International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 04 | Apr 2025 www.irjet.net p-ISSN: 2395-0072

A Review of Multi-Model Databases: Unifying Relational, Document, and Graph Data Models

KM. Anjali Kushwaha[1], Deepshikha[2]

[1]Master of Technology, Computer Science and Engineering, Lucknow Institute of Technology, Lucknow, India
[2]Assistant Professor, Department of Computer Science and Engineering, Lucknow Institute of Technology, Lucknow, India

---

Abstract - *With the advancement of data driven applications, database systems that jumped to the development of different data models efficiently were needed quickly. Over the past few years, there has been a major push to drive relational, document, and graph data models to merge into a unified system with the help of multi model databases. This review paper discusses the notion, architecture and implementation of multi-model databases in the context of being an integration platform for structured, semi-structured and graph data. We study the challenges of combining these different models together due to the heterogeneity in data representation, query languages, and performance tradeoffs. We also provide an overview of the benefits of using multi model databases as they provide increased flexibility, simplified data management, and cost efficiency but mention their disadvantages like increased complexity and performance overheads. A number of case studies are presented describing real world multi model databases with real work applications such as ArangoDB, OrientDB, and Microsoft Azure Cosmos DB. We also discuss the future trends for the following: (i) integration with AI and machine learning; (ii) further improvement on query optimization; and (iii) standardization efforts. In this paper, an attempt is made to provide a detailed insight into the multi model database, with a special focus on how it is suitable for modern data management, and how it can possibly change data storage, processing and knowledge extraction processes in the era of heterogeneous data.*

Key Words: Multi-Model Databases, Relational Data Model, Document Data Model, Graph Data Model, Data Unification, Cross-Model Querying, Database Architecture, Query Optimization.

1.INTRODUCTION

1.1.Background

The history of the database is filled with important milestones, and it really started with the time of the relational databases that triumphed at keeping control of structured data, structured after the example of tables, rows, columns. But with the arrival of the internet, social media and Internet of Things (IoT), the volume, variety and velocity, in terms of data, skyrocketed revealing the inadequacies of traditional relational systems. As a result, NoSQL databases rose to fill the gap through document, key-value, column-family, and graph data models that proved to be flexible enough in the changing technology environment. NoSQL databases were designed to resolve some specific use cases but the requirement to control many different data models inside a single application came more and more clear. Multi model databases takes a leap from here and it aims to unify different data models like relational, document and graph into one system. Multi model databases are important innovation in the database land scape which help to support the needs of modern applications, such as e commerce platforms, social networks, and IoT systems, which must co exist with structured, semi structured, and graph based data.

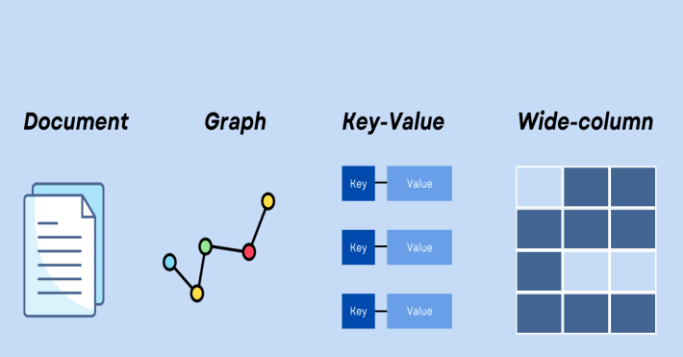

Figure-1: NoSQL Databases

1.2.Problem Statement

However, managing heterogeneous data models in separate systems is still a major challenge despite the great leap of database technologies. Organizations tend to use several database systems for the relational, document and graph data, which increases complexity, reduces the infrastructure costs and makes it difficult to keep data consistent and assured. This brings lack of a unified approach to querying and managing diverse data models and further exacerbates these challenges to lead data processing and analysis inefficient. With the increased complexity and data dependency of applications, there is a clear need of a solution that can do a constant integration and management of multiple data models in a single system.

1.3.Objectives

This paper aims primarily at investigating the unification of relational, document, and graph data models within the multi-model database. The paper analyzes the architectural principles and implementation strategies of these systems so that the issues they solve the problems of heterogeneous data management can be pointed out. Additionally, the paper attempts to evaluate the strengths and weaknesses of multi model databases by using their different features in order to if they are suitable for modern applications or not. This paper is about using the analysis of this exploration to offer insights into the potential and challenges of multi-model databases in the evolving data management landscape.

1.4.Scope of the Paper

In this paper, the relational, document and graph data models are focused on asthey are the most widely used and distinct paradigms of the database system. It also covers a review of existing multi model database system like ArangoDB, OrientDB and Microsoft Azure Cosmos DB who have pioneered the integration of these models. The paper also does not discuss other models like key-value or column families stores for keeping the discussion within a narrow scope about unification of relational, document and graph data. The paper aims at providing a thorough analysis of the advantages and disadvantages of multi model databases in the context of satisfying the needs of the modern, data intensive applications by concentrating on these models.

2.OVERVIEW OF DATA MODELS

2.1.Relational Data Model

In the 1970s, E.F. Codd introduced the relational data model that overthrew the data management world by storing data in structured tables of columns and rows. Based on the principles of normalization the data redundancy and its integrity too much is removed and thus this model is highly suited to the transactional systems that are for example banking, inventory management and enterprise resource planning (ERP). Relational databases are great at dealing with structured data where relationships between the entities are well understood and maintain consistency. Since they support ACID (Atomicity, Consistency, Isolation, Durability) transactions, they are integral to traditional data driven applications that rely on their reliability in situations when read and writes are important. Relational databases have rigid schema requirements, which may place a limit on the use of semi structured or fast changing data.

2.2.Document Data Model

This was the result of the presence of limitations in the relational databases to handle the semi structured and hierarchical data. This model stores data in flexible forms described by themselves like JSON or XML, each in which the document can have its special structure. Document databases are ideal for use cases such as content management systems, e commerce product catalogs and real time analytics since it provides lots of flexibility such that data schemas can vary or change as time goes by. Scalability and ease of development are placed ahead of predefined schemas and complex data structures are capable of being stored and retrieved. However, this lax enforcement of the schema can cause inconsistencies to arise, and querying nested data, or a graph of interrelated data can be demanding.

2.3.Graph Data Model

The graph data model is well designed for highly interconnected data and is therefore very suitable for social network, recommendation system and fraud detection kind of applications. Here, data is referred to as nodes (entities), edges (relationships) and properties (attributes) in this model. Traversing relationships deep into a graph is one area where graph databases excel, and performing queries across multiple levels of relationships is equally well suited for graph databases (e.g., finding the shortest path between two nodes, finding clusters in a network, etc.). In contrast to relational databases which use joins for relationships, graph databases are able to store relationships natively and query them more quickly and efficiently. That said, graph databases may not be able to efficiently deal with hundreds of millions or even billions of simple, nonrelational data, and their respective query languages like Cypher or Gremlin have a learning curve.

2.4.Comparison of Data Models

Similar scenario, each of the models has his little strengths and weaknesses. In an environment where one requires strict data integrity and complex transactions, the relational model does very well e.g. like in financial systems. Applications with evolving or semi-structured data make good usage of the document model: content management and real-time analytics. On the other hand, highly connected data such as in social networks or recommendation engines cannot be modeled with any graph model better than in a graph model. Although relational databases offer robust consistency and reliability, they don't work well on scale nor flexible. Schema flexibility and scalability come with some compromise on document databases when it comes to complex relationships. Relationship heavy queries are unparalleled in performance with graph databases, but simpler, non relational data may not perform so well in

graph databases. Therefore, understanding these tradeoffs is key to making the decision on choosing the right data model or to compensate for different data requirements on a single system, leveraging multi data model databases.

3.MULTI-MODEL DATABASES: CONCEPT AND ARCHITECTURE

3.1.Multi-Model Databases

Multi model databases are significant advancement in the database technology, which can support more than one data model in a single database system. In contrast to traditional databases that are designed to keep only one type of the data model (relational or document) in one place, multumodel databases allow to use one unified platform for operating with relational, document, graph and other types of the datamodel all at once. It is particularly suitable in modern applications dealing with various types of data in mixed structures which eliminates the necessity of having several specific databases and simplifies the whole process of data management.

3.2. Architecture

Broadly speaking, the multi-model databases have two types of architecture; unified storage layer and multi-engine approach. The unified storage layer takes the approach of storing data in a way such that different models can be interpreted and queried as a single underlying storage engine. This makes data management easier and all models consistent. Here, at the opposite end, the multi-engine approach implements the storage engines for each data model separately, and the integration occurs at a higher level, to present a unified interface. While this can improve performance for certain models, it can also cause some departure from consistency and synchronization. They must have an efficient mechanism to process and optimize queries in multi-model databases, as they must be able to process queries on various data models. Query translation, cross model indexing and adaptive optimization techniques are utilised so that the queries run successfully and gracefully.

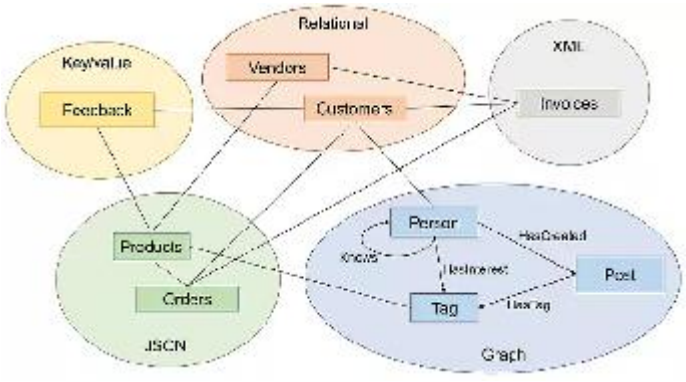

Figure-2:Architecture of Multi-Model Databases

3.3.Key Features

The versatile characteristics of multi model databases are characterized by several other features. Schema flexibility is one of the most notable being that developers are no longer constrained by rigid schemas when they store and manage data. Another important feature is cross model querying which allows the users to query data across different models with a single interface or query language. This is especially helpful for application looking at disparate data sources to gain some form of insight. Consistency and transaction management are also essential because multi model databases cannot sacrifice data integrity and ACID across models amidst heterogeneous data structures.

4.UNIFICATION OF RELATIONAL, DOCUMENT, AND GRAPH DATA MODELS

4.1.Challenges in Unification

There are many challenges associated with unifying relational, document, and graph data models within one and the same database system. Data representation is the main challenge and there is a big difference between representation in a relational database factory, a JSON like representation in a document model, and a graph model that uses nodes and edges. However, because of these difference, having a singular storage and querying mechanism turns out to be hard to create. Furthermore, the languages to query across models differ; SQL for relational, JSON based for document and graph specific languages such as Cypher or Gremlin for graphs. Another concern is performance trade-offs – optimizing queries across these different models can become inefficient due to conflicting optimizations of one model versus another.

4.2.Techniques for Unification

Different techniques are used to tackle these challenges by multi model databases. The conversion of data from one model to another was done by means of data model mapping and transformation when data is integrated seamlessly. For example, a relational table can represent the document structure or graph representation. Query languages integrated into the database system, such as SQL extensions or hybrid query languages (Iqbal et al., 2009b) allow users to query the data from all the models in a single query language syntax. Some databases also provide more than one query language, allowing the user to choose the one that fits his or her needs the best. These techniques act as an intermediary between the various data models and aid in the work with the heterogeneous datasets.

4.3.Case Studies

Furthermore, several multi model databases have already been successfully using relational, document and graph data models. For instance, ArangoDB provides a unified storage layer like many other systems, but it is designed such that users can switch between the three models seamlessly. Taking the strengths of both document and graph models, OrientDB combines it into a hybrid approach. To suit different data models, Microsoft Azure Cosmo DB supports various APIs such as SQL, MongoDB, Gremlin et., which is a multi-engine approach. This unification of diverse data models via a single system is shown to be feasible and beneficial through these case studies.

5.ADVANTAGES OF MULTI-MODEL DATABASES

5.1.Flexibility

Consequently, one of the main advantages of multi model databases is that they are very flexible. They can process various data types and structures, and are thus appropriate for a number of different applications. Multi model database provides us the mechanism of storing, managing and querying of data whether structured, semi structured or highly interconnected.

5.2.Simplified Data Management

Multi model databases support multiple data models in a single system and thus help provide a single integrated solution rather than multiple specialized ones. Simplifying data management reduces complexity because developers don't have to maintain three systems for relational, document, and graph data. It also minimizes data silos and also enhances data consistency as a plus.

5.3.Enhanced Performance

The databases designed for the multi-model aspects are more efficient and more effective in query execution over the different models. Cross model indexing and adaptive query optimization techniques guarantee an efficient query execution on the data types involved across multiple models. This comes in handy especially for data apps that need to bring real-time insights from one or more data sources.

5.4.Cost Efficiency

It is possible to save a lot of money by using a single multi-model database. Because organisations do not need to invest in multiple database systems, infrastructure and licensing costs are reduced. Moreover, multi-model databases consist of data management that has lower complexity resulting in the redu ction of cost due to operational and maintenance costs of an organization that is cost effective than modern applications.

6.LIMITATIONS AND CHALLENGES

6.1.Complexity

However, multi modeled databases have many benefits and also come with added complexity in design and implementation. It is important for the developers to understand how each of these data models interplays with each other in the system. Realising this increases the time and effort required to design and deploy applications.

6.2.Performance Overheads

However, the optimization techniques of multi model databases still may have performance tradeoffs. Overheads might be introduced by querying across models, especially because the data models may have adverse optimization requirements. Multi-model database system still needs to struggle in balancing performance across the models.

6.3.Lack of Standardization

Another limitation is the lack of standard query language and API. Multi-model databases, however, might differ in the way a single (or apart) data model(s) are unified, resulting in variation of queries written and executed. Such lack of standardization poses a serious problem for developers to transfer between systems, and to adopt known best practices.

6.4.Learning Curve

Another challenge is that multi model databases have a steep learning curve. However, the developers and administrators must learn multiple data models, query languages, and optimisation techniques. Faster learning curve needs to be dealt with as it slows down adoption and stretch the timeline for proficiency.

7.APPLICATIONS OF MULTI-MODEL DATABASES

7.1.E-Commerce

E-commerce applications fit well in the multi-model databases because they manage the seamless handling of myriad data types. A single multi model database can efficiently handle product catalogs that can include structured data, such as price and inventory levels, semi structured data, such as product descriptions, reviews etc, and graph such as product recommendations etc. Also, user data is stored and queriable in a unified way including profiles, purchase histories and preferences. By integrating B2C plans, e-commerce platforms can offer

personalized recommendations, manage inventory more efficiently and improve the personal customer experience.

7.2.Social Networks

When it comes to use case, social networks generate enormous volume of data where every data point is connected to millions of other data points. Social networks are an ideal use case for multi model database. For example, user profiles, which we usually consider as semi structured, can be saved as documents, and the relationship among users, e.g., friendships, follows, is explicitly represented as graphs. Structured and semi structured data can also be managed efficiently for activity feeds. Multi model databases which unify these data models allow social networks to answer complex queries such as finding mutual connections or recommending new friends with high performance and scalability.

7.3.IoT and Big Data

Various high velocity data streams, such as structured sensor data, semi structured logs, and graph like relationship of devices, are generated in many applications of the Internet of Things (IoT) and big data. This data is stored, processed, and analyzed in real time on a unified platform with multi model databases which helps in providing real-time insights and decision making. For instance, in a smart city application, traffic sensor data, weather station data and social media data can be combined and used for traffic flow optimization or condition of environment prediction. Multi model databases are the powerful tool for IoT and big data use cases also known for their flexibility and scalability.

7.4.Healthcare

In the healthcare space, multi model databases have several good use cases such as merging patient records, medical studies, and doctor histories. The data in patient records often contains structured information, such as lab results, and semi-structured information, such as doctor's notes; as well as graphlike relationships like family medical history. By integrating these models, healthcare providers can get one holistic view of patient health, enhance the accuracy of diagnoses and aid in tailoring treatment plans to individual patients. Also, multi model databases helps medical research in its ability to integrate various datasets like clinical trials and genomic data for analysis.

8. CONCLUSION

Multi model database represents a revolutionary manner to manage data by providing capability to combine relational, document and graph data model to an single system. This review paper has discussed the concept, architecture, and applications of multi-model databases, and elaborated on their pros, namely, flexibility, easier data management, and cost efficiency, and also their cons, namely, complexity and trade-off of performance. Modern applications such as e-commerce and social networks, IoT and healthcare can be addressed using the multi model database, as it integrates diverse data models.

The importance of multi model databases in the modern data management can not be overlooked, especially as the volume and variety of a data grows. The research and development activities for the future will focus on the creation of an AI and machine learning based back end system, query optimization, creation of standard query languages and standards for APIs, integration of scalability and performance, etc. These advancements will put multi model databases on the firm footing as they continue to be the crucial tool in managing the complex and extensive data needs of the future. Therefore, in summary, the multi model databases are on the verge of disrupting how data is stored, processed and analyzed and are going to enable new applications and insights in the big data and ai era.

REFERENCES

1. J. Lu and I. Holubová, "Multi-model databases: A new journey to handle the variety of data," ACM Comput. Surv., vol. 52, no. 3, pp. 1–38, 2019.
2. C. Zhang and J. Lu, "Benchmarking multi-model databases," in Proc. IEEE Int. Conf. Data Eng. (ICDE), 2019, pp. 1–12.
3. Z. Huang and Y. Chen, "A survey of multi-model databases: From relational to NoSQL and NewSQL," J. Comput. Sci. Technol., vol. 33, no. 6, pp. 1236–1250, 2018.
4. M. Stonebraker, "The case for polystores," Commun. ACM, vol. 61, no. 11, pp. 46–53, 2018.
5. A. Pavlo and M. Aslett, "What's really new with NewSQL?" ACM SIGMOD Rec., vol. 45, no. 2, pp. 45–55, 2016.
6. S. K. Garg and R. Buyya, "NoSQL databases: Critical analysis and comparison," in Proc. IEEE Int. Conf. Cloud Comput. Emerg. Mark. (CCEM), 2016, pp. 1–6.
7. P. Cudré-Mauroux et al., "NoSQL databases for RDF: An empirical evaluation," in Proc. Int. Semantic Web Conf. (ISWC), 2013, pp. 310–325.
8. K. Grolinger et al., "Data management in cloud environments: NoSQL and NewSQL data stores," J. Cloud Comput.: Adv., Syst. Appl., vol. 2, no. 1, pp. 1–24, 2013.
9. R. Angles, "A comparison of current graph database models," in Proc. IEEE 28th Int. Conf. Data Eng. Workshops (ICDEW), 2012, pp. 171–177.

10. D. J. Abadi, "Consistency tradeoffs in modern distributed database system design," IEEE Comput., vol. 45, no. 2, pp. 37–42, 2012.

11. J. Han et al., "Survey on NoSQL database," in Proc. 6th Int. Conf. Pervasive Comput. Appl. (ICPCA), 2011, pp. 363–366.

12. B. G. Tudorica and C. Bucur, "A comparison between several NoSQL databases with comments and notes," in Proc. 10th RoEduNet IEEE Int. Conf., 2011, pp. 1–6.

13. M. A. Bornea et al., "Building an efficient RDF store over a relational database," in Proc. 2013 ACM SIGMOD Int. Conf. Manage. Data, 2013, pp. 121–132.

14. K. Kaur and R. Rani, "Modeling and querying data in NoSQL databases," in Proc. IEEE Int. Conf. Big Data, 2015, pp. 1–6.

15. R. Cattell, "Scalable SQL and NoSQL data stores," ACM Comput. Surv., vol. 44, no. 4, pp. 1–42, 2011.

16. I. Robinson, J. Webber, and E. Eifrem, Graph Databases: New Opportunities for Connected Data, 2nd ed. O'Reilly Media, 2015.

17. S. Sakr and M. Gaber, Large Scale and Big Data: Processing and Management. CRC Press, 2014.

18. C. Vicknair et al., "A comparison of a graph database and a relational database," in Proc. 48th Annu. Southeast Regional Conf., 2010, pp. 1–6.

19. ArangoDB Team, "ArangoDB: A native multi-model database," ArangoDB Documentation, 2020. [Online]. Available: https://www.arangodb.com/.

20. Microsoft Azure Cosmos DB Team, "Azure Cosmos DB: Globally distributed, multi-model database service," Microsoft Azure Documentation, 2021. [Online]. Available: https://azure.microsoft.com/en-us/services/cosmos-db/.

21. M. Stonebraker and J. Hellerstein, "What goes around comes around," in Readings in Database Systems, 4th ed., MIT Press, 2005, pp. 2–41.

22. E. F. Codd, "A relational model of data for large shared data banks," Commun. ACM, vol. 13, no. 6, pp. 377–387, 1970. (Reprinted in 2000 for its historical significance.)

23. A. Thusoo et al., "Hive: A warehousing solution over a map-reduce framework," Proc. VLDB Endowment, vol. 2, no. 2, pp. 1626–1629, 2009.

24. N. Francis et al., "Cypher: An evolving query language for property graphs," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2018, pp. 1433–1445.

25. J. Pokorný, "NoSQL databases: A step to database scalability in web environment," Int. J. Web Inf. Syst., vol. 9, no. 1, pp. 69–82, 2013.

26. T. Neumann and G. Weikum, "The RDF-3X engine for scalable management of RDF data," VLDB J., vol. 19, no. 1, pp. 91–113, 2010.

27. A. Ghazal et al., "BigBench: Towards an industry standard benchmark for big data analytics," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2013, pp. 1197–1208.

28. M. Armbrust et al., "Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics," in Proc. CIDR, 2021, pp. 1–11.