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

www.surveyx.cn

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 transformation in data management, addressing the complexities and variety of data types produced by contemporary applications [1]. These databases adeptly manage structured, semi-structured, and unstructured data, overcoming the limitations of traditional relational Database Management Systems (DBMSs) that are constrained by rigid schemas [2]. The capacity of MMDBs to support multiple data models within a single system is vital in data-rich environments such as social media, e-commerce, and IoT applications.

MMDBs play a crucial role in integrating diverse data sources, meeting the growing demand for scalable data science applications [3]. By offering a unified framework for data integration, they empower organizations to harness heterogeneous data for comprehensive analytics, thus enhancing competitive advantage in data-driven decision-making [4]. This integration capability is particularly essential in fields requiring complex data processing, exemplified by the use of Graph Database Management Systems (GDBMSs) in autonomous driving for tasks like route planning and traffic flow prediction [5].

Furthermore, MMDBs tackle the complexities associated with querying in NoSQL databases, such as MongoDB, where aggregation frameworks can complicate query processes [6]. By supporting multi-model query languages, these databases streamline data retrieval, enhancing their applicability

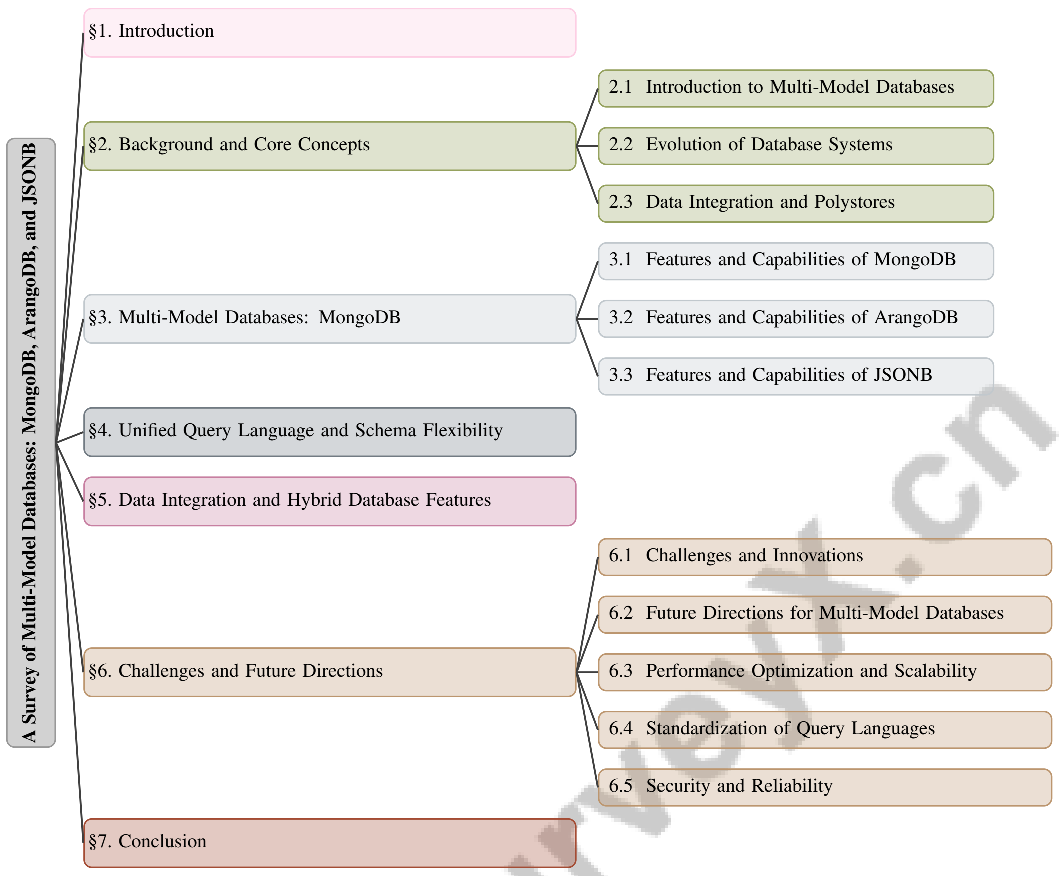
A Survey of Multi-Model Databases: MongoDB, ArangoDB, and JSONB
§1. Introduction
§2. Background and Core Concepts
2.1 Introduction to Multi-Model Databases
2.2 Evolution of Database Systems
2.3 Data Integration and Polystores
§3. Multi-Model Databases: MongoDB
3.1 Features and Capabilities of MongoDB
3.2 Features and Capabilities of ArangoDB
3.3 Features and Capabilities of JSONB
§4. Unified Query Language and Schema Flexibility
§5. Data Integration and Hybrid Database Features
§6. Challenges and Future Directions
6.1 Challenges and Innovations
6.2 Future Directions for Multi-Model Databases
6.3 Performance Optimization and Scalability
6.4 Standardization of Query Languages
6.5 Security and Reliability
§7. Conclusion

Figure 1: chapter structure

across various domains [1]. The integration of advanced analytical techniques, including machine learning, into MMDBs further amplifies their relevance, facilitating the modeling and analysis of intricate datasets to effectively address real-world challenges [7].

1.2 Focus on MongoDB, ArangoDB, and JSONB

This section evaluates MongoDB, ArangoDB, and JSONB, each representing a critical aspect of multi-model databases through their integration of NoSQL and traditional database functionalities. MongoDB, a leading NoSQL database, is noted for its schema flexibility and ability to manage JSON-like documents, enabling seamless data integration across diverse applications [8]. Its effectiveness in processing various data types is exemplified in the Smart Attendance System, which employs Convolutional Neural Networks (CNN) for face detection and recognition [9]. The formalization of MongoDB's query languages highlights its complexity and adaptability to diverse data requirements [10]. Additionally, MongoDB's replication mechanism, utilizing the Raft protocol, enhances reliability in distributed systems, reinforcing its significance in multi-model data management [11]. Its relevance is further underscored by BACKREST, a greybox fuzzer that improves vulnerability detection in web applications [12].

ArangoDB stands out for its native multi-model capabilities, supporting document, graph, and key-value data models within a single engine, essential for applications necessitating complex data relationships [13]. Its unified query language, AQL, promotes efficient querying across various data models, enhancing data integration and management [5]. ArangoDB's adeptness at handling ephemeral data in microservices environments further illustrates its flexibility and adaptability in modern data architectures [14].

JSONB, an extension of PostgreSQL, merges JSON data storage and manipulation within a relational database framework, offering NoSQL-like flexibility while preserving traditional relational database transactional integrity [15]. This hybrid approach allows JSONB to utilize a comprehensive suite of

AI-generated, for reference only.

SQL features, facilitating effective management of structured, semi-structured, and unstructured data, aligning with the broader objectives of multi-model DBMSs [16].

Together, MongoDB, ArangoDB, and JSONB exemplify diverse strategies in multi-model database design, each contributing unique strengths tailored to various data management needs. Their significance in modern data environments is underscored by their capacity to manage complex, heterogeneous data types, making them essential tools for organizations aiming to maximize their data assets [17].

1.3 Structure of the Survey

This survey is meticulously organized to provide a thorough examination of multi-model databases, focusing on MongoDB, ArangoDB, and JSONB. The paper comprises 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 explains the necessity of schema flexibility, unified query languages, and integration capabilities, which are essential for handling complex datasets across various domains [18].

Following the introduction, the background and core concepts section explores the foundational principles of multi-model databases, including schema flexibility, the evolution from traditional to NoSQL and hybrid models, and the role of polystores and hybrid databases in integrating diverse data models. These discussions lay a solid groundwork for understanding the dynamic landscape of data management technologies.

The subsequent section focuses on the specific features and capabilities of MongoDB, ArangoDB, and JSONB, analyzing how each database supports various data models, schema flexibility, and unified query languages, thereby emphasizing their unique contributions to multi-model data management. This analysis is crucial for appreciating the diverse strategies these databases employ to address modern data challenges [19].

The survey then delves into the concept of unified query languages and schema flexibility, highlighting their benefits for querying across different data models. This section underscores the importance of these features in achieving seamless data integration and management, which are vital for effectively leveraging data assets.

The mechanisms and strategies used by multi-model databases to integrate diverse data sources are examined next, analyzing hybrid database features that combine NoSQL and traditional functionalities. This section provides insights into how these databases facilitate comprehensive data management through innovative architectural designs.

Challenges and future directions form the penultimate section, identifying current challenges such as performance optimization, scalability, and query language standardization. This section also explores innovative approaches to address these issues and discusses potential areas for future research and development.

Finally, the conclusion synthesizes the survey's key findings, reinforcing the importance of multi-model databases in contemporary data management and their potential to tackle complex data integration challenges. By employing a structured methodology, this survey offers a comprehensive examination of multi-model databases (MMDBs), detailing their diverse data models—including document, graph, relational, and key-value—and analyzing their performance, usability, and future potential. The insights derived from this evaluation enhance the reader's understanding of MMDB capabilities while highlighting the challenges and innovations in managing complex, heterogeneous data environments [20, 1, 21, 16, 17].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 struggle with diverse data types due to their rigid schemas [15, 22]. MMDBs support multiple data models within a

AI-generated, for reference only.

single system, enhancing flexibility and storage capabilities [1]. They integrate and query structured, semi-structured, and unstructured data, enabling comprehensive cross-domain analysis, which is crucial in fields like healthcare for developing AI-driven diagnostic tools [23]. MMDBs also manage schema evolution effectively, ensuring consistency in dynamic environments [15].

Security is a significant concern, particularly for robust transaction processing across diverse data models. Neural computing techniques demonstrate MMDBs' efficacy in addressing real-world challenges [24]. They efficiently manage various data types, as seen in applications like social media topic detection and tracking [25]. The multi-model approach is essential for understanding complex ecosystem interactions and managing modern data environment heterogeneity [13]. By consolidating various data models, MMDBs enhance the usability of complex datasets, benefiting big data applications like agricultural data mining [26, 27].

MMDBs address schema inference challenges in NoSQL databases, where the absence of a predefined schema complicates data structure understanding and query formulation [2]. They streamline data management processes, ensuring data integrity and facilitating rigorous evaluations of data management systems [28, 17]. As organizations increasingly value integrated data systems, MMDBs are crucial for effective data management strategies. The BigDAWG polystore architecture exemplifies this by integrating and querying diverse data sources across different database systems, addressing the challenge of managing complex datasets that do not fit neatly into a single database engine [29, 30].

2.2 Evolution of Database Systems

The evolution from traditional RDBMSs to NoSQL and hybrid models reflects the need to manage increasingly complex and heterogeneous data. RDBMSs offer robust transaction management and data integrity but face limitations due to inflexible schemas, hindering adaptability in dynamic environments [31]. This led to NoSQL databases, which provide schema flexibility and are optimized for distributed, large-scale data environments, meeting demands for high availability and scalability [8].

NoSQL databases like MongoDB and ArangoDB manage diverse data models, including document, key-value, column-family, and graph models, advantageous for scalability and high availability [32]. However, they introduce challenges such as the need for innovative query languages and consistency models differing from traditional SQL and ACID properties [33]. Efficient management of interconnected data is critical in domains like academic publications and social networks [25].

Hybrid database models integrate RDBMS and NoSQL paradigms, offering NoSQL flexibility while preserving RDBMS transactional integrity. JSONB, an extension of PostgreSQL, exemplifies this by enabling JSON data storage and manipulation within a relational framework, providing a unified platform for structured, semi-structured, and unstructured data management [13, 15]. The integration of advanced techniques, including machine learning, enhances these systems' capacity to manage complex datasets [34]. Despite advancements, optimizing performance and scalability remains challenging, especially in hybrid configurations where geographical distribution affects latency and throughput [19]. Schema evolution in multi-model databases presents obstacles, necessitating careful coordination [32].

Ongoing development of innovative database architectures is essential as data complexity and volume continue to grow. The transition from single-model to multi-model approaches in ecological modeling reflects the evolution towards more integrated and versatile database solutions [27]. The shift from traditional to model-based fuzzing methods highlights the need for advanced techniques in database management [12].

2.3 Data Integration and Polystores

Polystores and hybrid databases are crucial for integrating diverse data models in environments with heterogeneous data sources. The BigDAWG polystore architecture enables applications to leverage diverse databases' benefits while insulating them from complexities [33, 30]. This approach is vital for seamless data integration, especially when data is distributed across various technologies optimized for specific types or workloads.

Integrating diverse data models in polystores faces challenges like semantic heterogeneity, complicating querying and integration. Federated database architectures address these challenges by employing a global conceptual schema to encapsulate data heterogeneity and location, allowing simpler queries [22]. Polystores facilitate multimodal data integration, such as images and text, underscoring their relevance in managing complex data interactions [24].

Hybrid databases like MongoDB enhance data integration by supporting multiple data models on a single platform, crucial for applications requiring structured, semi-structured, and unstructured data integration. However, inefficiencies in data movement and processing across various models and engines remain challenges, particularly in polyglot persistence environments [4]. Performance evaluation frameworks, using benchmarks like BigDAWG, provide insights into polystore systems' capabilities in handling modern workloads, emphasizing efficient query evaluation for complex queries involving multiple models [29]. Integrating machine learning methodologies, including ensemble learning, offers promising avenues for enhancing multi-view data integration, enabling sophisticated analyses and insights [34].

Polystores and hybrid databases represent significant advancements in data management, providing robust solutions for integrating diverse data models. Despite challenges, ongoing research and development aim to improve efficiency, scalability, and fault tolerance amid increasing data complexity and volume [3].

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

Multi-model databases are pivotal in addressing modern application requirements by offering diverse functionalities tailored to complex data management needs. As illustrated in Figure 2, the hierarchical structure of these databases showcases the strengths and capabilities of MongoDB, ArangoDB, and JSONB. This figure highlights their primary features, advanced techniques, and applications, emphasizing their contributions to data integration, querying, and management within modern data environments. In the following sections, we will delve deeper into each database's unique attributes and how they collectively enhance data handling in contemporary applications.

3.1 Features and Capabilities of MongoDB

MongoDB stands out as a leading document-oriented database system, renowned for supporting various data models, including document, key-value, and graph [8]. Its schema flexibility, enabled by the BSON format, allows seamless management of semi-structured and unstructured data, ideal for dynamic environments [35]. Advanced querying is facilitated by a robust aggregation framework, supporting operations from simple retrieval to intricate transformations, essential for applications with demanding data processing needs [10]. The introduction of Mongolog further enhances adaptability by allowing Prolog-like query writing [6]. Scalability is ensured through sharding, crucial for rapid query execution in API development [36].

In cloud environments, MongoDB optimizes throughput and latency via cloud bursting techniques [37]. Its distributed architecture, featuring sharded clusters, enhances high availability and fault tolerance [38]. MongoDB supports transactions across various deployments, ensuring data consistency [39]. The MongoRaftReconfig protocol improves reliability by enabling safe node replacements without traditional log-based methods [11]. However, challenges like performing JOIN operations and manual transaction management may affect performance [36].

As illustrated in Figure 3, MongoDB's efficiency is demonstrated in applications like Tweet storage and analysis [40]. The figure highlights MongoDB's features and capabilities, focusing on data models and flexibility, scalability and performance, and advanced applications. It emphasizes MongoDB's support for document-oriented data models, scalability through sharding and cloud optimization, and its use in advanced applications such as tweet analysis and multi-model data management. The MM-cat framework enhances schema inference and data management capabilities [21]. The HDMS supports effective data access and preparation for machine learning models, showcasing MongoDB's schema flexibility [41]. An automated performance testing framework ensures consistent evaluations in cloud settings, providing essential insights into MongoDB's capabilities, especially in read operations compared to other NoSQL systems [42, 31].

MongoDB's comprehensive features position it as a leading solution for modern data management, offering versatility through support for multiple data models and robust integrity protocols [43]. Integration with systems like FluidMem enhances performance through memory disaggregation [44].

AI-generated, for reference only.

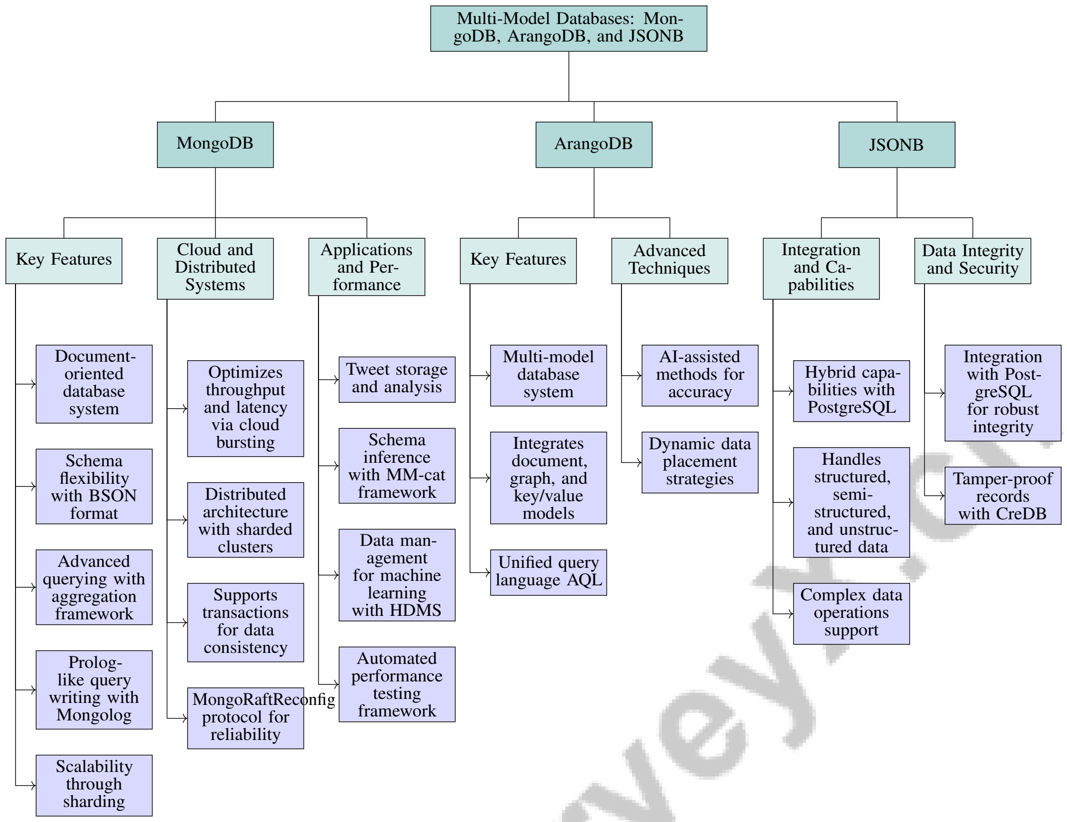
Multi-Model Databases: MongoDB, ArangoDB, and JSONB
MongoDB
ArangoDB
JSONB
Key Features
Cloud and Distributed Systems
Applications and Performance
Key Features
Advanced Techniques
Integration and Capabilities
Data Integrity and Security
Document-oriented database system
Schema flexibility with BSON format
Advanced querying with aggregation framework
Prolog-like query writing with Mongolog
Scalability through sharding
Optimizes throughput and latency via cloud bursting
Distributed architecture with sharded clusters
Supports transactions for data consistency
MongoRaftReconfig protocol for reliability
Tweet storage and analysis
Schema inference with MM-cat framework
Data management for machine learning with HDMS
Automated performance testing framework
Multi-model database system
Integrates document, graph, and key/value models
Unified query language AQL
AI-assisted methods for accuracy
Dynamic data placement strategies
Hybrid capabilities with PostgreSQL
Handles structured, semi-structured, and unstructured data
Complex data operations support
Integration with PostgreSQL for robust integrity
Tamper-proof records with CreDB

Figure 2: This figure illustrates the hierarchical structure of multi-model databases, focusing on MongoDB, ArangoDB, and JSONB. It highlights their primary features, advanced techniques, and applications, emphasizing their capabilities in data integration, querying, and management within modern data environments.

Advanced data management techniques, including machine learning, bolster MongoDB's ability to handle complex datasets, as evidenced by physics-ML models surpassing traditional methods in prediction accuracy [45]. Frameworks like AWESOME optimize analytical workloads involving relational, graph, and text data, providing a unified querying framework [3].

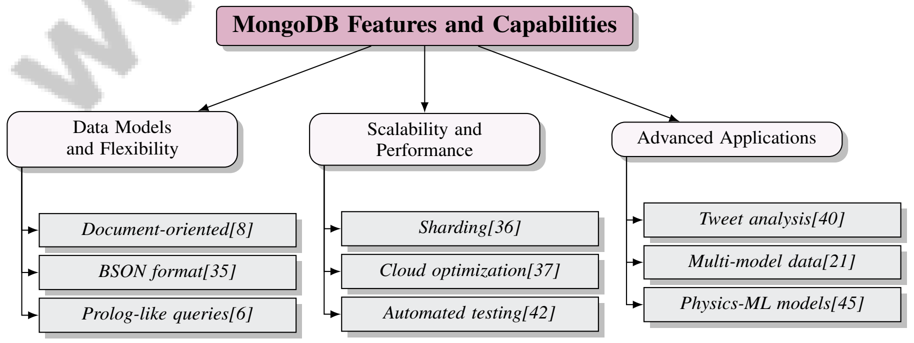

Figure 3: This figure illustrates MongoDB's features and capabilities, focusing on data models and flexibility, scalability and performance, and advanced applications. It highlights MongoDB's support for document-oriented data models, scalability through sharding and cloud optimization, and its use in advanced applications such as tweet analysis and multi-model data management.

AI-generated, for reference only.

3.2 Features and Capabilities of ArangoDB

ArangoDB exemplifies a multi-model database system adept at integrating and querying diverse data types, including document, graph, and key/value models. This flexibility is crucial for applications requiring complex data relationships [13]. Its architecture supports multiple data models within a single engine, essential for modern data environments [17]. The unified query language, AQL, facilitates efficient querying across different data models, enabling seamless execution of complex queries [5]. Benchmarks against other graph database management systems highlight ArangoDB's proficiency in managing varying loads and data sizes [5].

Advanced data management techniques, including AI-assisted methods, enhance accuracy and reduce user workload, particularly in diagnostic applications [13]. ArangoDB's architecture allows for dynamic adaptation of data placement strategies, optimizing performance based on real-time application behavior.

3.3 Features and Capabilities of JSONB

JSONB, an extension of PostgreSQL, showcases the hybrid capabilities of multi-model databases by integrating JSON data storage within a traditional relational framework. This integration enables effective utilization of SQL features, managing structured, semi-structured, and unstructured data. JSONB's ability to handle diverse data types is advantageous for real-time integration and querying [46]. Its support for complex data operations, such as those required in experiments utilizing JSON files for parameter definitions, underscores its flexibility [47]. Integration with PostgreSQL ensures robust data integrity and security, akin to systems like CreDB, which maintain tamper-proof records while allowing efficient data access [43]. This feature is vital for ensuring reliability and consistency in multi-model environments.

4 Unified Query Language and Schema Flexibility

Multi-model databases benefit significantly from a unified query language, which simplifies cross-model querying, schema inference, and data migration. This integration effectively manages diverse data representations like relational tables, JSON documents, and graph structures. By utilizing category theory, a framework has been developed to support the transformation and representation of these models, enhancing data management and ensuring seamless interactions across formats [1, 16, 48, 21]. Implementing such a language is crucial for improving interoperability and addressing challenges in multi-model environments. Schema flexibility complements this by allowing databases to dynamically adapt to evolving data structures and requirements.

4.1 Concept and Benefits of Unified Query Language

The unified query language is pivotal in multi-model databases, facilitating seamless data querying and integration across various models. This standardized framework enhances interoperability and simplifies heterogeneous data source management. For example, Mongolog provides formal semantics for queries in MongoDB's aggregation pipelines, utilizing logic programming for complex retrieval operations [6].

A unified query language also supports comprehensive data analysis by integrating structured, semi-structured, and unstructured data, as demonstrated by query augmentation methods that enrich local query results with related data from other databases [4]. The domain-specific language ADIL in AWESOME exemplifies this by supporting tri-model analytics and automating optimizations across data stores [3].

Standardizing query capabilities across systems mitigates vendor lock-in, akin to the associative array model, which provides a common mathematical foundation for relational models, thereby improving consistency and efficiency. Integrating advanced data management techniques, such as adaptive learning algorithms that adjust the influence of unlabeled data, further optimizes querying processes in multi-model databases [7].

The unified query language enhances data querying and integration by enabling seamless interactions among diverse models, addressing the complexities of managing heterogeneous data sources. This approach streamlines data migration, facilitates cross-model transactions, and optimizes query execution, ultimately enhancing operational effectiveness [1, 15, 21, 48, 49].

4.2 Importance of Schema Flexibility

Schema flexibility is essential in multi-model databases, enhancing their capability to integrate and manage diverse data types—structured, semi-structured, and graph-based—by allowing dynamic adaptation to evolving data structures. This adaptability addresses challenges posed by heterogeneous data, improving performance in analytical tasks by bridging traditional warehouses and modern data lakes, thus reducing ETL costs [21, 2]. This flexibility is crucial in environments with rapidly changing data types and volumes, enabling efficient management without rigid schema constraints.

Dani's approach [50] exemplifies schema flexibility by enabling inference directly from queries, eliminating external tools. The UDBMS platform [15] highlights managing multiple data models within a single framework, addressing traditional schema rigidity.

The concept of FluidMem [44] shows how schema flexibility enhances data management by allowing remote memory access without system modifications, underscoring the role of flexible schema architectures. The proposed NVCache admission policy [51] further demonstrates how schema flexibility can enhance cache performance through dynamic balancing of data admission and lookup rates.

Schema flexibility is critical for MMDBs, facilitating the integration and management of diverse data types without rigid schemas. This adaptability allows systems to respond to evolving data landscapes, bridging data lakes and traditional warehouses, reducing ETL costs, and enhancing flexibility. By leveraging schemaless models, MMDBs enable organizations to store and query heterogeneous data in its native form, addressing big data complexities [21, 2, 52, 48, 16]. This flexibility supports merging diverse data sources, enhances storage and query performance, and ensures databases meet modern data demands.

5 Data Integration and Hybrid Database Features

5.1 Enabling Data Integration through Multi-Model Databases

Multi-model databases are pivotal in integrating diverse data sources by employing mechanisms that facilitate seamless data integration across heterogeneous environments. These databases enhance integration and management efficiency by supporting multiple data models. Query augmentation methods, for instance, enrich local query results with related data from other databases, optimizing querying and integration processes [4]. Systems like AWESOME further enhance these capabilities by optimizing cross-engine dataflows and supporting complex analytical operations, addressing challenges inherent in traditional data management systems [3]. Such advanced techniques empower organizations to maximize their data assets by effectively managing and integrating diverse data types.

The strategies employed by multi-model databases, such as query augmentation and optimized cross-engine dataflows, are crucial for integrating various data sources. These mechanisms enable efficient and comprehensive data integration, allowing for seamless management of structured, semi-structured, and unstructured data across heterogeneous environments. This integration supports advanced analytics and decision-making processes, leveraging domain-specific languages and automated query optimizations to enhance scalability and runtime efficiency [20, 2, 53, 3, 16].

5.2 Hybrid Database Features: Combining NoSQL and Traditional Functionalities

Hybrid database systems, integrating NoSQL and traditional database functionalities, represent a significant advancement in data management. These systems leverage NoSQL's flexibility and scalability alongside the robustness and transactional integrity of traditional relational databases. This combination is beneficial for applications requiring high availability and managing diverse data types, including structured, semi-structured, and graph-based data, while ensuring consistency and reliability. Multi-model databases (MMDBs) facilitate native handling of this data variety, bridging the gap between data lakes and traditional data warehouses, reducing costs associated with ETL processes, and enhancing flexibility and scalability for complex analytical tasks in environments with large volumes of heterogeneous data [54, 55, 2, 39, 16].

MongoDB exemplifies this hybrid approach by supporting various data models essential for applications like social networking [36]. Its architecture addresses traditional database limitations, offering schema flexibility and efficient data processing capabilities. Future research should focus on optimizing MongoDB for specific applications to enhance its viability and address existing limitations [36].

Hybrid databases incorporate advanced features, such as support for JSON data formats, facilitating the integration of structured, semi-structured, and unstructured data. JSONB, an extension of PostgreSQL, exemplifies effective integration of NoSQL and relational systems by enabling manipulation of JSON data within a structured relational framework. This capability enhances the flexibility and scalability associated with NoSQL databases while preserving the transactional integrity and strong consistency of traditional relational databases. Consequently, JSONB supports dynamic schema management alongside reliable data transactions, addressing the evolving needs of modern data applications [56, 57, 50, 53].

The hybrid features of modern databases, combining NoSQL's adaptability with the reliability of traditional databases, provide a robust framework for comprehensive data management. These systems enable effective management of complex datasets, ensuring scalability, consistency, and flexibility in dynamic data environments. As demand for versatile data management solutions increases, hybrid databases are recognized for their ability to manage diverse data types—structured, semi-structured, and unstructured—within a unified framework. This capability addresses the complexities of modern data management, enabling seamless integration and querying of heterogeneous data sources, supporting a wide range of applications across various sectors. Hybrid databases improve data synchronization and integration, enhance flexibility, reduce data transformation costs, and bridge the architectural gap between data lakes and traditional data warehouses [15, 2, 58, 52, 37].

6 Challenges and Future Directions

Understanding the challenges and innovations in multi-model databases (MMDBs) is crucial for advancing the field. As data environments grow more complex, identifying obstacles to effective MMDB deployment and the innovative solutions addressing these challenges is vital. This section examines the specific difficulties MMDBs face, such as data integration, query processing, and system performance, while highlighting innovative approaches to overcome them. A thorough exploration of these challenges and innovations provides insights into the current landscape and future potential of MMDBs.

6.1 Challenges and Innovations

The deployment of MMDBs encounters several challenges that must be addressed to leverage their potential in managing diverse data environments. A primary concern is the absence of a universal data model that can encapsulate the inherent differences in multi-model data, complicating query processing and optimization [6]. This issue is compounded by the lack of a global schema in polystores, complicating data integration and access [4]. Integrating cross-modal data is further challenged by system incompatibilities and difficulties in developing effective connectors and translators, impacting the efficacy of analytics requiring comprehensive data integration [45]. Additionally, managing transactions and high RAM usage for indexing complicate MMDB adoption, with concurrency issues affecting performance and flexibility [5].

Innovative solutions are emerging to tackle these challenges. The UDBMS platform addresses critical issues of diversity, extensibility, and flexibility in multi-model data management [4]. The MM-cat approach offers a unified schema inference and data management framework, enhancing stability and efficiency [4]. In query processing, the Quantum-Inspired Keyword Search (QIKS) system significantly improves accuracy and efficiency, effectively addressing multi-model database query challenges [45]. Benchmarking tools like UniBench provide insights into MMDB performance, identifying bottlenecks in query processing and enabling performance comparisons across systems [5]. Innovative methods also address dataset replication challenges across multiple systems, which can increase write latency, by facilitating efficient query translation [12]. The lack of comprehensive benchmarks for performance comparisons across graph database systems highlights the need for further research [9]. Despite these challenges, ongoing research initiatives, such as the UniBench benchmark's development, are essential for enhancing MMDB capabilities [20, 17]. 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 will be shaped by key research initiatives aimed at improving scalability, performance, and integration capabilities across diverse data environments. A significant focus is on developing more expressive query languages and a universal data model to streamline query processing and enhance interoperability, leading to robust data management solutions [4]. Another promising direction involves optimizing graph algorithms for real-time applications and integrating graph databases with emerging technologies like AI and machine learning, enhancing the analytical capabilities of MMDBs in dynamic environments. Future research should also enhance multimodal aspects, as seen in multimodal machine learning frameworks, contributing to comprehensive data integration and analysis [45].

Exploring dynamic data generation methods, evaluating performance under different sharding strategies, and optimizing query execution plans in MMDBs are critical areas for future research, potentially leading to efficient and scalable data management solutions. Enhancing cross-system query mechanics and expanding supported data models and engines, as demonstrated in systems like AWESOME, are essential for optimizing performance and ensuring seamless data integration across platforms [3]. Future research should also focus on optimizing software compatibility and exploring additional applications for existing systems, such as the Octa-Cluster, to improve performance [9]. Extending support for various backend storage solutions, similar to FluidMem, presents additional research opportunities for more flexible and adaptable data management architectures.

Integrating additional columnar storage options and data analysis tools to enhance benchmarking frameworks is vital, as it could yield insights into MMDB performance and capabilities. Additionally, refining benchmarks to include real-world datasets and exploring more graph database management systems (GDBMSs) could validate findings and enhance MMDB applicability in diverse environments [12].

6.3 Performance Optimization and Scalability

Performance optimization and scalability are critical for MMDBs, given their role in managing diverse and complex data environments. MMDBs must efficiently handle large volumes of heterogeneous data while maintaining optimal performance, which presents significant challenges. Effective resource management is essential to ensure databases can scale horizontally and maintain 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, effectively managing resources and processing large datasets [59]. Innovative frameworks like TorchQL achieve up to 13x faster query executions than baseline systems, crucial for addressing MMDB challenges where efficient query processing across diverse data models is essential [60]. Rapid query execution is particularly important in applications requiring real-time data processing and analytics.

MongoDB showcases notable scalability and memory efficiency, particularly in managing larger object-centric event logs, providing advantages over traditional in-memory approaches [61]. The scalability of MongoDB and similar systems is vital for supporting applications involving extensive data storage and retrieval operations. Benchmarking tools like UniBench offer valuable insights into the performance characteristics of MMDBs, identifying areas for improvement and guiding future research efforts [62]. However, challenges persist, particularly in integrating and managing sensor data, where issues such as reliability, data overload, and integration complexity are prevalent, necessitating standardized integration methods for consistent and reliable data processing across diverse sources [27].

6.4 Standardization of Query Languages

Standardizing query languages in MMDBs is crucial for enhancing interoperability and usability across diverse data management systems. The lack of a universally accepted query language presents

AI-generated, for reference only.

significant challenges, as existing methods often fail to provide the necessary flexibility to accommodate user needs. For instance, in the Virtual Observatory context, methods such as SIAP and ADQL are not interoperable and lack the flexibility required for varied user requirements [63]. The complexity of certain query language fragments, such as those in MQuery, complicates standardization efforts. These complexities can hinder practical implementations, particularly in managing the expressivity and complexity of MongoDB's extended query language [10]. Addressing these challenges necessitates developing more streamlined and adaptable query languages capable of efficiently handling the diverse data models inherent in MMDBs.

A promising approach to enhancing query language standardization is the development of portable ontological expressions that improve query portability across different NoSQL implementations. This method allows for greater flexibility in working with diverse data field names without necessitating rigid standardization [64]. By facilitating query portability, this approach supports seamless integration and querying of heterogeneous data sources, enhancing the overall interoperability of MMDBs.

The standardization of query languages in MMDBs remains a complex but essential endeavor. By tackling challenges of interoperability, complexity, and portability, ongoing research aims to enhance data management solutions. These advancements focus on accommodating the diverse range of data types—structured, semi-structured, and graph-based—found in modern data environments. The adoption of MMDBs and knowledge graphs is promising, as these technologies bridge the gap between traditional data warehouses and more flexible data lakes, reducing the need for extensive ETL transformations and enabling schemaless models. Furthermore, comparative analyses of various data management systems reveal that while no single solution excels across all query types, integrated approaches can significantly improve performance when processing complex queries on large knowledge graphs. Ultimately, these efforts aim to develop robust and versatile data management frameworks that align with the FAIR principles of findability, accessibility, interoperability, and reusability in scholarly data [65, 2].

6.5 Security and Reliability

Security and reliability are paramount in deploying and operating MMDBs, given their role in managing complex data environments. The integration of 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 significant advancements in enhancing security, privacy, and data management, particularly in sensitive domains like healthcare, where traditional electronic health record (EHR) systems face numerous security challenges [66]. Reliability in MMDBs is challenged by the need to maintain consistent performance across diverse data sources and infrastructures. Limitations in existing systems, such as potential delays in alarm notifications and difficulties in standardizing log file formats across subsystems, underscore the need for more reliable monitoring and alerting mechanisms [67]. These challenges are compounded by the complexity of integrating occupant behavior data, which often lacks generalizability across contexts and building types due to specific data requirements [68].

In IoT applications, the reliability of sensor data and connectivity poses additional challenges. The performance of automated systems, such as irrigation systems, can be significantly affected by the reliability of IoT sensors and network connectivity, highlighting the need for robust mechanisms to ensure data integrity and system reliability [26]. Moreover, the complexity involved in understanding and implementing advanced mathematical concepts, such as those found in category theory, can hinder the practical application of security and reliability frameworks in MMDBs, particularly for practitioners unfamiliar with these mathematical underpinnings [48].

To effectively tackle the security and reliability challenges inherent in MMDB systems, a comprehensive strategy integrating advanced security frameworks, such as trusted execution environments and blockchain technology, alongside robust reliability mechanisms is essential. For instance, systems like CreDB leverage novel methods to ensure data integrity and auditability, including creating immutable transaction records, policy enforcement for access control, and timeline inspection for data history analysis. This multi-faceted approach enhances performance consistency across diverse data environments and addresses the unique complexities presented by heterogeneous data from various sources, such as sensors and relational databases, ultimately enabling more reliable and secure data

AI-generated, for reference only.

management in complex application scenarios [16, 43]. By addressing these challenges, MMDBs can enhance their utility and effectiveness in managing complex data-driven applications.

7 Conclusion

Multi-model databases (MMDBs) significantly reshape the landscape of contemporary data management by adeptly handling a variety of data types and facilitating seamless integration within complex systems. The examination of systems like MongoDB, ArangoDB, and JSONB exemplifies the innovative approaches that enhance efficiency and integration capabilities in data management. These databases are crucial in addressing the intricate challenges of data integration, which are fundamental for applications requiring comprehensive analytics and insights.

The evaluation of NoSQL databases, such as MongoDB and CouchDB, demonstrates their superior performance in CRUD operations compared to traditional relational databases, highlighting their efficacy in managing dynamic and large-scale data environments. This advantage is further supported by advancements in performance testing infrastructures that enhance the identification and management of performance regressions, thus optimizing database performance.

The incorporation of advanced data management techniques, as seen in automated systems for clinical outcomes, underscores the potential of MMDBs to drive innovation in data-centric applications. Furthermore, the ability of MMDBs to address structural uncertainties in ecological modeling underscores their importance in modern data management, offering robust solutions for handling complex datasets.

As research in MMDBs continues to evolve, these systems are poised to become integral to the future of data management. Their capacity to integrate diverse data models, coupled with efficient querying and management functionalities, positions them as indispensable tools for overcoming complex data integration challenges. The ongoing development of MMDBs is expected to foster innovation and efficiency in managing complex, voluminous datasets, thereby enhancing decision-making processes and supporting a wide range 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] Pavel Koupil. Modelling and management of multi-model data. 2022.

[22] 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.

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

[24] 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.

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

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

[27] 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.

[28] 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.

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

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.

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

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

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

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

[39] Verifying transactional consistency of mongodb.

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

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

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

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

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

[45] 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.

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

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

[48] A unified representation and tra.

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

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

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

[52] 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.

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

[54] Ciprian-Octavian Truica, Elena Apostol, Jérôme Darmont, and Ira Assent. Textbends: a generic textual data benchmark for distributed systems, 2021.

[55] Perspective.

[56] 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.

[57] Carlos J. Fernández Candel, Diego Sevilla Ruiz, and Jesús J. García-Molina. A unified metamodel for nosql and relational databases, 2021.

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

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

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

[61] 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.

[62] 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.

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

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

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

[66] 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.

[67] 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.

[68] 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.