Cloud Native NoSQL Database vs SQL Databases

Ashik Adnan
Deaprtment of CSE
Brac University
Dhaka, Bangladesh
ashik.adnan@g.bracu.ac.bd

S M Mahsanul Islam
Deaprtment of CSE
Brac University
Dhaka, Bangladesh
sm.mahsanul.islam@g.bracu.ac.bd

Fahim Hasnat
Deaprtment of CSE
Brac University
Dhaka, Bangladesh
md.fahim.hasnat@g.bracu.ac.bd

Nazifa Khanom
Deaprtment of CSE
Brac University
Dhaka, Bangladesh
nazifa.khanom@g.bracu.ac.bd

Md Sabbir Hossain
Deaprtment of CSE
Brac University
Dhaka, Bangladesh
md.sabbir.hossain1@g.bracu.ac.bd

Md Humaion Kabir Mehedi
Deaprtment of CSE
Brac University
Dhaka, Bangladesh
humaion.kabir.mehedi@g.bracu.ac.bd

Annajiat Alim Rasel
Deaprtment of CSE
Brac University
Dhaka, Bangladesh
annajiat@gmail.com

Abstract—**Software development companies faces a plethora regarding cloud migration possibilities because the cloud computing technologies advance for better accommodation of hosted software applications. Data management is one of the main issues. While there isn't yet a concept for cloud-native databases, research on cloud-native apps has inspired the development of extremely elastically scalable and robust stateless systems. Additionally, SQL (short for Structured Query Language) has become the de facto language for expressing database requests during the past few decades. SQL makes it simple to explain simple and mid-level data searches, while at the same time gives experienced SQL developers the flexibility and capacity to write complicated and very expressive queries. Even non-technical people could easily understand and write SQL when it was first created. In this research, we attempted to compare Cloud Native Databases to SQL Databases**

Index Terms—**NoSQL, SQL, Cloud Native, Spanner, MongoDB, DynamoDB, Aurora, PostgreSQL, BigQuery, BigTable**

I. Introduction

No-SQL databases are high-performance and non-relational data repositories. Column stores, key value stores, document stores, and graph databases are the four basic categories of NoSQL databases. There is currently no native Graph NoSQL database option available on Google Cloud. It does, however, support Janusgraph and Neo4j, two popular Kubernetes-hosted open source graph database solutions. The storage backend for the Kubernetes implementation of Janusgraph can be Bigtable. To satisfy these needs, NoSQL databases have grown in popularity. They use key-value databases or JSON-based databases, to name a couple of the main storage types, to store their data differently than relational databases do. PostgreSQL, which uses an OORDMS-based architecture and supports JSON, illustrates how these NoSQL databases can persist.

The backend enterprise's tried-and-true workhorse for years, SQL databases are at the core of all we do in this digital age. Early on in the 1970s, IBM developed SQL as a way to access their System R database system. PostgreSQL, MariaDB, and MySQL are just a few examples of SQL databases. Because of how deeply ingrained SQL databases are in our daily lives, these highly functioning and durable RDBMS will persist to be a mainstay of the business for later generations. In this paper, we compared SQL databases to Cloud Native NoSQL databases.

Identify applicable funding agency here. If none, delete this.

II. Literature Review

Goldschmidt, T. et al. [1] propose a cloud-native monitoring system architecture. First, they conducted testing on cloud infrastructures with up to 36 nodes and commercial workloads using three open source time-series databases (OpenTSDB, KairosDB, and Databus). KairosDB meets our original scalability and reliability hypotheses. They published a benchmark to evaluate its time-series database scalability and robustness using realistic industrial workloads. Maulidin, A. J. et al. [2] measure the synchronization system execution time and test data integration on web services. Data integration for 2 web services passed. The synchronization system's execution time is then measured. This work tries to automatically integrate two separate datasets online. They presented a Web services-based method for aggregating databases. Ghule et al. [3] suggested a Data Adapter system to promote a hybrid Relational/NoSQL database design. Database transformations can be synchronized and scaled-out Object Storage Systems can now use Data Analytics. A Data Analytics Layer executes in-place Map-Reduce on pre-existing data on an Object-based Storage Cluster. Through an intermediary layer that keeps track of ongoing transactions and handles concurrent transactions, it preserves the ACID capabilities of RDBMS while providing NoSQL's advantages. MongoDB and PosgresSQL were used as test databases in Hsu, TC. et al. paper [4]. The study used a local PostgresSQL database as a control group. They looked into the validity of database information and cloud computing. They wrote, read, and queried data in the test scenario. The outcomes support a novel cloud salesforce model. Heroku's subordinate mode aids in the comprehension of cloud architecture, design structure across various databases, and the validity of data consumption analysis. Relational databases are unable to handle high parallelism in read or write and large-scale spatial data organization and management, according to LI Chaokui et al [5]. They suggested a non-relational database for the storing of remote sensing image data based on Mongo DB, with high concurrent access and no reduction in read/write speed. They did this by using the debris type partition technique for picture division. Mongo DB's many concurrent access efficiency is noticeably better than the remote sensing picture storing method based on SQL Server database. Vial, G. et al. [6] claims that DBMSs provide non-relational systems, streaming-oriented technologies, and relational approaches for transactional applications. Users have a variety of DBMS technologies to choose from to meet their needs for cloud, embedded, real-time, social, mobile, analytics, and online applications. Gregory Vial gives a description of DBMS technology and providers in this issue of Software Technology, as well as a case study of an Internet application. SQL Injection is a risky hacking tactic, according to Gupta et al. [7]. In this paper, they proposed a novel SQL Injection Attack mitigation method. They used an integrated encryption mechanism to prevent SQL Injection Attack on web application databases. A database invader can inflict catastrophic damage and steal data. To mitigate these attacks, a hybrid strategy is proposed, using Advanced Encryption Standard (AES) upon login to fend off unauthorized access to databases and for the purpose of encoding the database Elliptical Curve Cryptography (ECC) is used so that without the key no one can access the database information. Comparing RDBMS, MongoDB, and graph-based No SQL databases, Sharma, M. et al. [8] stated that unexpected results were obtained as Neo4j took a bit longer response time than MongoDB and Postgre SQL. A closer look found that the cypher Query was the culprit. Neo4j's response time will change in real life when a user uses an application to access the Graph DB. They used the query access platform directly, and their results suggest MongoDB for large records.

III. Databases

A. What is Spanner?

Google's Spanner is a system for managing and storing SQL databases across several servers. In terms of functionality, [9] it supports things like cross-regional transactions, multi-site replication and failover, and strongly consistent reads. Google Ads' database, Google F1, makes use of Spanner. In 2012, Google's internal data centers were the original focus of the Spanner description [10]. A SIGMOD 2017 paper details the addition of SQL support for Spanner in 2017. In 2017, it was released as "Cloud Spanner" [11] as part of Google Cloud Platform. Massive volumes of structured data that can be changed often are kept in Spanner. Spanner uses synchronous replication to guarantee strong consistency and high availability for relational data while enabling users to execute any SQL query on it. The primary attributes of Spanner are: Transactions can span rows, columns, tables, and databases in a Spanner universe in addition to cells. Client control over data replication and placement is provided by automatic multi-site replication and failover. An object's duplicates are all made at the same moment and are highly similar to one another. Reads are highly consistent and data is versioned to allow stale reads, allowing clients to access older versions of the data within the boundaries of garbage collection windows. enables data reading and writing using an internal SQL interface.

B. What is MongoDB?

MongoDB is a cross-platform, open-source database management system. MongoDB uses documents that resemble JSON and have optional schemas. It is governed by the SSPL, a proprietary license that costs money. Field, range, and regex queries can be used with MongoDB. JavaScript functions that you define may appear in query results. A random sample of a specific size may be returned via queries. MongoDB document fields can be indexed by primary and secondary indexes. Replica sets in it boost availability [12], and a replica set includes duplicate data. Several servers can run MongoDB simultaneously, distributing the load or replicating data in case of hardware failure. Load balancing and data replication across several computers are features of the MongoDB file system GridFS. Grid file system is one of MongoDB's drivers. MongoDB provides content and file functions to developers. GridFS can be accessed with Mongofiles, Nginx, or lighttpd plugins. Every file chunk that GridFS stores is a separate document.

C. What is DynamoDB?

It is a document and key-value store which guarantees millisecond performance, regardless of the database size. For the use in web applications, DynamoDB is a multi-region, multi-master, fully managed, persistent database management system with built-in security, backup and restore capabilities, and in-memory caching. It has peak rates of up to 20 million requests [13] per second and is designed to handle more than 10 trillion requests per day. It was created in response to the demand for scalable, dependable cloud computing services and the focus on creating large-scale non-relational databases for Amazon.com. Managed, scalable, fast, durable and highly available, flexible, and low-cost services are just some of the benefits of using Amazon DynamoDB's NoSQL database service. Amazon DynamoDB is great for quickly prototyping and deploying a key-value store database that can scale to gigabytes or terabytes of data. DynamoDB's scalability and high availability make it suited for "always on" use cases with big transactional queries (reads and writes). DynamoDB's high operational expenses make it unsuitable for petabyte-sized data sets and frequent transactions. DynamoDB, a NoSQL database with a JSON-based query API, should be utilized when data models don't require normalized data with cross-table joins.

D. What is AWS Aurora?

It is a relational database engine that is fully managed and compatible with PostgreSQL and MySQL (Aurora). MySQL and PostgreSQL are two examples of open-source databases that also share the dependability and speed of commercially used databases without any hefty price tag. Aurora may be used with the same applications and tools as your current MySQL and PostgreSQL databases. For some workloads, Without requiring any changes to the current applications, Aurora has the capacity to provide up to 5x the throughput [14] of MySQL and up to 3x the throughput of PostgreSQL. A quick storage subsystem is present in this database management system. Both PostgreSQL and MySQL are compatible with it, and both systems' rapid distributed storage is geared for them. The storage underneath it grows automatically when more space is required. There is a hard limit of 128 terabytes on the size of data stored in an Aurora cluster. It also standardizes and automates database replication and clustering, which is one of the most difficult aspects of database implementation and maintenance.

E. What is PostgreSQL?

An open-source relational database management system, PostgreSQL, places a high value on extensibility and SQL conformance. Its original name, POSTGRES, reflected the fact that it was intended to replace the Ingres database, which was also developed at Berkeley. In order to more accurately reflect the fact that it was a SQL database, the project's name was changed to PostgreSQL in 1996. The development team opted in 2007 to keep using PostgreSQL and the Postgres alias after an evaluation. Aside from ACID-compliant transactions, PostgreSQL also supports materialized views, triggers, foreign keys, stored procedures, and automated view changes. It can manage everything, including single workstations, data warehouses, and Web services with tens of thousands of users. Available for multiple operating systems including macOS Server, Windows, Linux, FreeBSD, and OpenBSD, this database is the de facto standard for macOS Server.

F. What is BigQuery?

The query engine in Google BigQuery is built right into the database, making it a serverless data warehouse that can handle a large amount of data. The query engine can handle terabytes of data in seconds, and petabytes of data in just a few minutes. This performance can be attained without the need for index creation or maintenance. The capacity to scale and the lightning-fast performance of BigQuery are only two of its many impressive features. Because serverless, ad hoc [15] querying is so simple, the absence of infrastructure management is even more revolutionary because it makes way for novel approaches to work. The use of data to inform business decisions is becoming more common, and companies are promoting an environment in which information is shared freely across teams. BigQuery makes a large impact on the rate of innovation because it provides the technological means for a culture shift toward agility and openness.

G. What is BigTable?

BigTable is a NoSQL column-wide database, therefore it can store a lot of data with relatively few values in each column and row [16]. This system is also optimized for fast data reads and writes. The amount of information generated and stored increases as the number of people using popular apps rises. Since this is the case, commercially used databases are inadequate. For the purpose of managing large amounts of structured data, Google developed BigTable. It's original purpose was to manage Google's own data-intensive services like Google Earth and Google Analytics. The data items in BigTable are stored as key-value pairs in rows and columns. Multiple pieces of data can be stored in a single table cell, and column families are possible. Each piece of information also has a timestamp associated with it, which is used to track changes over time.

IV. MongoDB vs SQL

There are primarily two types of databases that are often used: SQL databases and NoSQL databases. There are some significant distinctions between these two databases that should be taken into account while choosing a database. Starting with storage architecture, relational databases like SQL are preferred over non-relational databases like NoSQL. According to these architectures, NoSQL databases have dynamic schema to deal with unstructured data, whereas SQL databases use Structured Query Language with a preset schema. Scalability-wise, NoSQL databases are vertically scalable, and SQL databases are horizontally scalable. SQL databases use table-based structures instead of NoSQL's document, key-value, graph, or wide-column stores. While processing multi-row transactions is the primary use case for SQL, unstructured data like documents and JSON are better suited for NoSQL.

The most popular SQL databases are MySQL and PostgreSQL, whereas the most popular NoSQL databases are MongoDB and ElasticSearch. MongoDB adheres to a cloud-agnostic architecture, placing an emphasis on operating without difficulty in any cloud environment and independent of any particular cloud platform's constrained functionality. Tools that are cloud-agnostic can operate without issue on any cloud provider, including Google, AWS, and Microsoft Azure, which facilitates smooth service portability. No matter how the tools are moved, the outcome will remain the same because they are not dependent on the platform's core bespoke features. The SQL database, on the other hand, is a cloud-native database that is reliant on a certain cloud environment. One can effortlessly benefit from all the advanced features and plugins within that cloud platform thanks to this integration.

A. *Ability to scale*

When scalability is a concern, SQL databases can typically only be scaled vertically, which simply refers to expanding the server's processing power. The cost of this option is high, and the upgrading or downgrading process takes time. Multiple read replicas in MySQL can improve user experience. MongoDB, unlike SQL, offers horizontal scaling through the use of sharding. Data distribution across numerous hosts is called sharding. While the application is being scaled, MongoDB divides a huge dataset into smaller datasets across numerous instances without experiencing any downtime. Using a shard key, sharding manages horizontal scaling among servers. Sharding, then, copies bits of data called shards across many replica sets as opposed to copying data entirely. Together, these replica sets make use of all the data.

B. *Accessibility and Dependability*

The architecture of data distribution also has an impact on the dependability and availability indices. While SQL databases were initially intended for isolated servers and later turned toward distributed databases to tackle risks, NoSQL databases, like MongoDB, were initially constructed with resilience in mind. In general, SQL servers make use of replication's power. Data is simply copied from the primary server node to the secondary server nodes during replication. In the event that the primary server malfunctions, this can help boost data availability and serve as a backup. A sharded cluster is created when replication and sharding combine, with each shard being replicated in turn to maintain the same high availability. While MySQL servers like AWS RDS don't have automatic replication for data availability and require explicit setup, MongoDB provides automatic sharding.

C. *Distribution of Transactions*

An operation on a single document in MongoDB is atomic. MongoDB employs embedded documents and arrays to capture relationships between data in a single document structure, obviating the requirement for multi-document transactions for most use cases, as opposed to normalizing across numerous documents and collections. However, there are some circumstances when atomic reads and writes to many documents are necessary. MongoDB supports multi-document transactions. Transactions can be used across numerous activities, collections, databases, documents, and shards when they are dispersed.

In 2020, Amazon RDS for SQL Server introduced distributed transactions with the help of Microsoft Distributed Transaction Coordinator (MSDTC). Using the AWS Directory Service for Microsoft Active Directory, you may run distributed transactions on domain-joined DB instances. With MSDTC, you can either promote MSDTC operating on the same host as the Client application to the position of Transaction manager, or you can execute the transaction using SQL Server as the Transaction manager using connected servers.

V. Aurora vs the default MySQL and PostgreSQL

When configuring new database servers with Amazon RDS, the Aurora database engine is selected. Aurora utilizes the well-known Amazon Relational Database Service (Amazon RDS) management and administrative features. The Amazon RDS AWS Management Console, AWS CLI commands, and AWS API activities can be used by Aurora to manage provisioning, patching, backup, recovery, failure detection, and repair. In contrast to individual database instances, Aurora administrative tasks frequently require large clusters of synchronized database servers. Your largest MySQL and PostgreSQL deployments will be easy to set up, operate, and scale thanks to automatic clustering, replication, and storage allocation. You can configure one-way replication or make and restore snapshots in order to import data from Amazon RDS for MySQL and Amazon RDS for PostgreSQL into Aurora. Applications currently running on Amazon RDS for MySQL and Amazon RDS for PostgreSQL can be switched to Aurora with the use of one-click conversion tools. The design of Aurora preserves the key benefits of relational databases' transactional consistency. It makes use of advancements in the storage layer to build a cloud-based database that can support contemporary applications without losing performance. Customers appreciate Aurora because it offers the performance and availability of commercial-grade databases at a fraction of the expense. Since its inception, Aurora has enjoyed the most rapid growth of any service in AWS's history.

VI. BigQuery vs Bigtable

First, BigQuery is an OLAP (Online Analytical Processing) solution. Due to its high query latency, BigQuery is best suited for queries with considerable workloads, such as OLAP reporting and archiving operations. OLTP-style queries are discouraged by the architecture of BigQuery. A simple read-write operation in BigQuery requires nearly 1.8 seconds, whereas the identical action in Bigtable requires only 9 milliseconds. The quick read-by-key and update operations of Bigtable make it the ideal database for OLTP workloads. In the data model, tables contain data and rows contain columns (Type Array or Struct). Similar to a persistent map, the structure enables the addition of columns to rows. The unique primary key of each entry is displayed in rows to facilitate reading and updating. Bigtable can be utilized to manage reporting and OLAP workloads due to its efficient support for key-range iteration. Use BigQuery if interactive querying in an online analytical processing environment is a top priority. Secondly, BigQuery enables us to do complex, analytical SQL-based searches on massive datasets. Nevertheless, users might utilize NoSQL approaches. For performance reasons, it is recommended to denormalize data while establishing schemas and importing data into BigQuery. In contrast, the Bigtable NoSQL database service does not enable SQL or multi-row transactions. This is not a relational database. Bigtable is therefore unsuitable for many applications; it should only be used for data sets that can be modified and are at least 1 Terabyte in size, as anything smaller incurs significant overhead. After being uploaded to BigQuery, data is also immutable; it cannot be deleted or modified for a defined period of time. If an existing record requires modification, the division must be redone. BigQuery is an "append-only" database that reduces data storage by removing partitions older than the chosen time to live. Contrary to. Bigtable organizes data into scalable tables, each of which is an ordered key/value map indexed by a timestamp, column key, and row key. This enables quick key-based lookups and data modifications. Typically, each row represents a unique object, and each column has unique data for each row. Read and write operations on rows are atomic, independent of the number of columns being read or written.

VII. MongoDB vs Spanner

The Google Cloud Spanner is a stable and consistent, horizontally scalable relational database management service. The most well-known components of the organization, includes Ads and Google Play, are powered by the core Google database in its external incarnation. A fundamental database model, relational databases, are the main focus of Spanner. One of the most widely used document stores available right now is MongoDB, which is available as a fully managed cloud service as well as for implementation on self-managed infrastructure. The document store database model is MongoDB's main database model, and the spatial database management system (DBMS), search engine, and time series DBMS are its subsidiary database models. Available from MongoDB are ScaleGrid for MongoDB Database and MongoDB Atlas. ScaleGrid for MongoDB Database is a fully managed hosting service that is available on a variety of cloud service providers as well as On-Premises. Utilize a centralized platform to automate your scaling, backups, and management. Across AWS, Azure, and Google Cloud, MongoDB Atlas offers unprecedented data mobility and distribution, as well as automation for resource and workload optimization. Linux, OS X, Solaris, and Windows server operating systems are all supported by the schema-free database MongoDB. Alternately, Spanner is a problematic database architecture. Go, Java, JavaScript (Node.js), and Python are among the programming languages that are supported. Server-side scripts are not supported by Spanner, although it does provide multi-source replication with three replicas for regional instances. ACID transactions are also supported by Spanner. MongoDB supports a wide range of programming languages, even though it is implemented in C++. MongoDB facilitates multi-source processes by using The server side of MongoDB Atlas Global Clusters employs JavaScript and supports Multi-document ACID Transactions with Snapshot Isolation and Source-replica replication.

VIII. DynamoDB vs Spanner

Amazon DynamoDB is a hosted, scalable database solution that stores data in Amazon's cloud. It functions as a document store and key value store database system. DynamoDB does not accept XML and is therefore schema-free. In contrast to SQL and server-side scripts, it allows partitioning, replication, ACID transactions, and secondary indexes. The Google Cloud The Spanner database service is a horizontally scalable relational database that maintains data consistency across all nodes in the network. It's the public face of the internal database that drives Google's flagship products like Google Ads and Play. Because of its emphasis on the relational paradigm, Spanner is a basic database system. Despite allowing Multi-source replication with three copies for regional instances, Spanner lacks server-side scripts. ACID transactions are also supported in Spanner.

IX. Conclusion

A literature review on cloud native databases and the methodologies they use has been carried out for the purposes of this research. In addition to that, we went through some of the fundamentals of well-known databases like as Spanner, MongoDB, DynamoDB, Amazon Aurora, PostgreSQL, BigQuery, and BigTable. Despite this, in the relevant literature, we have pointed out certain significant difficulties that might be encountered when developing and using distributed Databases. Both of these databases have been compared, and the benefits and drawbacks of each have been outlined for you. In a later phase of this research project, an extension of the current investigation, we intend to do an analysis of the performance of these databases, as well as a full comparison of these databases, and we will then make some recommendations.

References

[1] T. Goldschmidt, A. Jansen, H. Koziolek, J. Doppelhamer, and H. P. Breivold, "Scalability and robustness of time-series databases for cloud-native monitoring of industrial processes," in *2014 IEEE 7th International Conference on Cloud Computing*, 2014, pp. 602–609.

[2] A. J. Maulidin, F. Renaldi, and F. R. Umbara, "Online integration of sql and no-sql databases using restapis: A case on 2 furniture e-commerce sites," in *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, 2020, pp. 261–266.

[3] S. Ghule and R. Vadali, "Transformation of sql system to nosql system and performing data analytics using svm," in *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, 2017, pp. 883–887.

[4] T.-C. Hsu, D.-M. Chang, and H.-J. Lee, "The study of application and evaluation with nosql databases in cloud computing," in *2014 International Conference on Trustworthy Systems and their Applications*, 2014, pp. 57–62.

[5] C. Li and W. Yang, "The distributed storage strategy research of remote sensing image based on mongo db," in *2014 Third International Workshop on Earth Observation and Remote Sensing Applications (EORSA)*, 2014, pp. 101–104.

[6] G. Vial, "Different databases for different strokes," *IEEE Software*, vol. 35, no. 2, pp. 80–85, 2018.

[7] H. Gupta, S. Mondal, S. Ray, B. Giri, R. Majumdar, and V. P. Mishra, "Impact of sql injection in database security," in *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 2019, pp. 296–299.

[8] M. Sharma, V. D. Sharma, and M. M. Bundele, "Performance analysis of rdbms and no sql databases: Postgresql, mongodb and neo4j," in *2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, 2018, pp. 1–5.

[9] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. C. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, "Spanner: Google's globally distributed database," *ACM Trans. Comput. Syst.*, vol. 31, p. 8, 2013.

[10] J. Shute, M. Oancea, S. Ellner, B. Handy, E. Rollins, B. Samwel, R. Vingralek, C. Whipkey, X. Chen, B. Jegerlehner, K. Littlefield, and P. Tong, “F1 - the fault-tolerant distributed rdbms supporting google’s ad business,” in *SIGMOD*, 2012, talk given at SIGMOD 2012.

[11] D. F. Bacon, N. Bales, N. Bruno, B. F. Cooper, A. Dickinson, A. Fikes, C. Fraser, A. Gubarev, M. Joshi, E. Kogan, A. Lloyd, S. Melnik, R. Rao, D. Shue, C. Taylor, M. van der Holst, and D. Woodford, “Spanner: Becoming a sql system,” in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 331–343. [Online]. Available: https://doi.org/10.1145/3035918.3056103

[12] L. de Espona Pernas and E. Pustulka[, “Document versioning for mongodb,” 2022.

[13] M. Diogo, B. Cabral, and J. Bernardino, “Cbench-dynamo: A consistency benchmark for nosql database systems,” in *Performance Evaluation and Benchmarking for the Era of Cloud(s)*, R. Nambiar and M. Poess, Eds. Cham: Springer International Publishing, 2020, pp. 84–98.

[14] A. Verbitski, A. Gupta, D. Saha, M. Brahmadesam, K. Gupta, R. Mittal, S. Krishnamurthy, S. Maurice, T. Kharatishvili, and X. Bao, “Amazon aurora: Design considerations for high throughput cloud-native relational databases,” in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1041–1052. [Online]. Available: https://doi.org/10.1145/3035918.3056101

[15] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, “Dremel: Interactive analysis of web-scale datasets,” in *Proc. of the 36th Int’l Conf on Very Large Data Bases*, 2010, pp. 330–339. [Online]. Available: http://www.vldb2010.org/accept.htm

[16] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable: A distributed storage system for structured data,” in *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006, pp. 205–218.