on graph datasets validate NoSQL's efficacy in handling large-scale data challenges, driving innovation across research fields [82].

8 Future Trends and Research Directions

The rapidly evolving database technology landscape necessitates an understanding of future trends and research directions, especially within NoSQL databases, which are pivotal in addressing the complexities of modern data environments where traditional relational databases often falter. This section identifies key advancements and innovations that are shaping data management's future, highlighting essential research opportunities aimed at enhancing performance, scalability, and integration capabilities. The following subsection focuses on specific advancements in NoSQL database optimization, providing a comprehensive overview of critical developments influencing future data management strategies.

8.1 Advancements in NoSQL Database Optimization

Recent advancements in NoSQL database optimization emphasize improving performance, scalability, and integration to meet the demands of modern data environments. Hybrid systems that merge SQL and NoSQL strengths offer unified data management approaches adaptable to diverse applications [36, 37]. These systems combine SQL's robust transaction management with NoSQL's scalability and flexibility, addressing individual system limitations and enhancing user experience across various data paradigms.

Optimizing transaction processing methods remains crucial for distributed NoSQL systems' efficiency and reliability. Future research should explore further transaction processing optimizations and scalability in systems like Warp [8]. Advancements in RCC design and its application in diverse consensus protocols also present promising optimization avenues [19], broadening NoSQL's applicability across sectors like finance, healthcare, and IoT.

Research on Log-Structured Merge-trees (LSM-trees) focuses on performance stability and benchmark refinement to accommodate diverse workloads [25]. These efforts ensure NoSQL databases efficiently handle varied scenarios, from high-throughput environments to complex query processing tasks [11]. Enhancing LSM-trees supports real-time data processing needs in applications like online analytics and social media.

Benchmarking frameworks are vital for optimizing NoSQL databases, yet comprehensive benchmarks across diverse workloads are lacking. Developing standardized benchmarking frameworks will address this gap, offering insights into performance and cost metrics for big data applications

[89]. Existing benchmarks often fail to capture real-world challenges, particularly in event-driven architectures [90]. Establishing robust benchmarks will guide database selection and optimization strategies.

Query processing optimization is pursued by expanding frameworks to support additional query languages and conducting extensive experimental evaluations across diverse scenarios [91]. These developments enhance NoSQL databases' versatility in handling complex queries and data interactions [75]. Optimizing validation algorithms for Graph Generating Dependencies (GGDs) and efficient graph pattern matching methods are key focus areas [11], enabling efficient management of interconnected data prevalent in social networks and recommendation systems.

Future directions include enhancing query planning capabilities, increasing supported islands and engines in frameworks like BigDAWG [38], improving OLAP query support on XML data [7], and exploring consistency mechanisms in hybrid clouds [39]. Expanding benchmarks to cover streaming data management and machine learning applications in graph contexts [40], optimizing annotation process data throughput [3], and enhancing predictive capabilities in quality check systems [16] are critical research areas. Addressing these challenges will significantly enhance NoSQL databases' effectiveness across diverse domains, paving the way for sophisticated data management solutions.

8.2 Hybrid and Multi-Model Database Solutions

Hybrid and multi-model database solutions represent significant advancements in data management, integrating NoSQL systems' scalability and flexibility with relational databases' structured querying capabilities. These solutions provide unified frameworks for managing diverse data models, facilitating seamless interoperability and enhanced querying across paradigms. Exploring multi-model databases is promising, with future research directions including cost models for migration strategies and improving schema evolution and data migration tools [57]. These advancements are crucial for organizations leveraging multiple database technologies in their practices.

Emerging hybrid database trends include integrating various protocols and enhancing scalability, particularly in IoT environments. The IoT Exchange exemplifies this by refining protocol integration and addressing data privacy regulatory challenges [92]. This integration is vital for hybrid databases' interoperability and functionality, accommodating complex data management scenarios. As organizations seek to harness their data assets' full potential, seamless integration of diverse data sources and protocols becomes increasingly important.

Benchmarking frameworks like the LDBC SNB offer modular and scalable approaches to assessing graph databases, targeting various functionalities [41]. Inspired by the Transaction Processing Performance Council (TPC), this approach aims to fulfill a similar role in graph data management [40]. Benchmarking is crucial for evaluating hybrid solutions' performance, ensuring efficient handling of diverse workloads and data types. Rigorous evaluation processes are essential for maintaining hybrid systems' reliability and performance in real-world applications.

Future research should focus on developing adaptive systems that seamlessly integrate diverse data stores and enhance query processing capabilities [14]. Incorporating machine learning techniques to adapt schemas in real-time advances hybrid database solutions, improving workload prediction accuracy and optimizing schema management. This approach benefits dynamic environments where data models must evolve rapidly to meet changing application requirements. Leveraging machine learning enhances hybrid databases' responsiveness and efficiency, ultimately improving user satisfaction and operational effectiveness.

Security considerations in hybrid database solutions are paramount, with research focusing on identifying new injection techniques specific to NoSQL databases. Addressing these security challenges ensures hybrid systems' reliability and integrity across diverse domains. Efficient algorithms for computing the Shapley value in database management present opportunities for optimizing resource allocation and enhancing decision-making processes [59]. Robust security measures are crucial for fostering trust in hybrid solutions, especially in sensitive applications like finance and healthcare.

Exploring hybrid and multi-model database solutions redefines data management practices, providing organizations with tools to manage increasingly complex environments. By integrating various database technologies' strengths, these solutions offer enhanced flexibility, scalability, and perfor-

mance, paving the way for innovative applications across sectors. Future research will continue uncovering opportunities and challenges in this evolving field.

8.3 Innovations in Data Management and Schema Evolution

Innovations in data management and schema evolution within NoSQL databases address modern applications' dynamic and heterogeneous data requirements. Enhancing automation features and expanding natural language processing (NLP) model support in systems like LiFE reflect trends towards more automated and intelligent data management practices. Automation is essential for managing schema evolution, allowing seamless adaptation to changing needs without substantial overhead. Future research should emphasize standardized management operations and integrating replication middleware with databases to address software upgrade complexities [93]. These advancements ensure NoSQL databases keep pace with rapid application requirements and user expectations evolution.

User-friendly visual tools for property graph schema extraction and refinement represent innovative schema management approaches. These tools facilitate complex data structures visualization and understanding, enabling intuitive schema interaction and refinement. Enhancements to suggestion algorithms, including better user feedback integration, could broaden these tools' application across various data graphs [94]. By improving user interaction with schema management, organizations empower users to derive greater value from their data assets.

Future research directions include optimizing schema design and expanding SPARQL features in distributed RDF data management systems, enhancing inference support and data integration capabilities [95]. Additionally, exploring alternative embedding techniques and optimizing degree grouping strategies in dynamic subgraph matching can enhance graph data management tasks' performance. Refining Monte Carlo approaches and exploring self-adaptive strategies for data migration enhance efficiency and compliance with service-level agreements [96]. These innovations ensure NoSQL databases effectively manage modern data environments' complexities.

Innovations in transaction management, such as specialized locking algorithms and structured approaches, differentiate these methods from existing solutions, offering enhanced performance and reliability in database access. Optimizing preprocessing stages and exploring merging algorithms' applications in big data analytics lead to more efficient data processing and management [97]. Future research should focus on integrating DRLISA with traditional index structures and emerging types to enhance index selection for SQL and NoSQL databases [98]. Advancing transaction management techniques significantly improves NoSQL databases' overall performance.

Refining benchmark techniques for query optimization and extending benchmarks to include additional NoSQL databases are crucial for evaluating and improving performance across diverse scenarios. Studying satisfiability and implication problems for Graph Generating Dependencies (GGDs) and developing inference rules provide insights into schema evolution and data integration challenges in graph data management [99]. Enhancing query execution engines for complex queries and exploring integration techniques for emerging storage solutions are promising research directions [100]. These advancements help practitioners navigate complex data environments effectively.

Future research will focus on refining the Quality of Data (QoD) model, exploring its application in different cloud infrastructures, and conducting experiments to validate its effectiveness across diverse scenarios [62]. Innovations in data management practices are demonstrated in systems like the inverted movement calculation method, enhancing NoSQL databases' capabilities in handling IoT data [101]. Exploring innovations in data management practices, particularly extending the ingestion process to include other data sources [69], further enhances NoSQL databases' adaptability and performance in evolving landscapes.

Enhancing metadata standards, developing new algorithms for data analysis, and creating sophisticated tools for interactive exploration of large datasets are crucial research areas [28]. Expanding datasets with additional knowledge bases and improving benchmarks to evaluate complex entity resolution scenarios could be beneficial [24]. Future improvements could include expanding databases by collaborating with more institutions and enhancing app functionalities based on user feedback [29]. Developing protected search systems supporting high data ingest rates and integrating cryptographic techniques into emerging database technologies are critical research areas [58].

Innovations in data management and schema evolution in NoSQL databases meet contemporary data environments' complex and dynamic needs. Exploring these research directions allows NoSQL systems to evolve, providing robust and adaptable data management strategies for modern applications. Future research may explore MongoDB's Raft protocol verification, improve models for concurrency, and investigate non-transactional operations' impact on consistency [102]. Additionally, future research could explore transfer learning applications to other non-functional properties, enhance sampling strategies, and investigate active learning techniques integration [5].

8.4 Performance Benchmarking and Real-World Applications

Benchmark	Size	Domain	Task Format	Metric
NoSQL-Benchmark[103]	30	Database Performance Evaluation	Performance Testing	Response Time, Throughput
NoSQL-Schema-Evo[104]	10	Nosql Database Schema Evolution	Schema Analysis	Schema-LoC
AsterixDB[105]	1,000,000	Social Media Analytics	Data Management	Response Time, Throughput
LBH[72]	10,000	Distributed Databases	Load Balancing	Uniformity, Moved Records
BDA-H[106]	3,000,000	Healthcare	Performance Evaluation	TPC-H
NoSQL-Benchmark[107]	80	Data Management	Feature Classification	Gini Importance
DDFL[108]	1,000,000	Image Classification	Image Classification	Accuracy, Training Time
-miner[109]	4,667,738	Incident Management	Process Discovery	Execution Time, Disk Usage

Table 4: The table presents a diverse selection of benchmarks used to evaluate NoSQL databases across various domains, including database performance evaluation, schema analysis, and incident management. It details the size, domain, task format, and metrics employed for each benchmark, illustrating the breadth of methodologies applied in performance testing and optimization of NoSQL systems.

Performance benchmarking is crucial for assessing NoSQL databases' capabilities and efficiencies, providing insights into their effectiveness across diverse real-world applications. Employing metrics like latency and throughput facilitates comprehensive evaluations under various workloads and conditions [66]. While traditional metrics like latency are widely used, advanced metrics such as Traversed Edges Per Second (TEPS) offer significant potential for evaluating graph data management systems, yet remain underutilized. This gap highlights the need for continued innovation in benchmarking methodologies to better reflect modern data environments' complexities. Table 4 offers a comprehensive overview of representative benchmarks utilized in the performance evaluation of NoSQL databases, highlighting their applicability across different domains and tasks.

Comprehensive benchmarking's necessity is underscored by studies like those on Bigtable Merge Compaction (BMC), demonstrating superior performance over existing algorithms [110]. Similarly, Layered List Labeling's performance benchmarking indicates effectiveness in real-world applications, emphasizing robust frameworks' need [111]. These findings illustrate performance benchmarking's critical role in guiding NoSQL database solutions' development and optimization.

Integrating Apache Hive with Spark through the Hive Warehouse Connector exemplifies benchmarking's role in enhancing interoperability and efficiency in data transfer processes [112]. Optimizing key-value stores tailored for flash storage characteristics demonstrates aligning NoSQL databases with specific technological environments boosts performance and efficiency [47]. Such targeted optimizations ensure NoSQL databases meet high-performance applications' demands, particularly where speed and reliability are paramount.

In cloud environments, probabilistic modeling techniques effectively predict system behavior under varying loads, underscoring benchmarking's role in understanding cloud elasticity's non-deterministic aspects [113]. These insights are vital for optimizing NoSQL databases for cloud applications, ensuring efficient handling of dynamic workloads and resource allocation challenges. Probabilistic models help organizations anticipate performance fluctuations and proactively adjust configurations to maintain optimal performance.

The automated extraction of NoSQL database models, as demonstrated by Query2Model, facilitates query expression for end-users, showcasing benchmarking's practical benefits in enhancing user interaction and data accessibility [114]. MCAS architecture's benchmarking highlights its potential

for future integration with emerging memory technologies, offering high-performance, low-latency data operations critical for real-world applications [115]. These advancements underscore continuous innovation in benchmarking practices to keep pace with evolving technological landscapes.

Further research should explore emerging trends and address SQL and NoSQL databases' current limitations, as highlighted by existing studies [116]. The Online Marketplace benchmark effectively illustrates microservice practitioners' data management challenges and its potential to guide future data system designs [117]. Identifying tractable cases for computing the Shapley value in database management indicates continued research needs in theoretical and practical applications [59]. Addressing these challenges enhances benchmarking frameworks' effectiveness and contributes to NoSQL database technologies' ongoing evolution.

Performance benchmarking is essential for validating and optimizing NoSQL databases, providing systematic approaches to assess capabilities in handling diverse and complex real-world application requirements. Current benchmarks, like the Yahoo! Cloud Serving Benchmark (YCSB), primarily focus on basic read and write operations, highlighting significant gaps in evaluating advanced workloads and features specific to NoSQL types, such as document stores. This evaluation ensures organizations select suitable NoSQL technologies to manage large volumes of unstructured data while maintaining performance, scalability, and interoperability across cloud platforms [118, 119, 36, 37]. Comprehensive benchmarking frameworks and advanced metrics exploration enhance NoSQL systems' performance and reliability, driving innovation and efficiency in data management practices.

9 Conclusion

NoSQL databases have emerged as pivotal tools in modern data management, offering unparalleled scalability and flexibility essential for handling vast amounts of unstructured and semi-structured data in distributed systems. Their capacity to support rapid data access and dynamic schema evolution makes them indispensable for applications with fluctuating data requirements. The ability to manage high-velocity data streams and facilitate horizontal scaling empowers organizations to swiftly adapt to evolving data landscapes. Nonetheless, optimizing the deployment of NoSQL systems demands a strategic alignment with application-specific requirements to fully leverage their capabilities.

In distributed environments, the implementation of NoSQL databases has demonstrated substantial improvements in processing efficiency, notably reducing intra-node join spans and achieving near-linear speedup as the number of nodes increases. Such advancements are crucial for applications necessitating real-time data processing and analytics, fostering efficient resource utilization and minimizing latency. The integration of graph databases further exemplifies the transformative impact of NoSQL systems, prompting a reevaluation of traditional querying approaches. Enhanced querying and analytics capabilities, such as those offered by SPARQL, underscore the growing need for sophisticated tools to manage interconnected data effectively.

Security concerns persist as a significant challenge in the realm of NoSQL databases, with the diverse architectures and data models presenting unique vulnerabilities. Addressing these security gaps is imperative to ensuring data integrity and protection. Solutions like SEC-NoSQL offer promising avenues for secure data querying, maintaining high performance even under demanding conditions. The insights provided by benchmarks such as the LDBC SNB are instrumental in advancing our understanding of graph database technologies, laying the groundwork for future research and development.

The integration of machine learning into big data analytics within NoSQL environments enhances collaborative efforts between data architects and machine learning experts, thereby expanding the analytical capabilities of these systems. This synergy is vital as organizations increasingly depend on data-driven strategies, necessitating robust analytical tools. The potential for machine learning algorithms to refine data processing and retrieval opens new research pathways and application opportunities. As NoSQL databases continue to evolve, they promise to play a pivotal role in advancing data management and analytics, ensuring they remain integral to addressing the complex demands of distributed systems in the future.

References

[1] Souad Amghar, Safae Cherdal, and Salma Mouline. Storing, preprocessing and analyzing tweets: Finding the suitable nosql system, 2020.

[2] Mahdi Bohlouli, Frank Schulz, Lefteris Angelis, David Pahor, Ivona Brandic, David Atlan, and Rosemary Tate. Towards an integrated platform for big data analysis, 2020.

[3] Randal Burns, William Gray Roncal, Dean Kleissas, Kunal Lillaney, Priya Manavalan, Eric Perlman, Daniel R. Berger, Davi D. Bock, Kwanghun Chung, Logan Grosenick, Narayanan Kasthuri, Nicholas C. Weiler, Karl Deisseroth, Michael Kazhdan, Jeff Lichtman, R. Clay Reid, Stephen J. Smith, Alexander S. Szalay, Joshua T. Vogelstein, and R. Jacob Vogelstein. The open connectome project data cluster: Scalable analysis and vision for high-throughput neuroscience, 2013.

[4] Florida Estrella, Richard McClatchey, Zsolt Kovacs, Jean-Marie Le Goff, and Steven Murray. Using self-description to handle change in systems, 2002.

[5] Pooyan Jamshidi, Miguel Velez, Christian Kästner, Norbert Siegmund, and Prasad Kawthekar. Transfer learning for improving model predictions in highly configurable software, 2017.

[6] Yongrui Qin, Quan Z. Sheng, Nickolas J. G. Falkner, Schahram Dustdar, Hua Wang, and Athanasios V. Vasilakos. When things matter: A data-centric view of the internet of things, 2014.

[7] Ciprian-Octavian Truică, Elena-Simona Apostol, Jérôme Darmont, and Torben Bach Pedersen. The forgotten document-oriented database management systems: An overview and benchmark of native xml dodbmses in comparison with json dodbmses, 2021.

[8] Robert Escriva, Bernard Wong, and Emin Gün Sirer. Warp: Lightweight multi-key transactions for key-value stores, 2015.

[9] Fares Zaidi, Laurent Amanton, and Eric Sanlaville. Towards a novel cooperative logistics information system framework, 2019.

[10] A B M Moniruzzaman and Syed Akhter Hossain. Nosql database: New era of databases for big data analytics - classification, characteristics and comparison, 2013.

[11] Arkady Zaslavsky, Charith Perera, and Dimitrios Georgakopoulos. Sensing as a service and big data, 2013.

[12] Andreas Meier, Michael Kaufmann, Andreas Meier, and Michael Kaufmann. Nosql databases. *SQL & NoSQL databases: Models, languages, consistency options and architectures for big data management*, pages 201–218, 2019.

[13] Sergio Esteves, Joao Nuno Silva, and Luis Veiga. Palpatine: Mining frequent sequences for data prefetching in nosql distributed key-value stores, 2020.

[14] Daniel Glake, Felix Kiehn, Mareike Schmidt, Fabian Panse, and Norbert Ritter. Towards polyglot data stores – overview and open research questions, 2022.

[15] Mohamed Nadjib Mami, Damien Graux, Harsh Thakkar, Simon Scerri, Sören Auer, and Jens Lehmann. The query translation landscape: a survey, 2019.

[16] Marco Landoni, Laurent Marty, Dave Young, Laura Asquini, Stephen Smartt, Sergio Campana, Riccardo Claudi, Pietro Schipani, Matteo Aliverti, Federico Battaini, Andrea Baruffolo, Sagi Ben-Ami, Federico Biondi, Andrea Bianco, Giulio Capasso, Rosario Cosentino, Francesco D'Alessio, Paolo D'Avanzo, Matteo Genoni, Ofir Hershko, Hanindyo Kuncarayakti, Matteo Munari, Giuliano Pignata, Adam Rubin, Salvatore Scuderi, Fabrizio Vitali, Jani Achren, Jose Antonio Araiza Duran, Iair Arcavi, Anna Brucalassi, Rachel Bruch, Enrico Cappellaro, Mirko Colapietro, Massimo Della Valle, Marco De Pascale, Rosario Di Benedetto, Sergio D'Orsi, Avishay Gal Yam, Marcos Hernandez, Jari Kotilainen, Gianluca Li Causi, Seppo Mattila, Luca Oggioni, Giorgio Pariani, Michael Rappaport, Kalyan Radhakrishnan, Davide Ricci, Marco Riva, Bernardo Salasnich, and Ricardo Zanmar Sanchez. The quality check system architecture for son-of-x-shooter soxs, 2022.

AI-generated, for reference only.

[17] Jun-Sung Kim, Kyu-Young Whang, Hyuk-Yoon Kwon, and Il-Yeol Song. Odysseus/dfs: Integration of dbms and distributed file system for transaction processing of big data, 2014.

[18] Samiya Khan, Xiufeng Liu, Syed Arshad Ali, and Mansaf Alam. Storage solutions for big data systems: A qualitative study and comparison, 2019.

[19] Ziqing Guo, Hua Zhang, Xin Zhang, Zhengping Jin, and Qiaoyan Wen. Secure and efficiently searchable iot communication data management model: Using blockchain as a new tool, 2018.

[20] Marcos K. Aguilera, Joshua B. Leners, Ramakrishna Kotla, and Michael Walfish. Yesquel: scalable sql storage for web applications, 2014.

[21] Douglas Kunda and Hazael Phiri. A comparative study of nosql and relational database. *Zambia ICT Journal*, 1(1):1–4, 2017.

[22] Alejandro Corbellini, Cristian Mateos, Alejandro Zunino, Daniela Godoy, and Silvia Schiaffino. Persisting big-data: The nosql landscape. *Information Systems*, 63:1–23, 2017.

[23] Hayden Jananthan, Ziqi Zhou, Vijay Gadepally, Dylan Hutchison, Suna Kim, and Jeremy Kepner. Polystore mathematics of relational algebra, 2017.

[24] Mayank Kejriwal and Daniel P. Miranker. Self-contained nosql resources for cross-domain rdf, 2016.

[25] Véronique Benzaken, Giuseppe Castagna, Kim Nguy ên, and Jérôme Siméon. Static and dynamic semantics of nosql languages, 2013.

[26] Chaimae Asaad, Karim Baïna, and Mounir Ghogho. Nosql databases: Yearning for disambiguation, 2020.

[27] Carlos Javier Fernández Candel, Jesús Joaquín García Molina, and Diego Sevilla Ruiz. Skiql: A unified schema query language, 2022.

[28] Jim Gray, David T. Liu, Maria Nieto-Santisteban, Alexander S. Szalay, David DeWitt, and Gerd Heber. Scientific data management in the coming decade, 2005.

[29] Sanjay Rathee and Sheah Lin Lee. So you want to be a super researcher?, 2021.

[30] Andrew Habib, Avraham Shinnar, Martin Hirzel, and Michael Pradel. Type safety with json subschema, 2020.

[31] Harsh Thakkar, Dharmen Punjani, Soeren Auer, and Maria-Esther Vidal. Towards an integrated graph algebra for graph pattern matching with gremlin (extended version), 2019.

[32] Jiaheng Lu, Zhen Hua Liu, Pengfei Xu, and Chao Zhang. Udbms: Road to unification for multi-model data management, 2016.

[33] Xingbo Wu, Fan Ni, and Song Jiang. Wormhole: A fast ordered index for in-memory data management, 2019.

[34] Rui Xie, Linsen Ma, Alex Zhong, Feng Chen, and Tong Zhang. Zipcache: A dram/ssd cache with built-in transparent compression, 2024.

[35] Olivier Curé, Myriam Lamolle, and Chan Le Duc. Ontology based data integration over document and column family oriented nosql, 2013.

[36] Wisal Khan, Teerath Kumar, Zhang Cheng, Kislay Raj, Arunabha M Roy, and Bin Luo. Sql and nosql databases software architectures performance analysis and assessments – a systematic literature review, 2022.

[37] Wisal Khan, Teerath Kumar, Cheng Zhang, Kislay Raj, Arunabha M Roy, and Bin Luo. Sql and nosql database software architecture performance analysis and assessments—a systematic literature review. *Big Data and Cognitive Computing*, 7(2):97, 2023.

[38] Vijay Gadepally, Peinan Chen, Jennie Duggan, Aaron Elmore, Brandon Haynes, Jeremy Kepner, Samuel Madden, Tim Mattson, and Michael Stonebraker. The bigdawg polystore system and architecture, 2016.

[39] Yaser Mansouri and M. Ali Babar. The impact of distance on performance and scalability of distributed database systems in hybrid clouds, 2020.

[40] Gábor Szárnyas, Brad Bebee, Altan Birler, Alin Deutsch, George Fletcher, Henry A. Gabb, Denise Gosnell, Alastair Green, Zhihui Guo, Keith W. Hare, Jan Hidders, Alexandru Iosup, Atanas Kiryakov, Tomas Kovatchev, Xinsheng Li, Leonid Libkin, Heng Lin, Xiaojian Luo, Arnau Prat-Pérez, David Püroja, Shipeng Qi, Oskar van Rest, Benjamin A. Steer, Dávid Szakállas, Bing Tong, Jack Waudby, Mingxi Wu, Bin Yang, Wenyuan Yu, Chen Zhang, Jason Zhang, Yan Zhou, and Peter Boncz. The linked data benchmark council (ldbc): Driving competition and collaboration in the graph data management space, 2024.

[41] Renzo Angles, János Benjamin Antal, Alex Averbuch, Altan Birler, Peter Boncz, Márton Búr, Orri Erling, Andrey Gubichev, Vlad Haprian, Moritz Kaufmann, Josep Lluís Larriba Pey, Norbert Martínez, József Marton, Marcus Paradies, Minh-Duc Pham, Arnau Prat-Pérez, David Püroja, Mirko Spasić, Benjamin A. Steer, Dávid Szakállas, Gábor Szárnyas, Jack Waudby, Mingxi Wu, and Yuchen Zhang. The ldbc social network benchmark, 2024.

[42] Yao Wu and Henan Guan. biggy: An implementation of unified framework for big data management system, 2018.

[43] Stefanie Scherzinger, Meike Klettke, and Uta Störl. Managing schema evolution in nosql data stores, 2013.

[44] Khanh Dang, Khuong Vo, and Josef Küng. A nosql data-based personalized recommendation system for c2c e-commerce, 2018.

[45] Ciprian-Octavian Truică, Florin Rădulescu, Alexandru Boicea, and Ion Bucur. Performance evaluation for crud operations in asynchronously replicated document oriented database, 2018.

[46] Mohamed Hassan. Big data, big decisions choosing the right database, 2024.

[47] Krijn Doekemeijer and Animesh Trivedi. Key-value stores on flash storage devices: A survey, 2022.

[48] Jaroslav Pokorny. Integration of relational and graph databases functionally, 2018.

[49] Xavier Martinez-Palau, David Dominguez-Sal, Reza Akbarinia, Patrick Valduriez, and Josep Lluís Larriba-Pey. On demand memory specialization for distributed graph databases, 2013.

[50] Martin Junghanns, André Petermann, Kevin Gómez, and Erhard Rahm. Gradoop: Scalable graph data management and analytics with hadoop, 2015.

[51] Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. Rcc: Resilient concurrent consensus for high-throughput secure transaction processing, 2020.

[52] Larissa C. Shimomura, Nikolay Yakovets, and George Fletcher. Reasoning on property graphs with graph generating dependencies, 2022.

[53] Benji Peng, Xuanhe Pan, Yizhu Wen, Ziqian Bi, Keyu Chen, Ming Li, Ming Liu, Qian Niu, Junyu Liu, Jinlang Wang, Sen Zhang, Jiawei Xu, and Pohsun Feng. Deep learning and machine learning, advancing big data analytics and management: Handy appetizer, 2024.

[54] Jim Gray. Data management: Past, present, and future, 2007.

[55] Muntasir Raihan Rahman, Wojciech Golab, Alvin AuYoung, Kimberly Keeton, and Jay J. Wylie. Toward a principled framework for benchmarking consistency, 2012.

[56] Jan Hoffmann, Ankush Das, and Shu-Chun Weng. Towards automatic resource bound analysis for ocaml, 2016.

[57] Uta Störl, Meike Klettke, and Stefanie Scherzinger. Nosql schema evolution and data migration: State-of-the-art and opportunities. In *EDBT*, volume 20, pages 655–658, 2020.

[58] Benjamin Fuller, Mayank Varia, Arkady Yerukhimovich, Emily Shen, Ariel Hamlin, Vijay Gadepally, Richard Shay, John Darby Mitchell, and Robert K. Cunningham. Sok: Cryptographically protected database search, 2017.

[59] Leopoldo Bertossi, Benny Kimelfeld, Ester Livshits, and Mikaël Monet. The shapley value in database management, 2024.

[60] Shuangshuang Cui, Hongzhi Wang, Xianglong Liu, Zeyu Tian, and Xiaoou Ding. Ts-cabinet: Hierarchical storage for cloud-edge-end time-series database, 2023.

[61] Nazim Faour. Data consistency simulation tool for nosql database systems, 2018.

[62] Álvaro García-Recuero, Sérgio Esteves, and Luís Veiga. Quality-of-data for consistency levels in geo-replicated cloud data stores, 2014.

[63] Jiaan Zeng. Resource sharing for multi-tenant nosql data store in cloud, 2016.

[64] Ahmad Maatouki, Marek Szuba, Jörg Meyer, and Achim Streit. A horizontally-scalable multiprocessing platform based on node.js, 2015.

[65] Abhirup Chakraborty. Processing database joins over a shared-nothing system of multicore machines, 2018.

[66] Miyuru Dayarathna and Toyotaro Suzumura. Benchmarking graph data management and processing systems: A survey, 2021.

[67] Raik Niemann, Nikolaos Korfiatis, Roberto Zicari, and Richard Göbel. Does query performance optimization lead to energy efficiency? a comparative analysis of energy efficiency of database operations under different workload scenarios, 2013.

[68] Massimo Bartoletti, Andrea Bracciali, Stefano Lande, and Livio Pompianu. A general framework for blockchain analytics, 2017.

[69] Fatma Abdelhedi, Rym Jemmali, and Gilles Zurfluh. Relational databases ingestion into a nosql data warehouse, 2022.

[70] Sahar Vahdati, Farah Karim, Jyun-Yao Huang, and Christoph Lange. Mapping large scale research metadata to linked data: A performance comparison of hbase, csv and xml, 2015.

[71] Anastasija Nikiforova, Artjoms Daskevics, and Otmane Azeroual. Nosql security: can my data-driven decision-making be influenced from outside?, 2023.

[72] Alexander Slesarev, Mikhail Mikhailov, and George Chernishev. Benchmarking hashing algorithms for load balancing in a distributed database environment, 2022.

[73] Michael Assad, Christopher Meiklejohn, Heather Miller, and Stephan Krusche. Can my microservice tolerate an unreliable database? resilience testing with fault injection and visualization, 2024.

[74] Anastasija Nikiforova. Data security as a top priority in the digital world: preserve data value by being proactive and thinking security first, 2023.

[75] Sabrina Sicari, Alessandra Rizzardi, and Alberto Coen-Porisini. Security&privacy issues and challenges in nosql databases. *Computer Networks*, 206:108828, 2022.

[76] Beng Chin Ooi, Gang Chen, Dumitrel Loghin, Wei Wang, and Meihui Zhang. 5g: Agent for further digital disruptive transformations, 2019.

[77] Andrea Detti, Michele Orru, Riccardo Paolillo, Giulio Rossi, Pierpaolo Loreti, Lorenzo Bracciale, and Nicola Blefari Melazzi. Application of information centric networking to nosql databases: the spatio-temporal use case, 2017.

[78] Yusuke Wakuta, Michael Mior, Teruyoshi Zenmyo, Yuya Sasaki, and Makoto Onizuka. Nosql schema design for time-dependent workloads, 2023.

[79] Hasan M. Jamil. Knowledge rich natural language queries over structured biological databases, 2017.

[80] Lin Sun, Jun Zhao, Xiaojun Ye, Shuo Feng, Teng Wang, and Tao Bai. Conditional analysis for key-value data with local differential privacy, 2019.

[81] Daniel Beßler, Sascha Jongebloed, and Michael Beetz. Prolog as a querying language for mongodb, 2021.

[82] Dehua Liu, Selasi Kwashie, Yidi Zhang, Guangtong Zhou, Michael Bewong, Xiaoying Wu, Xi Guo, Keqing He, and Zaiwen Feng. An efficient approach for discovering graph entity dependencies (geds), 2023.

[83] Jason Mohoney, Anil Pacaci, Shihabur Rahman Chowdhury, Ali Mousavi, Ihab F. Ilyas, Umar Farooq Minhas, Jeffrey Pound, and Theodoros Rekatsinas. High-throughput vector similarity search in knowledge graphs, 2023.

[84] Marina Delianidi, Michail Salampasis, Konstantinos Diamantaras, Theodosios Siomos, Alkiviadis Katsalis, and Iphigenia Karaveli. A graph-based method for session-based recommendations, 2021.

[85] Alberto Hernández Chillón, Meike Klettke, Diego Sevilla Ruiz, and Jesús García Molina. A taxonomy of schema changes for nosql databases, 2022.

[86] Theofanis P. Raptis, Claudio Cicconetti, Manolis Falelakis, Tassos Kanellos, and Tomás Pariente Lobo. Design guidelines for apache kafka driven data management and distribution in smart cities, 2022.

[87] Phanwadee Sinthong and Michael J. Carey. Aframe: Extending dataframes for large-scale modern data analysis (extended version), 2019.

[88] Nimo Beeren. Designing a visual tool for property graph schema extraction and refinement: An expert study, 2022.

[89] Vitor Furlan de Oliveira, Marcosiris Amorim de Oliveira Pessoa, Fabrício Junqueira, and Paulo Eigi Miyagi. Sql and nosql databases in the context of industry 4.0. *Machines*, 10(1):20, 2021.

[90] Jie Song, Yichuan Zhang, Yubin Bao, and Ge Yu. Probery: A probability-based incomplete query optimization for big data, 2019.

[91] Hubert Naacke, Olivier Curé, and Bernd Amann. Sparql query processing with apache spark, 2016.

[92] Oleg Berzin, Rafael Ansay, James Kempf, Imam Sheikh, and Doron Hendel. The iot exchange, 2021.

[93] Emmanuel Cecchet, George Candea, and Anastasia Ailamaki. Middleware-based database replication: The gaps between theory and practice, 2008.

[94] Nandish Jayaram, Rohit Bhoopalam, Chengkai Li, and Vassilis Athitsos. Orion: Enabling suggestions in a visual query builder for ultra-heterogeneous graphs, 2016.

[95] Craig Franke, Samuel Morin, Artem Chebotko, John Abraham, and Pearl Brazier. Distributed semantic web data management in hbase and mysql cluster, 2011.

[96] Andrea Hillenbrand, Uta Störl, Shamil Nabiyev, and Stefanie Scherzinger. Migcast in monte carlo: The impact of data model evolution in nosql databases, 2021.

[97] Burak Yıldız, Tolga Büyüktanır, and Fatih Emekci. Equi-depth histogram construction for big data with quality guarantees, 2016.

[98] Shun Yao, Hongzhi Wang, and Yu Yan. Index selection for nosql database with deep reinforcement learning, 2020.

[99] Larissa C. Shimomura, George Fletcher, and Nikolay Yakovets. Ggds: Graph generating dependencies, 2020.

[100] Sergey Gorshkov, Alexander Grebeshkov, and Roman Shebalov. Ontology-based industrial data management platform, 2021.

[101] Hung-Fu Chang and Tzu-Kang Lin. Real-time structural health monitoring system using internet of things and cloud computing, 2019.

[102] Verifying transactional consistency of mongodb.

[103] Omar Almootassem, Syed Hamza Husain, Denesh Parthipan, and Qusay H. Mahmoud. A cloud-based service for real-time performance evaluation of nosql databases, 2017.

[104] Stefanie Scherzinger and Sebastian Sidortschuck. An empirical study on the design and evolution of nosql database schemas, 2020.

[105] Sattam Alsubaiee, Yasser Altowim, Hotham Altwaijry, Alexander Behm, Vinayak Borkar, Yingyi Bu, Michael Carey, Inci Cetindil, Madhusudan Cheelangi, Khurram Faraaz, Eugenia Gabrielova, Raman Grover, Zachary Heilbron, Young-Seok Kim, Chen Li, Guangqiang Li, Ji Mahn Ok, Nicola Onose, Pouria Pirzadeh, Vassilis Tsotras, Rares Vernica, Jian Wen, and Till Westmann. Asterixdb: A scalable, open source bdms, 2014.

[106] Martin Štufi, Boris Bačić, and Leonid Stoimenov. Big data architecture in czech republic healthcare service: Requirements, tpc-h benchmarks and vertica, 2020.

[107] Samiya Khan, Xiufeng Liu, Syed Arshad Ali, and Mansaf Alam. Bivariate, cluster and suitability analysis of nosql solutions for different application areas, 2019.

[108] Muhammad Jahanzeb Khan, Rui Hu, Mohammad Sadoghi, and Dongfang Zhao. Comparative evaluation of data decoupling techniques for federated machine learning with database as a service, 2023.

[109] Kunal Gupta, Astha Sachdev, and Ashish Sureka. Empirical analysis on comparing the performance of alpha miner algorithm in sql query language and nosql column-oriented databases using apache phoenix, 2017.

[110] Claire Mathieu, Carl Staelin, Neal E. Young, and Arman Yousefi. Bigtable merge compaction, 2015.

[111] Michael A. Bender, Alex Conway, Martin Farach-Colton, Hanna Komlos, and William Kuszmaul. Layered list labeling, 2024.

[112] Michele Gentile and Massimiliano Morrelli. Integrazione di apache hive con spark, 2019.

[113] Athanasios Naskos, Emmanouela Stachtiari, Anastasios Gounaris, Panagiotis Katsaros, Dimitrios Tsoumakos, Ioannis Konstantinou, and Spyros Sioutas. Cloud elasticity using probabilistic model checking, 2014.

[114] Amal Ait Brahim, Rabah Tighilt Ferhat, and Gilles Zurfluh. Incremental extraction of a nosql database model using an mda-based process, 2019.

[115] Daniel Waddington, Clem Dickey, Moshik Hershcovitch, and Sangeetha Seshadri. An architecture for memory centric active storage (mcas), 2021.

[116] Widya Nita Suliyanti. Studi literatur basis data sql dan nosql. *Kilat*, 8(1):48–51, 2019.

[117] Rodrigo Laigner, Zhexiang Zhang, Yijian Liu, Leonardo Freitas Gomes, and Yongluan Zhou. Online marketplace: A benchmark for data management in microservices, 2025.

[118] Muhammad Zohaib Khan, Fahim Uz Zaman, Muhammad Adnan, Aisha Imroz, Mahira Abdul Rauf, and Zuhaib Phul. Comparative case study: An evaluation of performance computation between sql and nosql database. *Journal of Software Engineering*, 1(2):14–23, 2023.

[119] Vincent Reniers, Dimitri Van Landuyt, Ansar Rafique, and Wouter Joosen. On the state of nosql benchmarks. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, pages 107–112, 2017.

AI-generated, for reference only.

Disclaimer:

SurveyX is an AI-powered system designed to automate the generation of surveys. While it aims to produce high-quality, coherent, and comprehensive surveys with accurate citations, the final output is derived from the AI's synthesis of pre-processed materials, which may contain limitations or inaccuracies. As such, the generated content should not be used for academic publication or formal submissions and must be independently reviewed and verified. The developers of SurveyX do not assume responsibility for any errors or consequences arising from the use of the generated surveys.

AI-generated, for reference only.