DynamoDB: A Comprehensive Analysis of a NoSQL Database Solution

Anonymous
Rochester, USA

Anonymous
Rochester, USA

Anonymous
Rochester, USA

Anonymous
Rochester, USA

Anonymous
Rochester, USA

Abstract

Amazon, a E-commerce tech giant as we all know, is highly dependent on quick search results whenever a customer tries to buy a product. This gives them a competitive edge over other such companies in the market. To enable such fast query processing and handling of huge amounts of data, Amazon Web Services came up with DynamoDB. The reason behind this being that it offers high performance and scalability for managing unstructured data. This paper explores the core features and advantages of DynamoDB over traditional RDBMS. A detailed analysis will be done by us on factors such as data storage, indexing, query processing, transaction management, and security support. This will help us illustrate how DynamoDB serves as an optimal solution for modern applications requiring flexible and efficient data management. We will also present a case study to demonstrate the practical application and benefits of DynamoDB in a real-world scenario.

Keywords

Amazon, DynamoDB, RDBMS, querying, indexing, DAX, joins, AWS, S3, NoSQL, ACID, MySQL

ACM Reference Format:

Anonymous, Anonymous, Anonymous, and Anonymous. 2018. DynamoDB: A Comprehensive Analysis of a NoSQL Database Solution. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Overview

Amazon DynamoDB is widely used in today's time for various applications due to its features and advantages, namely scalability, low latency data access,flexible schema, cost effectiveness. Many businesses use DynamoDB for real time processing and analytics where it can process huge amounts of data efficiently. Another major application is its use in IoT processes for storage and querying purposes of sensor data. It is more helpful it can be easily integrated with other AWS services like Lambda which aids in serverless applications, S3 for storage purposes and many other such important services. DynamoDB has a good support for ACID transactions.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

DynamoDB has various features such as low latency, flexible data model, scalability and high availability. DynamoDB is scaled horizontally and vertically. Vertical scaling specifically helps in increasing the servers and horizontal scaling would mean addition of servers to mainly distribute the load. RDBMS would usually scale vertically and it also depends on the load the server would actually take. Next, the feature would be the data management. NoSQL works with unstructured data whereas RDBMS is more useful for structured data. Next would be cost effectiveness. Amazon DynamoDB is a price go pay scheme which means that we pay as per the usage of the platform. This is useful for not only independent use but also for business needs. While for RDBMS, the amount of data would not matter for the pricing that means that it is not a price go pay scheme. Some other features would include management of the data,fault tolerance and many more.

One of the running examples would be the usage of Ecommerce applications. The data management would include data storage which would store the data of each and every customer with some unique IDs. In this case, the schema would be quite flexible meaning that addition of data or maybe deletion would be possible without any issues which means that it would be dynamic.

2 Data Storage and Indexing

DynamoDB is no different than any other database.It supports documents as well as key value data schemas which aids in better representation of the data. It consists of tables, attributes and such other components. It supports the basic data types along with storage in the form of JSON or XML.It has two kinds of primary keys - a partition key and a composite key. Primary keys are unique values and can only hold one value hence it is scalar in nature. Composite keys include two attributes: partition and sort key which help in easing the querying that have similar partition keys but maybe different sort keys. In terms of data distribution, DynamoDB plays an important role in replication and partitioning. AWS works in various availability regions to give high availability due to which replication is necessary as and when required. Partitioning helps in handling large servers without any loss of data. In terms of indexing, DynamoDB gives both local as well as global indexes. Global index is useful with complex queries while local index is useful in range queries depending on a specific range of items that need to be queried.

Taking the example of the Ecommerce application, DynamoDB would have a table schema with proper indexing on any item if needed. Some of the basic fields the table would have is id, name, email and payment details. It would have at least two base tables Customers and Products. Other possible schemas could be Orders.

DynamoDB is schema less which would allow easy changes whenever needed which is not the case in RDBMS. It would even play an important role in scaling, meaning for example we can have more users during festivals or sales then we could easily scale and expand the database which would be rather difficult in terms of RDBMS.Hence, the application would manage various important components through DynamoDB which would rather be difficult in case of RDBMS or rather expensive.

3 Query processing and optimization

DynamoDB has a very good functionality in terms of querying. It can help us select multiple items with the same partition or hash but different sort key. Hence, it helps in direct access which helps in quicker retrieval of data. It also has a functionality with respect to a full table scan which would scan the complete table to get the records. This would be less efficient as the complete table is retrieved without any conditions. There are query filters that help in filtering records after querying hence reducing the data needed. Query Optimization plays a very big role in DynamoDB. Since it has an on demand functionality hence it can adjust the needs and capacities as per the load needed. This helps in optimizing the performance as well. DynamoDB consists of a DAX which stands for DynamoDB Accelerator which is an memory caching provider for big applications that help in enhancing the performance. Composite keys play a good part as they can handle both range queries along with partitions.

DynamoDB schema design can be enhanced by adopting a Nested Normal Form approach, originally applied to nested relational databases and XML databases. This approach ensures that DynamoDB schemas are free of unwanted redundant data, optimizing both data integrity and access speed[4].

Comparing it with RDBMS, DynamoDB has faster access to the same data that when used in RDBMS would slow the process. Scalability feature aids here as data is distributed while in RDBMS, it uses a single server which would cost a lot of time and inefficiency.Results show the comaprison of a nosql and a sql database[6] by performing a hypothesis testing for insertion where we see that the sql database takes 350 miliseconds while nosql takes 69. DynamoDB would be a perfect case in terms of scalability and low latency, a major plus would be its distributed system along with efficient indexes. RDBMS on the other hand, would make use of really complex queries for the same task and it would also face some issues related to scalability and also while dealing with heavy applications.

Coming back to the example of the Ecommerce application, the primary key of the customers table with proper indexing on the specific attributes would help in quicker retrieval of data. In RDBMS, it would take more memory and would be less efficient. Complex queries in RDBMS would also be time consuming and inefficient, more the number of joins more difficult to have a good performance. DynamoDB in retrieving data can speed up the operation to work faster than using only MySQL to retrieve data[2]. In DynamoDB, denormalization can be utilized to reduce the joins to a good extent and also DAX can be used for quicker retrieval of data.

Recent studies and developments highlight several advanced features and strategies in DynamoDB, including cost-based query rewriting which aims to minimize the financial costs of query executions by dynamically rewriting queries for cost efficiency[1]. This approach leverages algebraic expressions and histograms of attribute-values to predict and reduce execution costs effectively[1].

Furthermore, performance improvements have been noted in web applications utilizing DynamoDB as a cache database[2]. This setup significantly enhances read-and-write efficiencies compared to traditional MySQL databases, particularly in high-load environments such as online learning platforms during exam periods[2].

DynamoDB's design, which integrates flexibility in data modeling with its capability for high-performance read and write operations, makes it a robust choice for managing web-scale databases. Its architectural benefits are particularly evident when compared to traditional RDBMS, making it a superior choice for applications requiring high scalability, performance, and lower latency[1][2][4].

4 Transaction management and security support

DynamoDB offers robust transaction management and security support as the transactions adhere to ACID (Atomicity, Consistency, Isolation, Durability) properties. These ACID properties are responsible for transactions to be processed reliably, maintaining data integrity even in the event of errors, system failures or concurrent operations. DynamoDB supports various transactional APIs such as TransactWriteItems and TransactGetItems which are responsible for operations such as Put, Update, Delete and ConditionCheck. This enables us to group these actions (Put, Update, Delete, ConditionCheck) within a single atomic transaction. DynamoDB also ensures serializability and isolation, even in highly concurrent environments. This is taken care of by implementing two-phase commit protocols and timestamp-based ordering. Because of this approach; transactional integrity is ensured in important applications which include banking, finance companies, e-commerce business platforms etc., where data consistency is required at all times.

Security of data in a cloud computing environment is an important topic[5]. And thus, DynamoDB has strong security features to secure user data and assure compliance with industry standards. There are two components to the DynamoDB security framework; the security of the AWS cloud platform and the security of individual user accounts. Every user is subject to consistent enforcement by AWS of stringent security requirements, such as data encryption for data that is in-transit and at-rest, identity and access management (IAM) for controlling user permissions, and compliance with international security standards including SOC, PCI DSS, and HIPAA. AWS CloudTrail and Amazon CloudWatch are two more services that DynamoDB offers, along with additional auditing and monitoring features and support for conditional writes. It also provides predictable and super-fast performance with scalability, making it a fundamental storage platform within Amazon's system [3]. With the use of these tools, users may quickly detect any security risks, monitor database changes, and establish comprehensive access controls. Organizations use DynamoDB to store and manage sensitive data with confidence, knowing that it is safeguarded against breaches and unauthorized access by utilizing these strong security and compliance capabilities.

Strong security and transaction management features are also offered by relational database management systems, or RDBMS. Because RDBMSs like MySQL support ACID transactions by default, operations are completed with reliability. To manage concurrent transactions and guarantee data consistency, they employ techniques like locking and isolation levels. In addition, RDBMSs have a long history of managing intricate transactions and guaranteeing data integrity using joins, foreign keys, and other relational techniques.

On the other side, being a NoSQL database, DynamoDB provides greater flexibility for managing unstructured data and allowing for horizontal growth. But unlike typical RDBMS queries, it does not allow complicated queries that need joins across numerous databases effectively. Rather, to improve query efficiency and flexibility, DynamoDB makes use of local and global secondary indexes [1]. Data encryption, frequent audits, and role-based access control (RBAC) are commonly used in RDBMS security management. For these systems to reach the degree of automatic security compliance and monitoring that DynamoDB offers natively through AWS services, further settings and third-party tools are frequently needed.

DynamoDB excels in environments requiring high scalability and flexible data models, whereas RDBMSs are yet to tackle scalability, consistency, system efficiency, data collection and integration of data extraction[3]. RDBMSs are better suited for applications requiring complex transactional operations and relational data integrity.

5 NoSQL Database Application

We have implemented this application using a combination of AWS services to create a full-stack web application. This application deals with customers in form of a simple baning app for loan approval wherein the customer or the user can enter their details along with credit score and income to decide if they are eligible to get approved for a loan or not.

The key components and their roles are as follows:

Design and Implementation

(1) Frontend (Done with React App using Tailwind CSS):
The app's frontend is designed as a form created using React. This form will basically contain fields like Name, Email and two deciding fields like Credit Score and Income which are optional. These fields decide whether the loan will be approved or not, there are certain threshold that decide the decision as a response/ alert. These thresholds can be edited as per the admin control of the app.

(2) AWS Amplify:
We have used AWS Amplify to host our react application. It basically provides us with a CI/CD pipeline which make it easy for us to deploy updates from a GitHub Repository that we have created where all the app's files are stored. Basically we have went all in with respect to using AWS, and used almost the whole suite and not just DynamoDB to gain a good learning in terms of developing apps with NoSQL Database in backend.

(3) AWS Lambda:
To eliminate the management of server infrastructure and operational complexities, even the processing of form data is done on AWS using Lambda Function facility that they provide. It extracts the data from the form and saves it into our DynamoDB table that we created.

(4) AWS API Gateway:
To route the HTTP request that we are making (POST) to the lambda function, we are using AWS API Gateway. So the RESTful API's are being exposed using this to our application, so that it can interact with it.

(5) Amazon DynamoDB:
Coming to the DynamoDB part, it is a NoSQL database service which is used here to store our processed form data. Using it is beneficial to us as it is designed for high availability, scalability and performance which are a must in the banking space.

(6) AWS IAM:
We have set the IAM to manage permissions towards using the DynamoDB securely. So only the Lambda function that we made is given the permission with the help of AWS IAM to perform operations on our table. This increases the security towards our app data.

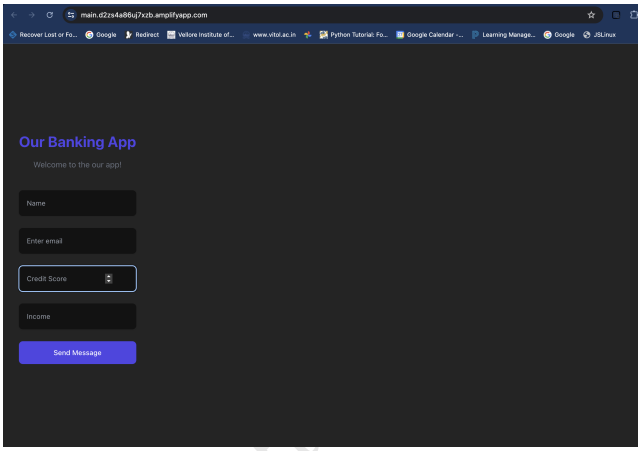

Figure 1: Home Page of the Web App

Justification for Choosing DynamoDB over other RDBMS:

(1) Scalability and Performance:
As everything is handled by AWS, we are stress free here and we just need to connect the dots of our application, other things like systems design, infrastructure, increase in number of users, spikes in page visits, all those factors are handled by AWS. So maintaining the application up online is AWS's job now.

(2) Flexible Schema:
Like any NoSQL based DB, DynamoDB too offers us the flexibility of requiring no predefined schemas, additional attributes can be added later to the data as per requirements and we don't have to edit the table for that, also attributes can be left empty in our forms which is advantageous as some people may choose to not enter any data in credit score and income or some people may just have an income and not a credit score.

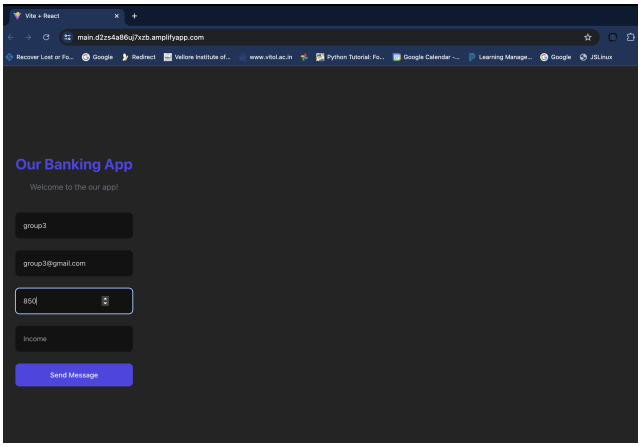

Figure 2: Banking App - Home Page

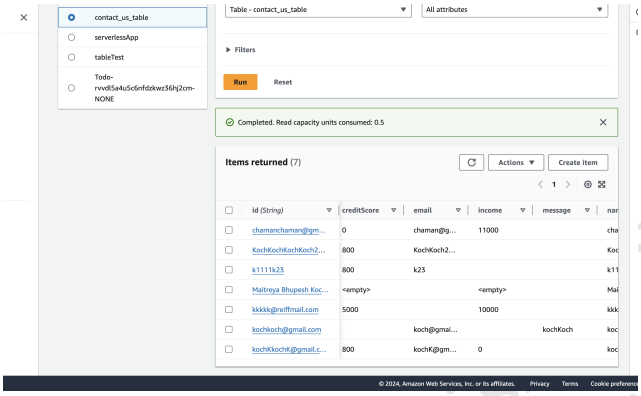

Figure 3: DynamoDB - Table where data is stored

There are other important advantages too that get undermined sometimes like DynamoDB having High Availability and Fault Tolerance, it being Cost Effective and also the fact that it is simply server-less for us, as all those factors are managed by AWS. This also increases our speed of development and thus reduces the time to market for web applications.

6 Final Remarks

DynamoDB excels as a highly efficient and scalable NoSQL database, ideal for modern applications needing rapid data access and flexibility with unstructured data. Its features, such as both horizontal and vertical scaling, comprehensive data management, and robust security, make it a strong alternative to traditional RDBMS. DynamoDB handles large data volumes with low latency and supports ACID transactions, ensuring reliability and performance under high demand.

Our e-commerce case study shows DynamoDB's real-world effectiveness. Its flexible data models, efficient indexing, and powerful query processing provide seamless data management and quick retrieval, crucial for competitive industries. The integration with

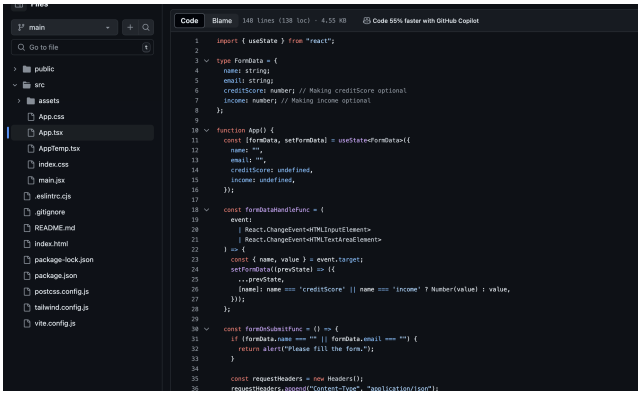

Figure 4: App.tsx file - From Our React Code Repository for web app hosted on GitHub

AWS services enhances its capabilities, offering a full ecosystem for developers to create and deploy scalable applications easily.

As cloud-based technologies evolve and the need for high-performance databases grows, DynamoDB's role in the market is likely to strengthen. Its adaptability and robust features make it a dependable choice for organizations leveraging NoSQL databases for dynamic data management.

7 Appendices

In conclusion, we have all contributed towards the completion of this project. Maitreya and Rutvi worked primarily on the implementation using of DynamoDB, while Shreya and Atharva worked mostly on the paper which detailed the aspects and features of DynamoDB. We had regular check-ins with each other where we helped each other to resolve our doubts and issues.

In the paper, Rutvi did the Overview and Data Storage and Indexing part while Shreya completed the Query processing and optimization, and Transaction management and Security Support part. Abstract and NoSQL Database Application part was done by Maitreya and Atharva.

References

[1] S. S. Chawathe, "Cost-Based Query-Rewriting for DynamoDB: Work in Progress", in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, 2019, pp. 1-3, doi: 10.1109/NCA.2019.8935057. *Keywords*: Indexes; Database languages; Relational databases; Histograms; Syntactics; Task analysis; DynamoDB; query rewriting; query optimization; cost estimation; NoSQL databases; cloud computing

[2] S. Tantiphuwanart, N. Tuaycharoen, D. Wanvarie, N. Pratanwanich and A. Suchato, "Performance Improvement on a Learning Assessment Web Application Using AWS DynamoDB as a Cache Database", in *2023 20th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Phitsanulok, Thailand, 2023, pp. 303-308, doi: 10.1109/JCSSE58229.2023.10201973. *Keywords*: Computer science; Databases; Web services; Pandemics; Education; Transforms; Time factors; Performance improvement; DynamoDB; Cache Database; NoSQL database; Learning Assessment Web Application

[3] S. Kalid, A. Syed, A. Mohammad and M. N. Halgamuge, "Big-data NoSQL databases: A comparison and analysis of "Big-Table", "DynamoDB", and "Cassandra"", in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, Beijing, China, 2017, pp. 89-93, doi: 10.1109/ICBDA.2017.8078782. *Keywords*: Google; Scalability; NoSQL databases; Peer-to-peer computing; Generators; Distributed databases; Big Data; BigTable; DynamoDB; Cassandra; RDMS; CAP

[4] W. Yin Mok, "A Feasible Schema Design Strategy for Amazon DynamoDB: A Nested Normal Form Approach", in *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Singapore, Singapore, 2020, pp. 903-907, doi: 10.1109/IEEM45057.2020.9309967.
Keywords: Unified modeling language; Databases; XML; Data models; Cloud computing; NoSQL databases; Relational databases; Amazon DynamoDB Schema Design; Nested Normal Form; XML Databases

[5] S. S. Chawathe, "Data Modeling for a NoSQL Database Service", in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, New York, NY, USA, 2019, pp. 0234-0240, doi: 10.1109/UEMCON47517.2019.8992924.
Keywords: cloud computing; NoSQL; databases; data modeling

[6] B. Sirish Shetty and K. Akshay, "Performance Analysis of Queries in RDBMS vs NoSQL", in *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, Kannur, India, 2019, pp. 1283-1286, doi: 10.1109/ICICICT46008.2019.8993394.
Keywords: Relational databases; Non-relational databases; Oracle; MySQL; NoSQL; MongoDB; RDBMS,