International Journal of Core Engineering & Management
Volume-6, Issue-11, 2021 ISSN No: 2348-9510

EVALUATING DATA MODELING FLEXIBILITY: DYNAMODB'S KEY-VALUE STORE VS MYSQL'S RELATIONAL MODEL

Girish Ganachari, Rameshbabu Lakshmanasamy
Email: girish.gie@gmail.com

Abstract

Abstract: This paper reviews the flexibility of data modelling in an Amazon Dynamo DB key-value store versus a My Structured Query Language (MySQL) relational model on schema design, scalability, and performance. It discusses the strengths and weaknesses of both models: in Dynamo DB, adaptive access to diversified structures in exchange for weak transaction management and relational integrity; and with MySQL, strengths in transaction management and relational integrity exchanged for weak adaptively to data structure. The analysis brings out the contexts in which each system excels, guiding database selection based on application needs and use cases.

Keywords: Data Modelling, Amazon Dynamo DB, Key-Value Store, My Structured Query Language (MySQL), Relational Model, Flexibility, Scalability

I. INTRODUCTION

Data modelling refers to how data is handled or structured in modern applications and affects their performance and scalability. DynamoDB, being a NoSQL database, gives a flexible schema design; its key-value store makes changes to the data model easier and faster [1]. MySQL, on its part and being a traditional relational database, puts more emphasis on structured schema with strong transaction support [2]. Knowing the capabilities and constraints of these models enables organizations to select the appropriate database solution that best fits the needs of particular applications based on data handling requirements.

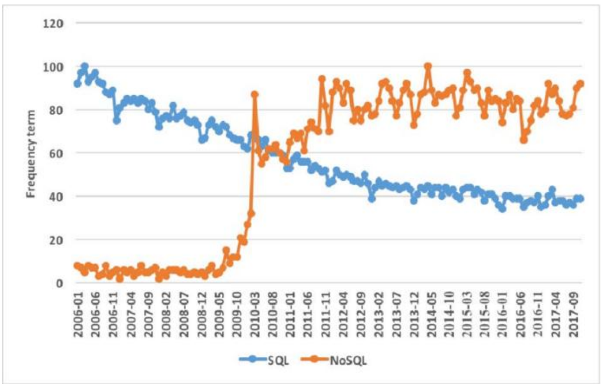

Figure 1: Google trends using NoSQL versus SQL terms
(Source: Beach, P.M., 2020)

The graph shows the usage frequency of SQL vs. NoSQL, which has considerably increased from 2005 to 2019. The trend using SQL around 2013 reflects a trend in flexible data modelling. This point to DynamoDB's schema flexibility and a key-value store advantage against the relational approach that MySQL has.

International Journal of Core Engineering & Management
Volume-6, Issue-11, 2021 ISSN No: 2348-9510

II. RESEARCH BACKGROUND

1. Evolution of NoSQL and SQL Databases

It underwent many changes until today, when a good number of models each suiting specific application requirements have been developed. Traditionally, SQL databases represented by systems such as MySQL have formed the backbone of data management because of structured storage, ACID (Atomicity, Consistency, Isolation, Durability) properties, and complex querying capabilities [3]. The relational model in MySQL is well-suited for structured schemas-based applications that need transactional integrity and relational data manipulation. With the need for flexible and scalable data management solutions, NoSQL databases, particularly key-value stores such as DynamoDB, were growing [4]. This kind of database is characterized by the architecture being schema-less and guaranteeing high throughput, processing large amounts of unstructured or semi-structured data.

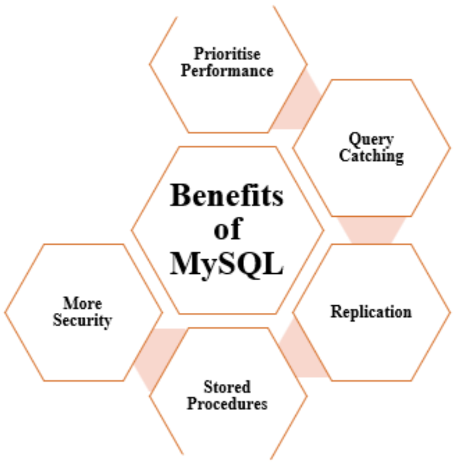

Figure 2: MySQL Benefits
(Source: Self-Developed)

2. Key-Value Store vs Relational Database: Definitions and Core Differences

DynamoDB's model is a key-value store, whereas MySQL's model is a type of relational one and quite strictly leads to data consistency and integrity [5]. This review paper attempts to assess the kind of flexibility extended to data modelling between both DynamoDB and MySQL concerning their scalability and performance from different use case scenarios. Employing discussing the strengths and weaknesses of each model of the database, the paper gives practical recommendations for the choice of database within each specific set of application requirements, to ensure optimal performance and scalability.

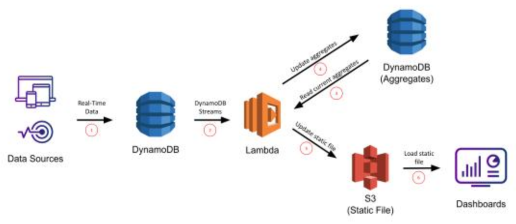

Figure 3: Real-Time Analytics on DynamoD
(Source: Ali et al., 2019)

3. Aim and Objectives

Aim: This review paper aims to assess the flexibility of data modelling in DynamoDB and MySQL.
Objectives:

- To assess the flexibility of data modelling in DynamoDB and MySQL
- To compare scalability and performance in different use cases
- To identify the strengths and weaknesses of each database model
- To provide practical recommendations for choosing the appropriate database based on application requirements

III. LITERATURE REVIEW

1. Flexibility in Data Modeling: Schema Design and Adaptability

Data modelling flexibility and database performance: according to the existing literature, there are visible distinctions between NoSQL and SQL database systems. Schema design and adaptability explained that MySQL, due to its non-flexible schema design, ensures data consistency; it enforces various constraints but more often than not, reflects a lack of adaptability to the fast-changing needs of the application [6]. In contrast, DynamoDB is schema-less by design. This allows for dynamic data structures that work quite well in applicative scenarios where the formats of data are subject to a lot of evolution [7]. However, the trade-offs come in the form of losing part of the intrinsic data validation that comes with relational models.

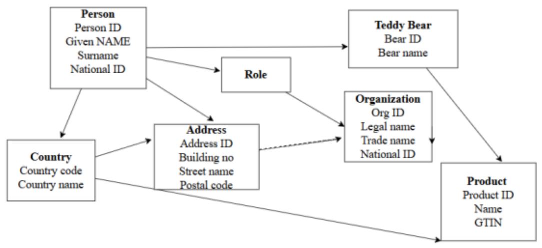

Figure 4: Flexible Data Models
(Source: Self-Developed)

2. Scalability and Performance: Handling Large-Scale Data

Performance metrics handling large-scale data reviews indicate that MySQL is very robust for transactional consistency but has problems with scalability since vertical scaling is used [8]. DynamoDB is horizontally scaled and stands at par in handling large volumes of data and high throughput, handling large-scale applications with varied data loads in a manner as required [9]. However, sometimes it gets affected by its eventual consistency model, which is not compatible with all scenarios.

A study investigated the adaptability of DynamoDB in fast-changing data environments. High flexibility, as compared to MySQL, has been observed in it. Their research proves that DynamoDB can support frequent schema changes, part of applications that require high agility, like content management systems [10].

However, another study investigated scalability challenges in MySQL versus DynamoDB. They concluded that while MySQL vertical scaling can support consistent transactional performance, DynamoDB horizontal scaling excelled more in handling large-scale data across the distributed system, though with some predictable eventual consistency issues in high-throughput scenarios 11].

3. Use Case Scenarios: Application-Specific Strengths
While relational databases like MySQL are preferred in applications involving complex queries and transactions heavy financial systems DynamoDB is very strong in scenarios that require high-speed data retrieval and storage, like in gaming and IoT applications. This brings out the segmentation clearly on the need to settle for a database depending on specific application requirements. Epic Games runs DynamoDB to handle the real-time game data for Fortnite, providing fast access to millions of players. Another case is Nest, which uses DynamoDB while handling huge amounts of sensor data from smart thermostats, facilitating real-time changes at breakneck speeds. [10].

4. Literature Gap: Areas Lacking In-depth Analysis
Areas needing further research there is inadequate research around the comparison of implementation challenges in the real world between DynamoDB and MySQL, particularly in the hybrid and multi-cloud cases [11]. This identifies the necessity of further research concerning the performance of these databases in different contexts of operation and how future use may be affected by emerging technologies [12].

IV. METHODOLOGY

Data Collection: Secondary Data Sources and Selection Criteria
The Secondary Data Sources and Selection Criteria used in this study include peer-reviewed journals, technical reports, industry white papers, and database documentation. The selection criteria were sources that had in-depth analyses with recent findings for relevance and accuracy [13].

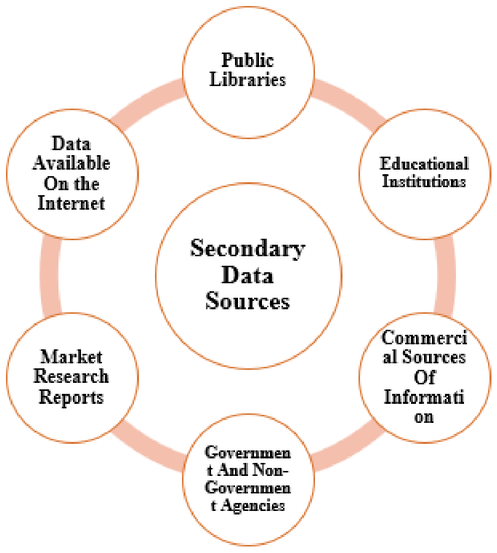
Public Libraries
Educational Institutions
Commercial Sources Of Information
Government And Non-Government Agencies
Market Research Reports
Data Available On the Internet
Secondary Data Sources

Figure 5: Secondary Data Source
(Source: Self-Developed)

V. RESULTS AND DISCUSSION

1. Assessment of Flexibility in Data Modeling

DynamoDB's schema-less nature allows for easy alterations and flexible data modelling, which is suitable for real-life applications featuring dynamic data structures; however, the downside is that the structure is more or less relational [14; 15; 16]. Correct schemas are one of SQL's great virtues. It provides high data integrity and allows complex queries, while MySQL's relational design is defined, as less flexible. Real-life examples show that generally, startups like DynamoDB because of the latter's flexibility, and big enterprises go for MySQL simply because they need the former [17].

2. Comparison of Scalability and Performance in Different Use Cases

As DynamoDB has the feature of capacity scaling, it can automatically partition data, making it suitable for high-volume situations like Netflix [18]. Adding ACID compliance by nature in MySQL also brings some overhead that may not support scaling perfectly. Most businesses needing a giant throughput with large data volumes often opt for DynamoDB, much like the gaming platform, whereas MySQL is the choice for applications needing complex transactions and data integrity, like financial systems.

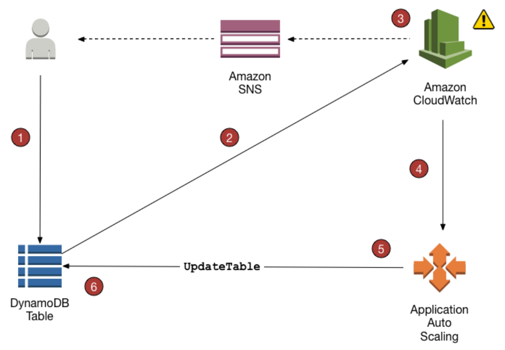

Figure 6: Dynamodb Auto Scaling
(Source: Farooq et. al., 2017)

3. Identification of Strengths and Weaknesses of Each Database Model

DynamoDB can easily scale and straightforwardly use the system. Still, it offers weak support for transaction capabilities, which is the reason why it cannot efficiently use this database for applications within the financial sector. MySQL strongly supports transactions, and shows consistency in data, with this facility provided; it is a complicated system when implementing scaling. E.g., shopping Websites may prefer to use MySQL which "coils it" to ensure efficient reliable transactions, and social media applications would like to use Dynamo DB for quick access to data [19].

4. Practical Recommendations for Choosing the Appropriate Database

It is highly recommended to use DynamoDB for highly dynamic fast-changing data structures. MySQL would be the proper choice if applications needed complex querying and high transactional strong consistency [20]. This can be guided by consideration of the requirements at the application level concerning scalability needs versus data integrity [21]. A new tech startup would use DynamoDB for quick iterations, while established banks would most probably have used MySQL to provide secure transactions with reliability.

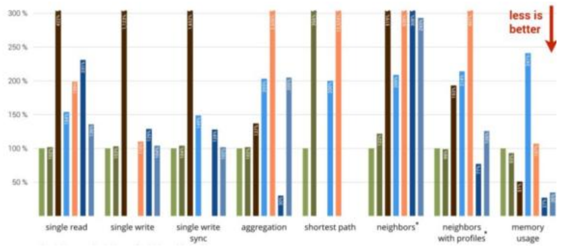

Figure 7: SQL vs. NoSQL Trends
(Source: Özsu et. al., 2020)

The graph contrasts SQL vs NoSQL frequency over time. The usage of the term NoSQL grew considerably from 2005 to 2019 and crossed SQL usage Frequency about 2013, which showed a trend towards flexible data modelling. This puts a greater emphasis on DynamoDB schema flexibility and key-value store advantage over MySQL's relational approach.

VI. CONCLUSION

DynamoDB's Strengths: This review thus presents evidence that DynamoDB is better at evolving data models and has superior scalability for high-traffic scenarios, while MySQL is better in complex transactions and data integrity.
Database Selection Based on Needs: From the results therefore, one can strongly infer that the selection of a database must be based on application needs; for example, using DynamoDB for cases of dynamic scaling or MySQL for transaction-heavy systems.
Limitations:

- Eventual Consistency: DynamoDB's eventual consistency can be limiting.
- Scalability: MySQL's vertical scaling is inadequate for large data volumes.
- Schema Rigidness: MySQL's fixed schema restricts flexibility.

VII. RECOMMENDATIONS

For choosing between DynamoDB and MySQL, consider DynamoDB for flexible schema design, high scalability, and rapid development. Choose MySQL where data integrity, complex transactions, and relational querying come first [22]. Some best practices are the use of indexing and partitioning for high performance in DynamoDB and normalizing schemas with indexing in MySQL for efficient querying [23]. Performance tuning and monitoring are regular activities required in both databases for efficient usage of resources and cost [24]

VIII. FUTURE WORK

Hybrid models of databases that integrate NoSQL and SQL for flexible and versatile data management are some of the future researches. Another important area involves investigating the impact of emerging technologies like AI-driven optimization and quantum computing on data modelling flexibility [25]. Case studies on real-world implementations and long-term performance analysis will go a long way in ascertaining practical application experience and sustainability of the different database models within varying environments.

REFERENCES

1. Gupta, A., Tyagi, S., Panwar, N., Sachdeva, S. and Saxena, U., 2017, October. NoSQL databases: Critical analysis and comparison. In 2017 International conference on computing and communication technologies for smart nation (IC3TSN) (pp. 293-299). IEEE.
2. Tang, E. and Fan, Y., 2016, November. Performance comparison between five NoSQL databases. In 2016 7th International Conference on Cloud Computing and Big Data (CCBD) (pp. 105-109). IEEE.
3. Ali, W., Shafique, M.U., Majeed, M.A. and Raza, A., 2019. Comparison between SQL and NoSQL databases and their relationship with big data analytics. Asian Journal of Research in Computer Science, 4(2), pp.1-10.
4. Malik, A.N., 2018. Exploring Multi-Model Features of Redis.
5. Lee, J., Wei, T. and Mukhiya, S.K., 2018. Hands-On Big Data Modeling: Effective database design techniques for data architects and business intelligence professionals. Packt Publishing Ltd.
6. Garba, M. and Abubakar, H., 2020. A comparison of nosql and relational database management systems (rdbms). Kasu Journal Of Mathematical Sciences, 1(2), pp.61-69.
7. Amghar, S., Cherdal, S. and Mouline, S., 2018, June. Which NoSQL database for IoT applications? In 2018 international conference on selected topics in mobile and wireless networking (mownet) (pp. 131-137). IEEE.
8. Lu, J. and Holubová, I., 2019. Multi-model databases: a new journey to handle the variety of data. ACM Computing Surveys (CSUR), 52(3), pp.1-38.
9. Sahatqija, K., Ajdari, J., Zenuni, X., Raufi, B. and Ismaili, F., 2018, May. Comparison between relational and NOSQL databases. In 2018 41st international convention on information and communication technology, electronics and microelectronics (MIPRO) (pp. 0216-0221). IEEE.
10. Hassan, M.A., 2021, December. Relational and nosql databases: The appropriate database model choice. In 2021 22nd International Arab Conference on Information Technology (ACIT) (pp. 1-6). IEEE.
11. Keshavarz, S., 2021. Analyzing Performance Differences Between MySQL and MongoDB.
12. Siddiqa, A., Karim, A. and Gani, A., 2017. Big data storage technologies: a survey. Frontiers of Information Technology & Electronic Engineering, 18, pp.1040-1070.
13. Messaoudi, C., Fissoune, R. and Badir, H., 2018. A performance evaluation of NoSQL databases to manage proteomics data. International Journal of Data Mining and Bioinformatics, 21(1), pp.70-89.
14. Moita, P.R.A., 2019. Modular and Adaptive Key-Value Storage Systems (Doctoral dissertation, NOVA University of Lisbon).
15. [15] Sutton, J. and Austin, Z., 2015. Qualitative research: Data collection, analysis, and management. The Canadian journal of hospital pharmacy, 68(3), p.226.
16. Palinkas, L.A., Horwitz, S.M., Green, C.A., Wisdom, J.P., Duan, N. and Hoagwood, K., 2015. Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. Administration and policy in mental health and mental health services research, 42, pp.533-544.
17. Roh, Y., Heo, G. and Whang, S.E., 2019. A survey on data collection for machine learning: a big data-ai integration perspective. IEEE Transactions on Knowledge and Data Engineering, 33(4), pp.1328-1347.
18. Farooq, H., Mahmood, A. and Ferzund, J., 2017. Do NoSQL databases cope with current data challenges. Int J Comput Sci Inform Secur (IJCSIS), 15(4).

19. Fernandez Canon, D., 2016. Evaluation of a data-model and a free-schemamodel for converting production shop floor datainto information using Relational and NoSQLdata management systems.
20. Mihai, G., 2020. Multi-model database systems: The state of affairs. Economics and Applied Informatics, (2), pp.211-215.
21. Petrovska, J. and Ajdari, J., 2019. Amazon's Role in the Field of Cloud Relational And noSQL Databases: A Comparison Between Amazon Aurora and DynamoDB. ISCBE 2019, p.214.
22. Singh, K., 2015. Survey of NoSQL Database Engines for Big Data (Master's thesis).
23. Özsu, M.T., Valduriez, P., Özsu, M.T. and Valduriez, P., 2020. NoSQL, NewSQL, and polystores. Principles of distributed database systems, pp.519-558.
24. Venkatraman, S., Fahd, K., Kaspi, S. and Venkatraman, R., 2016. SQL versus NoSQL movement with big data analytics. International Journal of Information Technology and Computer Science, 8(12), pp.59-66.
25. Chaudhry, N. and Yousaf, M.M., 2020. Architectural assessment of NoSQL and NewSQL systems. Distributed and Parallel Databases, 38(4), pp.881-926.
26. Zhu, S., 2015. Creating a NoSQL database for the Internet of Things: Creating a key-value store on the SensibleThings platform.
27. Beach, P.M., 2020. A Methodology to Identify Alternative Suitable NoSQL Data Models via Observation of Relational Database Interactions.Top of Form