A Survey of Multi-Model Databases: MongoDB, ArangoDB, and JSONB

Abstract

Multi-model databases (MMDBs) represent a significant advancement in data management, addressing the limitations of traditional relational databases by supporting diverse data types and enabling seamless integration across heterogeneous environments. This survey explores the capabilities of prominent MMDBs such as MongoDB, ArangoDB, and JSONB, highlighting their role in merging NoSQL and traditional database functionalities. The survey underscores the importance of schema flexibility and unified query languages in facilitating efficient data integration and querying across complex datasets. By examining the evolution of database systems, the survey illustrates how MMDBs overcome the challenges of schema rigidity, enabling comprehensive data analytics and insights in domains such as healthcare and IoT. The integration of advanced analytical techniques, including machine learning, further amplifies the utility of MMDBs in addressing real-world challenges. Despite their transformative potential, MMDBs face challenges related to performance optimization, scalability, and query language standardization. Innovative solutions, such as the development of a universal data model and enhanced query languages, are crucial for overcoming these obstacles. As research progresses, MMDBs are poised to play an indispensable role in modern data management, supporting the efficient handling of complex and voluminous data types and driving innovation across various applications.

1 Introduction

1.1 Significance of Multi-Model Databases

The emergence of multi-model databases (MMDBs) marks a significant evolution in data management, addressing the increasing complexity and variety of data types generated by contemporary applications [1]. MMDBs are adept at managing structured, semi-structured, and unstructured data, overcoming the limitations of traditional relational Database Management Systems (DBMSs) that struggle with rigid schema constraints [2]. Their ability to support multiple data models within a single framework is essential in data-rich environments such as social media, e-commerce, and IoT applications.

MMDBs play a crucial role in integrating diverse data sources, a necessity underscored by the growing demand for scalable data science applications [3]. By offering a unified platform for data integration, MMDBs empower organizations to harness heterogeneous data for comprehensive analytics, thereby enhancing data-driven decision-making and competitive advantage [4]. This integration is particularly vital in complex domains like autonomous driving, where Graph Database Management Systems (GDBMSs) facilitate route planning and traffic flow predictions [5].

Additionally, MMDBs alleviate the complexities associated with querying in NoSQL databases like MongoDB, where cumbersome aggregation frameworks can hinder efficiency [6]. By supporting multi-model query languages, these databases streamline data retrieval processes, thereby expanding their applicability [1]. The incorporation of advanced analytical techniques, including machine

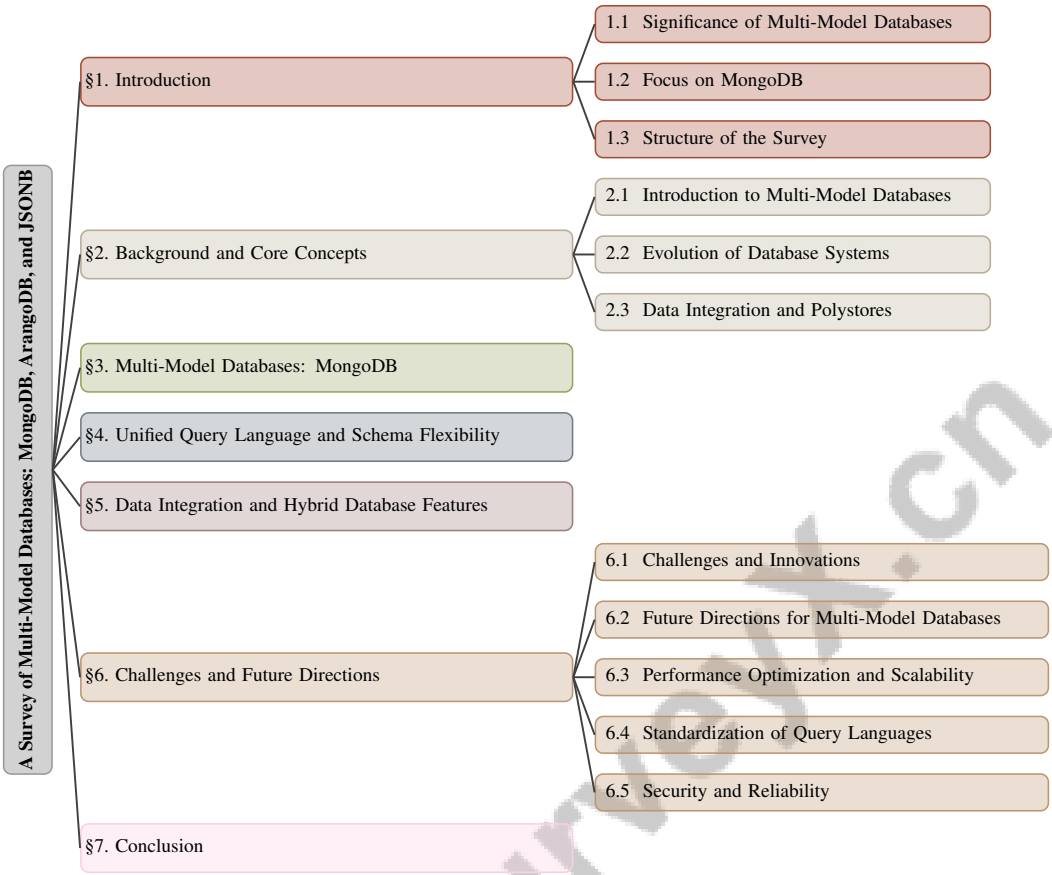

Figure 1: chapter structure

learning, further enhances the relevance of MMDBs, enabling effective modeling and analysis of complex datasets to tackle real-world challenges [7].

1.2 Focus on MongoDB, ArangoDB, and JSONB

This section investigates MongoDB, ArangoDB, and JSONB, each exemplifying key aspects of multi-model databases through their integration of NoSQL and traditional database functionalities. MongoDB, a leading NoSQL database, is characterized by its schema flexibility and capability to manage JSON-like documents, facilitating seamless data integration across diverse applications [8]. Its effectiveness is demonstrated in applications like the Smart Attendance System, which employs Convolutional Neural Networks (CNN) for face detection and recognition tasks [9]. The expressiveness and computational properties of MongoDB's query languages have been rigorously analyzed, showcasing its adaptability to varied data requirements [10]. Furthermore, its replication mechanism, utilizing the Raft protocol, enhances reliability in distributed systems, underscoring its significance in multi-model data management [11]. The relevance of MongoDB is further highlighted by BACKREST, a greybox fuzzer that improves vulnerability detection in web applications [12].

ArangoDB stands out for its native support of multi-model capabilities, integrating document, graph, and key/value data models within a single engine, essential for applications requiring intricate data relationships [13]. Its unified query language, AQL, streamlines querying across diverse data models, enhancing data management and integration [5]. ArangoDB's ability to handle ephemeral data in microservices environments further illustrates its adaptability in modern data architectures [14].

JSONB, an extension of PostgreSQL, offers a hybrid approach by incorporating JSON data storage and manipulation within a relational framework, providing the flexibility of NoSQL while maintaining the transactional integrity of traditional relational databases [15]. This integration allows JSONB

to utilize the full suite of SQL features, effectively managing structured, semi-structured, and unstructured data, aligning with the broader objectives of multi-model DBMSs [16].

Together, MongoDB, ArangoDB, and JSONB represent diverse strategies in multi-model database design, each offering unique strengths tailored to various data management needs. Their significance in contemporary data environments is underscored by their capacity to handle complex, heterogeneous data types, making them indispensable tools for organizations aiming to optimize their data assets [17].

1.3 Structure of the Survey

This survey is structured to provide a comprehensive analysis of multi-model databases, focusing on MongoDB, ArangoDB, and JSONB. The paper is organized into several key sections, each addressing critical aspects of multi-model database systems and their implications for modern data management.

The introduction establishes the significance of multi-model databases in managing diverse data types and facilitating integration. It emphasizes the necessity of schema flexibility, a unified query language, and robust integration capabilities, which are essential for handling complex datasets across various domains [18].

Following the introduction, the background and core concepts section explores the fundamental principles of multi-model databases, including schema flexibility, the evolution from traditional to NoSQL and hybrid models, and the role of polystores in integrating diverse data models. This foundation is crucial for understanding the dynamic landscape of data management technologies.

The subsequent section delves into the specific features and capabilities of MongoDB, ArangoDB, and JSONB, examining how each database supports various data models, schema flexibility, and unified query languages. This analysis is vital for appreciating the diverse strategies employed by these databases to address modern data challenges [19].

The survey then explores unified query languages and schema flexibility, highlighting their benefits for querying across different data models and their importance for effective data integration and management.

The following section analyzes the mechanisms and strategies employed by multi-model databases to integrate diverse data sources, focusing on hybrid database features that combine NoSQL and traditional functionalities. This examination provides insights into how these databases facilitate comprehensive data management through innovative architectural designs.

Challenges and future directions are addressed in the penultimate section, identifying current challenges such as performance optimization, scalability, and query language standardization. It also discusses innovative approaches to overcome these challenges and potential areas for future research.

Finally, the conclusion synthesizes the survey's key findings, reinforcing the importance of multi-model databases in modern data management and their potential to tackle complex integration challenges. Through this structured approach, the survey offers an in-depth analysis of multi-model databases (MMDBs), detailing their diverse data models—including document, graph, relational, and key-value—and evaluating their performance and usability via the UniBench benchmark. This benchmark simulates real-world applications, providing insights into internal data representations, query processing, and transaction performance of leading MMDBs like ArangoDB and OrientDB, thereby equipping readers with a thorough understanding of the capabilities and future developments of these databases in managing complex, heterogeneous data environments [20, 16].The following sections are organized as shown in Figure 1.

2 Background and Core Concepts

2.1 Introduction to Multi-Model Databases

Multi-model databases (MMDBs) revolutionize data management by overcoming the limitations of traditional relational database management systems (RDBMSs), which are constrained by rigid schemas and struggle with diverse data types [15]. MMDBs support multiple data models within a single architecture, enhancing flexibility and storage efficiency [21]. They adeptly integrate and query structured, semi-structured, and unstructured data, facilitating comprehensive analyses across

various domains, such as healthcare, where integrating multimodal datasets is crucial for AI-driven diagnostics [1, 22]. MMDBs manage schema evolution across multiple models, maintaining data consistency in dynamic environments [15].

Security in MMDBs, particularly in robust transaction processing across diverse data models, remains a critical concern. The use of neural computing techniques exemplifies MMDBs' capability to address real-world challenges [23]. They effectively manage various data types, as demonstrated in applications like detecting and tracking topics in streaming social media data [24]. The multi-model approach is vital for understanding complex interactions in ecosystems and managing modern data environments' heterogeneity [13]. By consolidating various data models, MMDBs enhance the usability of complex datasets, beneficial in big data applications like agricultural data mining [25].

MMDBs address schema inference challenges in NoSQL databases, where the absence of a predefined schema complicates data structure comprehension and query formulation [2]. They streamline data management and ensure integrity, providing comprehensive solutions to integration challenges across diverse domains [26]. MMDBs also enable rigorous evaluations of data management systems, facilitating comparisons across different models [17]. As organizations increasingly value integrated data systems, MMDBs are crucial for efficient data management strategies. The BigDAWG polystore architecture exemplifies this by integrating and querying diverse data sources, effectively managing complex datasets that do not conform to a single database engine [27, 28].

2.2 Evolution of Database Systems

The evolution of database systems from traditional RDBMSs to NoSQL and hybrid models is driven by the need to manage complex, heterogeneous data in modern applications. While RDBMSs offer robust transaction management and data integrity, their inflexible schema requirements limit adaptability [29]. NoSQL databases, such as MongoDB and ArangoDB, provide schema flexibility and optimize distributed, large-scale data environments, meeting demands for high availability and horizontal scalability [8]. They excel in handling diverse data models, including document, key-value, column-family, and graph models, which are advantageous for scalable, distributed architectures [30]. However, transitioning to NoSQL introduces challenges, including the need for innovative query languages and consistency models that diverge from traditional SQL and ACID properties [31]. Efficient management of complex, interconnected data is vital, especially in graph-structured data domains like academic publications and social networks [24].

Hybrid database models integrate the strengths of RDBMS and NoSQL paradigms, offering NoSQL flexibility while preserving traditional systems' transactional integrity. JSONB, an extension of PostgreSQL, exemplifies this integration by enabling JSON data manipulation within a relational framework, unifying structured, semi-structured, and unstructured data management [13]. This integration addresses operational complexities associated with managing multiple data models [15]. Advanced data management techniques, including machine learning, further enhance these systems' capabilities to manage complex datasets effectively [32]. Despite advancements, optimizing performance and scalability remains challenging, particularly in hybrid configurations where geographical distribution impacts latency and throughput [19]. The complexity of schema evolution in multi-model databases presents significant obstacles, as changes in one model can affect others, necessitating careful coordination [30].

The ongoing development of innovative database architectures is essential to address modern data management challenges as data complexity and volume continue to grow. As organizations seek to leverage their data assets, evolving database systems is crucial for facilitating efficient management strategies. The shift from single-model studies to multi-model approaches reflects broader trends toward more integrated and versatile solutions [25]. Additionally, the transition from traditional fuzzing methods to model-based approaches underscores the evolution of vulnerability detection strategies, emphasizing the need for advanced techniques in database management [12].

2.3 Data Integration and Polystores

Polystores and hybrid databases are pivotal in integrating diverse data models, addressing complexities in environments characterized by heterogeneous data sources and formats. The BigDAWG polystore architecture exemplifies this by enabling applications to transparently benefit from various databases while insulating them from underlying intricacies [31]. This mechanism is essential for seamless data

integration, particularly when data is distributed across different technologies optimized for specific data types or workloads [28].

The integration of diverse data models in polystores faces challenges related to semantic heterogeneity, complicating querying and integration processes. Solutions like federated database architectures mitigate these challenges by employing a global conceptual schema to encapsulate data heterogeneity and location, allowing users to write simpler, more intuitive queries [21]. These systems facilitate the integration of multimodal data, such as images and text, underscoring their relevance in managing complex data interactions [23].

Hybrid databases, such as MongoDB, enhance data integration by supporting multiple data models within a single platform, crucial for applications requiring the integration of structured, semi-structured, and unstructured data. However, inefficiencies in data movement and processing across various models and engines remain significant challenges, especially in polyglot persistence environments [4]. Performance evaluation frameworks, including benchmarks like BigDAWG, provide insights into polystore systems' capabilities in managing modern workloads. These benchmarks emphasize efficient query evaluation, particularly for complex queries involving multiple data models [27]. Additionally, integrating machine learning methodologies, including ensemble learning strategies, presents promising avenues for enhancing multi-view data integration, enabling more sophisticated analyses and insights [32].

Polystores and hybrid databases signify substantial advancements in data management, offering robust solutions for integrating diverse data models. Despite challenges, ongoing research and development efforts continue to refine these systems, aiming to enhance efficiency, scalability, and fault tolerance amid increasing data complexity and volume [3].

In recent years, the landscape of database management systems has evolved significantly, with a growing emphasis on multi-model capabilities that cater to diverse data requirements. As illustrated in Figure 2, this figure presents a hierarchical classification of features and capabilities of MongoDB, ArangoDB, and JSONB. It highlights their support for various data models, scalability, performance, and integration capabilities within multi-model database environments. This visual representation not only enhances the understanding of these systems' comparative strengths but also serves as a crucial reference point for evaluating their suitability in practical applications.

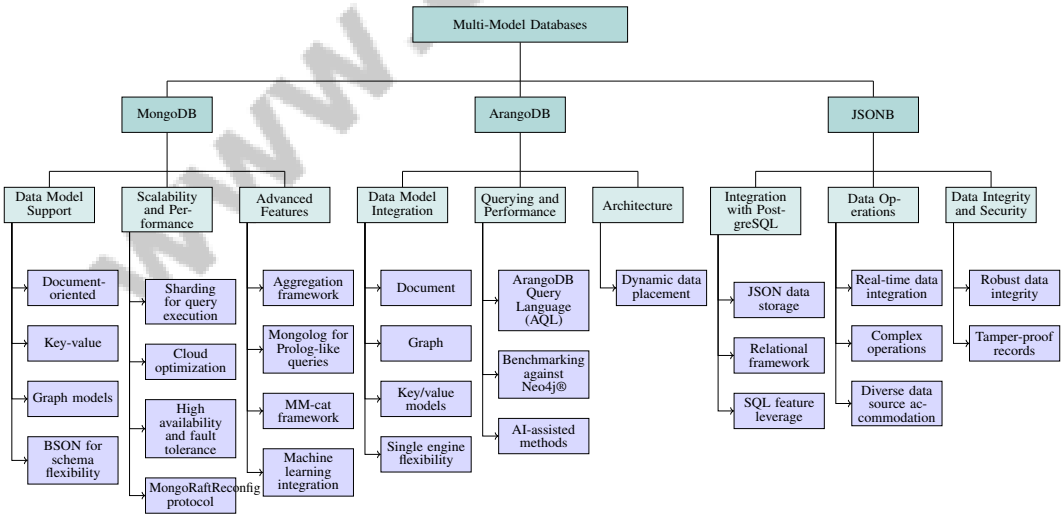

Figure 2: This figure illustrates the hierarchical classification of features and capabilities of MongoDB, ArangoDB, and JSONB, highlighting their support for various data models, scalability, performance, and integration capabilities within multi-model database environments.

3 Multi-Model Databases: MongoDB, ArangoDB, and JSONB

3.1 Features and Capabilities of MongoDB

MongoDB stands out as a leading document-oriented database, adept at supporting various data models such as document, key-value, and graph models, making it versatile for diverse applications [8]. Its use of BSON (Binary JSON) facilitates schema flexibility, allowing for the seamless integration of semi-structured and unstructured data, especially in environments where data structures are dynamic [33]. MongoDB's powerful aggregation framework supports a range of operations from simple retrievals to complex data transformations, catering to applications with sophisticated processing needs [10]. Additionally, Mongolog enhances adaptability by allowing queries in a Prolog-like syntax that are converted to MongoDB aggregation pipelines [6].

As illustrated in Figure 3, the key features and capabilities of MongoDB are highlighted, showcasing its support for various data models, scalability, and performance enhancements, along with advanced data management techniques. Scalability is a key feature, achieved through sharding, which expedites query execution, crucial for API development [34]. In cloud environments, MongoDB excels with optimized deployment techniques like cloud bursting, enhancing throughput and latency [35]. Its sharded cluster architecture ensures high availability and fault tolerance, essential for scalable applications [36]. The MongoRaftReconfig protocol supports safe node replacements, increasing system reliability without traditional log-based methods [11].

MongoDB's performance is exemplified in efficient storage and analysis of Tweets [37]. The MM-cat framework further enhances its functionality by allowing unified schema inference across different models [38]. MongoDB also supports efficient data access and preparation for machine learning, demonstrating its schema flexibility [39]. An automated performance testing framework ensures consistent system performance tests, crucial for evaluating MongoDB's scalability [40]. MongoDB's robust protocols and advanced data management techniques, including machine learning integration, make it indispensable for managing complex datasets [41, 3].

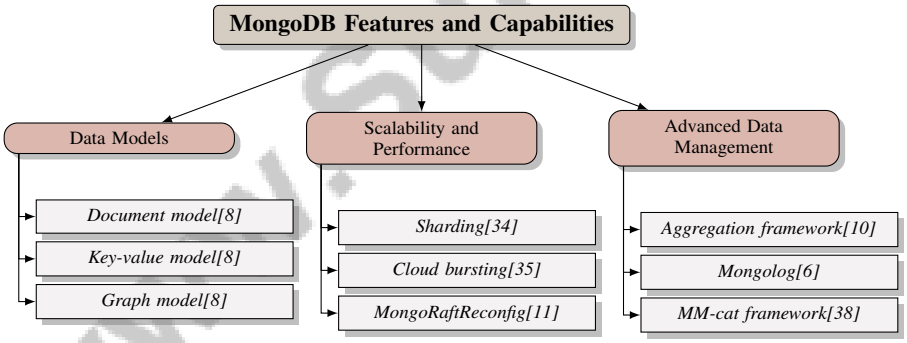

Figure 3: This figure illustrates the key features and capabilities of MongoDB, highlighting its support for various data models, scalability and performance enhancements, and advanced data management techniques. The data models include document, key-value, and graph models. Scalability is achieved through methods like sharding and cloud bursting, while MongoRaftReconfig ensures system reliability. Advanced data management is facilitated by MongoDB's aggregation framework, Mongolog for query simplification, and the MM-cat framework for unified schema inference.

3.2 Features and Capabilities of ArangoDB

ArangoDB is a multi-model database system adept at integrating and querying diverse data types, such as document, graph, and key/value models, making it ideal for applications with complex data relationships [13]. It supports multiple data models within a single engine, providing essential flexibility in modern environments [17]. The ArangoDB Query Language (AQL) allows efficient querying across data models, enhancing integration and management capabilities [5]. Benchmarking against other systems like Neo4j® highlights ArangoDB's proficiency in handling varying loads and data sizes [5].

ArangoDB's architecture supports AI-assisted methods, enhancing accuracy and reducing user workload in applications such as diagnostics [13]. The system dynamically adapts data placement strategies to optimize performance based on real-time application behavior.

3.3 Features and Capabilities of JSONB

JSONB, an extension of PostgreSQL, integrates JSON data storage and manipulation within a relational framework, leveraging SQL features to manage structured, semi-structured, and unstructured data. This capability is advantageous for real-time data integration and querying, as seen in applications like the Smart Attendance System [42]. JSONB supports complex operations, accommodating diverse data sources and formats, facilitating seamless integration across systems [43].

Its integration with PostgreSQL ensures robust data integrity and security, similar to systems like CreDB, which maintain tamper-proof records while allowing efficient data access [44]. This feature is crucial for maintaining reliability and consistency in multi-model environments.

4 Unified Query Language and Schema Flexibility

4.1 Concept and Benefits of Unified Query Language

A unified query language is pivotal in multi-model databases for enabling seamless data querying and integration across diverse data models. This framework enhances interoperability and simplifies the management of heterogeneous data sources. For example, Mongolog translates queries into MongoDB's aggregation pipelines, leveraging logic programming for complex data retrieval [6]. Such capabilities underscore the efficiency of a unified query language in data querying.

Furthermore, it supports comprehensive data analysis by integrating structured, semi-structured, and unstructured data within a single framework. The query augmentation method enriches local query results with related data from other databases, facilitating seamless integration [4]. The domain-specific language (ADIL) in AWESOME enhances tri-model analytics and automates optimizations across various data stores [3].

A unified query language mitigates vendor lock-in by standardizing query capabilities, akin to the associative array model, enhancing consistency and efficiency. Advanced data management techniques, such as adaptive learning algorithms, optimize querying in multi-model databases [7]. This implementation addresses the complexities of relational, graph, and semi-structured models, facilitating interaction across logical representations, schema inference, and data migration [15, 38, 45].

As illustrated in Figure 4, the figure highlights the main concepts and benefits of a unified query language in multi-model databases. It emphasizes its role in enhancing interoperability, comprehensive data analysis, and reducing vendor lock-in. Additionally, it outlines key methods such as Mongolog, Query Augmentation, and AWESOME, while addressing challenges like complex models, data integration, and schema inference. By providing a standardized framework, it enhances interoperability, reduces complexity, and supports comprehensive data analysis.

4.2 Importance of Schema Flexibility

Schema flexibility is crucial in multi-model databases, enhancing the ability to integrate and manage structured, semi-structured, and graph-based data by allowing systems to adapt dynamically to evolving structures and requirements. This flexibility streamlines storage and querying, reducing the need for costly ETL transformations and bridging the gap between data lakes and traditional warehouses. Schemaless models improve the extensibility and evolvability of databases, making them suitable for diverse data varieties [38, 2, 46]. This adaptability is vital in environments with rapid changes in data types and volumes, enabling efficient management of heterogeneous sources.

Dani's approach [47] exemplifies this flexibility by enabling schema inference directly from queries, eliminating external tools. The UDBMS platform [15] highlights managing multiple models within a single framework, addressing schema rigidity limitations.

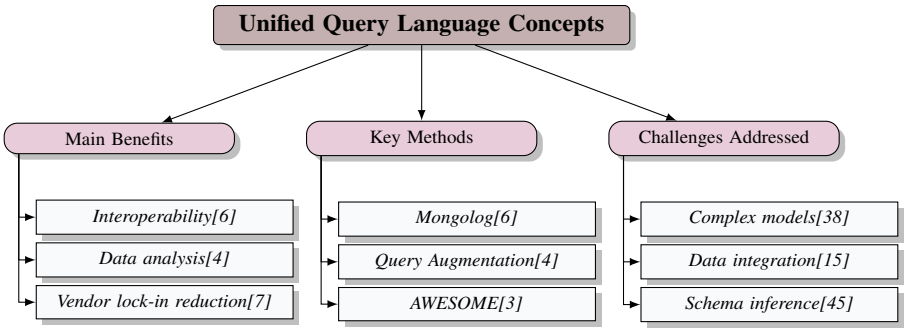

Figure 4: This figure illustrates the main concepts and benefits of a unified query language in multi-model databases, highlighting its role in enhancing interoperability, comprehensive data analysis, and reducing vendor lock-in. It also outlines key methods such as Mongolog, Query Augmentation, and AWESOME, and addresses challenges like complex models, data integration, and schema inference.

FluidMem [48] illustrates how schema flexibility enhances data management by allowing transparent remote memory access without system modifications. This principle underscores flexible schema architectures' role in improving integration and management across diverse models.

Additionally, NVCache's proposed admission policy [49] shows how schema flexibility can enhance cache performance by balancing data admission and lookup rates dynamically. This reflects the broader benefits of schema flexibility in optimizing data processing and storage operations.

Schema flexibility is a pivotal attribute of multi-model databases, enabling seamless integration and management by allowing adaptation to evolving data landscapes. It significantly improves storage efficiency and query performance, meeting contemporary data environments' changing requirements. By leveraging MMDBs and polystores, organizations can manage diverse data types without rigid schema constraints typical of traditional databases. Innovative querying techniques, like query augmentation, enhance data accessibility across storage engines, streamlining discovery and ensuring responsiveness to modern workloads [50, 4, 2, 51].

5 Data Integration and Hybrid Database Features

5.1 Enabling Data Integration through Multi-Model Databases

Multi-model databases are instrumental in integrating diverse data sources, employing various strategies to ensure seamless data integration across heterogeneous environments. By managing multiple data models, these databases enhance data integration and management efficiency. Query augmentation is a key strategy that enriches local query results by incorporating relevant data from other databases, thereby optimizing data querying processes [4]. Systems like AWESOME further enhance integration by optimizing cross-engine dataflows and supporting complex analytical operations, addressing challenges faced by conventional systems and enabling comprehensive analytics across varied data sources [3]. These strategies, including query augmentation, Multi-SQL, and optimized cross-engine dataflows, are crucial for managing relational, graph, document, and key-value models, facilitating efficient data integration in diverse environments [20, 16, 52].

5.2 Hybrid Database Features: Combining NoSQL and Traditional Functionalities

Hybrid databases represent a significant advancement in data management by integrating NoSQL and traditional relational database functionalities. They combine NoSQL's flexibility and scalability with the consistency and transactional integrity of relational databases, enabling effective management of diverse data types and large volumes of information. This integration is particularly beneficial for applications requiring high availability and reliable transactions, such as big data analytics and real-time processing [53, 51, 54, 37, 35]. MongoDB exemplifies this hybrid approach, supporting various data models critical for applications like social networking [34]. Its architecture addresses traditional databases' limitations by offering schema flexibility and efficient data processing, with future research focusing on optimization for specific applications [34].

Hybrid databases feature advanced support for multiple data formats, including JSON, XML, and graph data, enabling effective management and analysis of structured, semi-structured, and unstructured data. This capability addresses challenges posed by the increasing diversity of data from sources like IoT devices and social media, facilitating a unified data view and improving processing efficiency. MM-DBMSs eliminate traditional relational databases' limitations, offering enhanced flexibility, scalability, and performance for complex queries across heterogeneous data collections [55, 2, 56]. For instance, JSONB in PostgreSQL combines NoSQL's flexibility with relational transactional integrity, crucial for real-time processing and querying across diverse data types.

The hybrid architecture effectively merges NoSQL's scalability and flexibility with traditional databases' consistency and reliability, creating a comprehensive framework to address modern data management challenges, including heterogeneous data integration and efficient query processing [53, 37, 2, 51]. These systems empower organizations to manage complex datasets effectively, ensuring scalability, consistency, and flexibility in dynamic data environments. As the demand for versatile data management solutions grows, hybrid databases will play a pivotal role in addressing modern data management challenges across various domains.

6 Challenges and Future Directions

In the dynamic domain of multi-model databases (MMDBs), identifying challenges and innovations is crucial for advancement. As data ecosystems become more intricate, understanding the barriers to MMDB deployment and the innovative solutions designed to address these issues is essential. This section delves into challenges like data integration, query processing, and system performance, highlighting emerging strategies that aim to overcome these obstacles. This analysis provides insights into the current and future landscape of MMDBs.

6.1 Challenges and Innovations

MMDBs face several challenges that must be addressed to fully harness their potential in diverse data environments. A primary issue is the lack of a universal data model to encapsulate multi-model data differences, complicating query processing and optimization [6]. The absence of a global schema in polystores further complicates data integration and access [4]. Cross-modal data integration is hindered by system incompatibilities and the development of effective connectors and translators [41]. Additionally, high RAM usage for indexing and concurrency issues complicate MMDB adoption [5].

Innovative solutions are emerging to address these challenges. The UDBMS platform focuses on diversity, extensibility, and flexibility in multi-model data management [4], while the MM-cat approach enhances data management stability and efficiency through unified schema inference [4]. In query processing, the Quantum-Inspired Keyword Search (QIKS) system improves accuracy and efficiency in multi-model database queries [41]. Benchmarking tools like UniBench identify query processing bottlenecks and enable performance comparisons across systems [5].

Innovative methods are also being explored to address dataset replication challenges across multiple data management systems, which can increase write latency, by facilitating efficient query translation across different systems [12]. The need for comprehensive benchmarks for graph database systems highlights the necessity for further research [9]. Ongoing research initiatives, such as the development of UniBench and innovative methodologies, are essential for enhancing MMDB functionality and usability [20, 17, 57, 58]. By addressing schema management, data integration, and system adaptability, MMDBs can better support the diverse needs of contemporary data-driven applications.

6.2 Future Directions for Multi-Model Databases

The future of MMDBs is poised to be shaped by key research initiatives focusing on scalability, performance, and integration capabilities across diverse data environments. Developing more expressive query languages and a universal data model could streamline query processing and improve interoperability, leading to robust data management solutions for seamless integration and querying across heterogeneous data sources [4]. Optimizing graph algorithms for real-time applications and integrating graph databases with AI and machine learning can significantly enhance MMDB analytical capabilities [41].

Future research should prioritize multimodal aspects, as multimodal machine learning frameworks contribute to comprehensive data integration and analysis. Exploring dynamic data generation methods, evaluating performance under various sharding strategies, and optimizing query execution plans are critical areas for future research. These efforts could yield efficient and scalable data management solutions capable of handling modern datasets' increasing complexity and volume. Enhancing cross-system query mechanics and broadening supported data models and engines, as seen in systems like AWESOME, are essential for optimizing performance and ensuring seamless data integration across diverse platforms [3].

Further research should focus on optimizing software compatibility and exploring additional applications for existing systems, such as the Octa-Cluster, to improve hardware setups and performance [9]. Extending support for diverse backend storage solutions, as demonstrated by FluidMem, presents further research and development opportunities in MMDBs, enabling more flexible and adaptable data management architectures.

Integrating additional columnar storage options and data analysis tools to enhance benchmarking frameworks is another crucial area for future research, as this could yield valuable insights into MMDB performance and capabilities. Refining benchmarks to include real-world datasets and exploring additional graph database management systems (GDBMSs) could further validate findings and enhance MMDB applicability across various data environments [12].

6.3 Performance Optimization and Scalability

Benchmark	Size	Domain	Task Format	Metric
DB-HC[59]	10,000	Distributed Databases	Throughput Evaluation	Throughput, Scalability
DBP[60]	1,000,000	Medical Informatics	Query Performance Evaluation	Time to Query, CPU Score
MSESP[61]	20	Hydrology	Streamflow Prediction	CRPSS, RMSE
FPTP[62]	100,000	Database Management Systems	Conjunctive Range Queries	Accuracy, Average Performance Impact
NoSQL-Benchmark[63]	30	Database Performance Evaluation	Performance Testing	Time Taken
SM3[64]	40,000	Medical Informatics	Text-to-Query	Execution Accuracy, Valid Efficiency Score
DODBMS-Benchmark[56]	6,150,738	Information Retrieval	Filtering And Aggregation	Query Response Time
UniBench[20]	1,000,000	Social Commerce	Multi-model Query Processing	Query Response Time, Throughput

Table 1: Table presents a comprehensive overview of various benchmarks used for evaluating performance optimization and scalability in multi-model databases (MMDBs). It details the benchmark name, the size of the data set, the specific domain of application, the task format, and the metrics used for performance evaluation. This information is crucial for understanding the strengths and weaknesses of different MMDB systems and guiding future research efforts.

Performance optimization and scalability are crucial for MMDBs, designed to manage diverse and complex data environments. Efficiently handling large volumes of heterogeneous data while maintaining optimal performance presents significant challenges. Effective resource management is essential to ensure databases can scale horizontally and sustain performance under increasing loads. The Node Scala platform exemplifies advancements in this area, demonstrating superior response times and scalability compared to traditional Node Cluster setups [65].

Innovative frameworks like TorchQL enhance performance optimization in MMDBs, achieving query executions significantly faster than baseline systems, addressing inherent challenges in multi-model databases [66]. Rapid and efficient query execution is particularly important for applications requiring real-time data processing and analytics. MongoDB exhibits notable scalability and memory efficiency, especially in managing larger object-centric event logs, providing advantages over traditional in-memory approaches [67]. The scalability of MongoDB and similar systems is vital for supporting extensive data storage and retrieval operations.

Benchmarking tools like UniBench offer valuable insights into MMDB performance characteristics, identifying areas for improvement and guiding future research efforts. Table 1 provides a detailed overview of representative benchmarks that are instrumental in assessing the performance optimization and scalability of multi-model databases (MMDBs). By revealing MMDB systems' strengths and weaknesses, UniBench facilitates the development of more efficient and scalable data management solutions [68]. However, challenges persist, particularly in integrating and managing sensor data,

where issues such as sensor reliability, data overload, and integration complexity are prevalent. These challenges underscore the need for standardized integration methods to ensure consistent and reliable data processing across diverse data sources [25].

6.4 Standardization of Query Languages

Standardizing query languages in MMDBs is crucial for enhancing interoperability and usability across diverse data management systems. The absence of a universally accepted query language presents significant challenges, as existing methods often lack the flexibility and complexity to meet users' diverse needs. For instance, in the Virtual Observatory context, methods such as SIAP and ADQL are not interoperable and lack necessary flexibility [69].

The complexity of certain query language fragments, such as those in MQuery, further complicates standardization efforts, hindering practical implementations [10]. Addressing these challenges necessitates developing more streamlined and adaptable query languages capable of efficiently handling the diverse data models inherent in MMDBs.

One promising approach to enhancing query language standardization is developing portable ontological expressions that improve query portability across different NoSQL implementations, allowing greater flexibility without necessitating a rigid standardization process [70]. By facilitating query portability, this approach supports seamless integration and querying of heterogeneous data sources, enhancing MMDB interoperability.

Standardizing query languages in MMDBs remains a complex yet essential endeavor. By addressing interoperability, complexity, and portability challenges, ongoing initiatives can significantly enhance the development of robust and versatile data management solutions. These advancements are crucial for accommodating the increasing variety of data types—such as structured, semi-structured, and graph-based formats—present in modern data environments. Integrating MMDBs can bridge the architectural divide between data lakes and traditional data warehouses, reducing ETL transformation costs and improving flexibility through schemaless models. This holistic approach aligns with the FAIR principles of data management while addressing current systems' performance limitations in processing complex queries across diverse datasets [2, 58].

6.5 Security and Reliability

Security and reliability are paramount in deploying and operating MMDBs, given their role in managing diverse and complex data environments. Integrating various data models within a single system amplifies potential attack surfaces, necessitating robust security measures to guard against unauthorized access and data breaches. The proposed blockchain-enabled architecture exemplifies advancements in enhancing security, privacy, and data management, particularly in sensitive domains like healthcare systems, where traditional electronic health record (EHR) systems face numerous security challenges [71].

Reliability in MMDBs is further challenged by the need to maintain consistent performance across diverse data sources and infrastructures. Limitations in existing systems, such as delays in alarm notifications and difficulties in standardizing log file formats across subsystems, underscore the need for more reliable monitoring and alerting mechanisms [72]. These challenges are compounded by the complexity of integrating occupant behavior data, which often lacks generalizability due to specific data requirements [73].

In IoT applications, the reliability of sensor data and connectivity poses additional challenges. Automated systems' performance, such as irrigation systems, can be significantly affected by IoT sensors' reliability and network connectivity, highlighting the need for robust mechanisms to ensure data integrity and system reliability [74]. Moreover, understanding advanced mathematical concepts, such as category theory, can hinder the practical application of security and reliability frameworks in MMDBs, particularly for practitioners unfamiliar with these mathematical underpinnings [45].

To effectively address security and reliability challenges in MMDBs, a comprehensive strategy is essential. This strategy should integrate advanced security architectures, such as trusted execution environments offering blockchain-like integrity guarantees, and robust reliability mechanisms. For instance, the CreDB datastore employs innovative features like transaction witnesses for auditing, inviolable access policies, timeline inspection for data history analysis, and protected function

evaluation for secure computations over private data. Additionally, systems must manage the complexities of diverse data formats and sources, as seen in E-commerce applications, where data is stored in various models, including relational tables, graphs, JSON documents, and key/value pairs. Such an approach ensures consistent performance and data integrity across heterogeneous data environments [16, 44]. By addressing these challenges, MMDBs can enhance their utility and effectiveness in managing complex data-driven applications.

7 Conclusion

Multi-model databases (MMDBs) represent a significant advancement in the realm of data management, adeptly handling the complexity and diversity of modern data landscapes. Through the examination of MongoDB, ArangoDB, and JSONB, this survey illustrates the diverse strategies employed by MMDBs to facilitate efficient data integration and management. These databases exemplify the innovative approaches necessary for addressing the multifaceted challenges of data integration, which are crucial for deriving comprehensive analytics and insights.

The comparative analysis of NoSQL databases, such as MongoDB and CouchDB, highlights their superior performance in CRUD operations compared to traditional relational databases, underscoring their suitability for dynamic, large-scale data environments. This enhanced efficiency is further supported by the development of robust performance testing frameworks that aid in the early detection and resolution of performance issues, thereby ensuring optimal database functionality.

Moreover, the integration of cutting-edge data management techniques within MMDBs, as seen in automated clinical outcome systems, underscores their potential to drive significant advancements in data-centric applications. The ability of MMDBs to address structural uncertainties in complex modeling scenarios further cements their role as indispensable components in contemporary data management.

As MMDBs continue to evolve, they are poised to become central to the future of data management, offering unparalleled capabilities in integrating diverse data models and providing efficient querying solutions. Their ongoing development is expected to catalyze innovation and enhance efficiency in the management of expansive and intricate datasets, thereby improving decision-making processes and supporting a broad spectrum of applications across multiple domains.

References

[1] Qingsong Guo, Chao Zhang, Shuxun Zhang, and Jiaheng Lu. Multi-model query languages: taming the variety of big data. *Distributed and Parallel Databases*, 42(1):31–71, 2024.

[2] Sandro Bimonte, Yassine Hifdi, Mohammed Maliari, Patrick Marcel, and Stefano Rizzi. To each his own: Accommodating data variety by a multimodel star schema. In *Proceedings of the 22nd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data co-located with EDBT/ICDT 2020 Joint Conference (EDBT/ICDT 2020)*, 2020.

[3] Xiuwen Zheng, Subhasis Dasgupta, Arun Kumar, and Amarnath Gupta. Awesome: Empowering scalable data science on social media data with an optimized tri-store data system, 2022.

[4] Antonio Maccioni and Riccardo Torlone. Augmented access for querying and exploring a polystore. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 77–88. IEEE, 2018.

[5] Karin Festl, Patrick Promitzer, Daniel Watzenig, and Huilin Yin. Performance of graph database management systems as route planning solutions for different data and usage characteristics, 2023.

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

[7] Simon Torka and Sahin Albayrak. Alpaca – adaptive learning pipeline for comprehensive ai, 2024.

[8] Massimo Carro. Nosql databases, 2014.

[9] Joao Eduardo Montandon, Luciana Lourdes Silva, and Marco Tulio Valente. Identifying experts in software libraries and frameworks among github users, 2019.

[10] Elena Botoeva, Diego Calvanese, Benjamin Cogrel, and Guohui Xiao. Expressivity and complexity of mongodb (extended version), 2017.

[11] William Schultz, Siyuan Zhou, Ian Dardik, and Stavros Tripakis. Design and analysis of a logless dynamic reconfiguration protocol, 2021.

[12] François Gauthier, Behnaz Hassanshahi, Benjamin Selwyn-Smith, Trong Nhan Mai, Max Schlüter, and Micah Williams. Backrest: A model-based feedback-driven greybox fuzzer for web applications, 2021.

[13] Sydney Anuyah, Victor Bolade, and Oluwatosin Agbaakin. Understanding graph databases: A comprehensive tutorial and survey, 2024.

[14] Saverio Giallorenzo, Fabrizio Montesi, Larisa Safina, and Stefano Pio Zingaro. Ephemeral data handling in microservices - technical report, 2019.

[15] Jiaheng Lu, Zhen Hua Liu, Pengfei Xu, and Chao Zhang. Udbms: road to unification for multi-model data management. In *Advances in Conceptual Modeling: ER 2018 Workshops Emp-ER, MoBiD, MREBA, QMMQ, SCME, Xi'an, China, October 22-25, 2018, Proceedings 37*, pages 285–294. Springer, 2018.

[16] Jiaheng Lu and Irena Holubová. Multi-model databases: a new journey to handle the variety of data. *ACM Computing Surveys (CSUR)*, 52(3):1–38, 2019.

[17] Chao Zhang and Jiaheng Lu. Holistic evaluation in multi-model databases benchmarking. *Distributed and Parallel Databases*, 39(1):1–33, 2021.

[18] Junan Guo, Subhasis Dasgupta, and Amarnath Gupta. Multi-model investigative exploration of social media data with boutique: A case study in public health, 2019.

[19] Xiaoqing Zhang, Mingkai Xu, Yanru Li, Minmin Su, Ziyao Xu, Chunyan Wang, Dan Kang, Hongguang Li, Xin Mu, Xiu Ding, et al. Automated multi-model deep neural network for sleep stage scoring with unfiltered clinical data. *Sleep and Breathing*, 24:581–590, 2020.

[20] Holistic evaluation in multi-mod.

[21] Leonardo Guerreiro Azevedo, Renan Francisco Santos Souza, Elton F. de S. Soares, Raphael M. Thiago, Julio Cesar Cardoso Tesolin, Ann C. Oliveira, and Marcio Ferreira Moreno. A polystore architecture using knowledge graphs to support queries on heterogeneous data stores, 2024.

[22] Slavomir Matuska, Martin Paralic, and Robert Hudec. A smart system for sitting posture detection based on force sensors and mobile application, 2022.

[23] Meysam Asgari-Chenaghlu, Mohammad-Reza Feizi-Derakhshi, Leili farzinvash, Mohammad-Ali Balafar, and Cina Motamed. Topicbert: A transformer transfer learning based memory-graph approach for multimodal streaming social media topic detection, 2020.

[24] Andi Ferhati. Clustering graphs – applying a label propagation algorithm to detect communities in graph databases, 2022.

[25] Shu Tang, Dennis R Shelden, Charles M Eastman, Pardis Pishdad-Bozorgi, and Xinghua Gao. A review of building information modeling (bim) and the internet of things (iot) devices integration: Present status and future trends. *Automation in construction*, 101:127–139, 2019.

[26] James J. Cusick, William Miller, Nicholas Laurita, and Tasha Pitt. Design, construction, and use of a single board computer beowulf cluster: Application of the small-footprint, low-cost, insignal 5420 octa board, 2015.

[27] Vijay Gadepally, Jennie Duggan, Aaron Elmore, Jeremy Kepner, Samuel Madden, Tim Mattson, and Michael Stonebraker. The bigdawg architecture, 2016.

[28] Kyle OBrien, Vijay Gadepally, Jennie Duggan, Adam Dziedzic, Aaron Elmore, Jeremy Kepner, Samuel Madden, Tim Mattson, Zuohao She, and Michael Stonebraker. Bigdawg polystore release and demonstration, 2017.

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

[30] Adam P Arkin, Robert W Cottingham, Christopher S Henry, Nomi L Harris, Rick L Stevens, Sergei Maslov, Paramvir Dehal, Doreen Ware, Fernando Perez, Shane Canon, et al. Kbase: the united states department of energy systems biology knowledgebase. *Nature biotechnology*, 36(7):566–569, 2018.

[31] Jeremy Kepner, Vijay Gadepally, Dylan Hutchison, Hayden Jananthan, Timothy Mattson, Siddharth Samsi, and Albert Reuther. Associative array model of sql, nosql, and newsql databases, 2016.

[32] 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.

[33] Georg Schnabel. A computational exfor database, 2020.

[34] Sumitkumar Kanoje, Varsha Powar, and Debajyoti Mukhopadhyay. Using mongodb for social networking website, 2015.

[35] Yaser Mansouri, Victor Prokhorenko, and M. Ali Babar. An automated implementation of hybrid cloud for performance evaluation of distributed databases, 2020.

[36] Aaron Saxton and Stephen Squaire. Deploying a sharded mongodb cluster as a queued job on a shared hpc architecture, 2022.

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

[38] Pavel Koupil. Modelling and management of multi-model data. 2022.

[39] Chenguang Wan, Zhi Yu, Xiaojuan Liu, Xinghao Wen, Xi Deng, and Jiangang Li. A data management system for machine learning research of tokamak, 2022.

[40] Henrik Ingo and David Daly. Automated system performance testing at mongodb, 2020.

[41] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 1(1):1–34, 2020.

[42] Shailesh Arya, Hrithik Mesariya, and Vishal Parekh. Smart attendance system usign cnn, 2020.

[43] Tommaso Urli. json2run: a tool for experiment design analysis, 2013.

[44] Kai Mast, Lequn Chen, and Emin Gün Sirer. Enabling strong database integrity using trusted execution environments, 2018.

[45] A unified representation and tra.

[46] Sandro Bimonte, Enrico Gallinucci, Patrick Marcel, and Stefano Rizzi. Data variety, come as you are in multi-model data warehouses. *Information Systems*, 104:101734, 2022.

[47] Calvin Dani, Shiva Jahangiri, and Thomas Hütter. Introducing schema inference as a scalable sql function [extended version], 2024.

[48] Blake Caldwell, Youngbin Im, Sangtae Ha, Richard Han, and Eric Keller. Fluidmem: Memory as a service for the datacenter, 2017.

[49] Alexandra Fedorova, Keith Smith, Keith Bostic, Alexander Gorrod, Sue LoVerso, and Michael Cahill. Writes hurt: Lessons in cache design for optane nvram, 2022.

[50] Daniel Glake, Felix Kiehn, Mareike Schmidt, Fabian Panse, and Norbert Ritter. Towards polyglot data stores–overview and open research questions. *arXiv preprint arXiv:2204.05779*, 2022.

[51] Adam A. E. Alflahi, Mohammed A. Y. Mohammed, and Abdallah Alsammani. Enhancement of database access performance by improving data consistency in a non-relational database system (nosql), 2023.

[52] Yu Yan, Nan Jiang, Hongzhi Wang, Yutong Wang, Chang Liu, and Yuzhuo Wang. Multi-sql: An extensible multi-model data query language, 2021.

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

[54] Rishi Kesav Mohan, Risheek Rakshit Sukumar Kanmani, Krishna Anandan Ganesan, and Nisha Ramasubramanian. Evaluating nosql databases for olap workloads: A benchmarking study of mongodb, redis, kudu and arangodb, 2024.

[55] Sven Groppe and Jinghua Groppe. Hybrid multi-model multi-platform (hm3p) databases. In *DATA*, pages 177–184, 2020.

[56] 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.

[57] Perspective.

[58] Masoud Salehpour and Joseph G. Davis. Knowledge graphs for processing scientific data: Challenges and prospects, 2020.

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

[60] Johan Sandell, Einar Asplund, Workneh Yilma Ayele, and Martin Duneld. Performance comparison analysis of arangodb, mysql, and neo4j: An experimental study of querying connected data, 2024.

[61] Multi-model approaches for impro.

[62] Dawei Tao, Enqi Liu, Sidath Randeni Kadupitige, Michael Cahill, Alan Fekete, and Uwe Röhm. First past the post: Evaluating query optimization in mongodb, 2024.

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

[64] Sithursan Sivasubramaniam, Cedric Osei-Akoto, Yi Zhang, Kurt Stockinger, and Jonathan Fuerst. Sm3-text-to-query: Synthetic multi-model medical text-to-query benchmark, 2024.

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

[66] Aaditya Naik, Adam Stein, Yinjun Wu, Mayur Naik, and Eric Wong. Torchql: A programming framework for integrity constraints in machine learning, 2024.

[67] Alessandro Berti, Anahita Farhang Ghahfarokhi, Gyunam Park, and Wil M. P. van der Aalst. A scalable database for the storage of object-centric event logs, 2022.

[68] Chao Zhang, Jiaheng Lu, Pengfei Xu, and Yuxing Chen. Unibench: a benchmark for multi-model database management systems. In *Performance Evaluation and Benchmarking for the Era of Artificial Intelligence: 10th TPC Technology Conference, TPCTC 2018, Rio de Janeiro, Brazil, August 27–31, 2018, Revised Selected Papers 10*, pages 7–23. Springer, 2019.

[69] Yuji Shirasaki, Masatoshi Ohishi, Yoshihiko Mizumoto, Masahiro Tanaka, Satoshi Honda, Masafumi Oe, Naoki Yasuda, and Yoshifumi Masunaga. Structured query language for virtual observatory, 2004.

[70] Suresh K. Damodaran and Pedro A. Colon-Hernandez. Portable ontological expressions in nosql queries, 2016.

[71] Mohit Kumar, Hritu Raj, Nisha Chaurasia, and Sukhpal Singh Gill. Blockchain inspired secure and reliable data exchange architecture for cyber-physical healthcare system 4.0, 2023.

[72] L Gladstone, D Biare, L Cappelli, J S Cushman, F Del Corso, B K Fujikawa, K P Hickerson, N Moggi, C E Pagliarone, B Schmidt, S L Wagaarachchi, B Welliver, and L A Winslow. The cuore slow monitoring systems, 2016.

[73] Mathieu Bourdeau, Xiao qiang Zhai, Elyes Nefzaoui, Xiaofeng Guo, and Patrice Chatellier. Modeling and forecasting building energy consumption: A review of data-driven techniques. *Sustainable Cities and Society*, 48:101533, 2019.

[74] Ömer Aydin, Cem Ali Kandemir, Umut Kiraç, and Feriştah Dalkiliç. An artificial intelligence and internet of things based automated irrigation system, 2021.

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.