

# DynamoDB: a comprehensive analysis of a NoSQL solution



Amazon DynamoDB

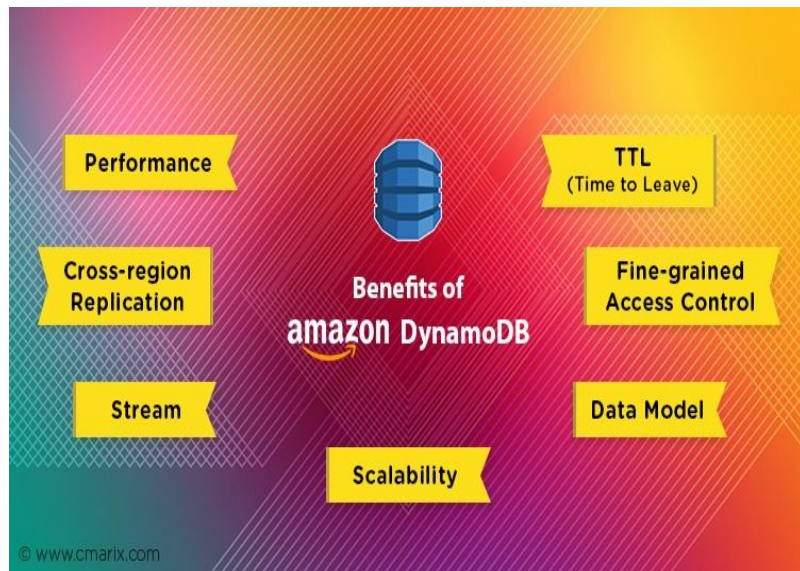
By Atharva Metkar, Maitreya Kocharekar, Rutvi Bheda, Shreya Sakpal

# Introduction

- Big tech companies today deal with huge amounts of data, all of which they need to store and process effectively.
- Seeing that traditional relational databases did not adapt well to the challenges of data at scale, AWS developed DynamoDB, based on the NoSQL paradigm.
- DynamoDB offers high performance and high availability for large amounts of data.
- Our project explains its various features and make a strong case for its use in robust applications.

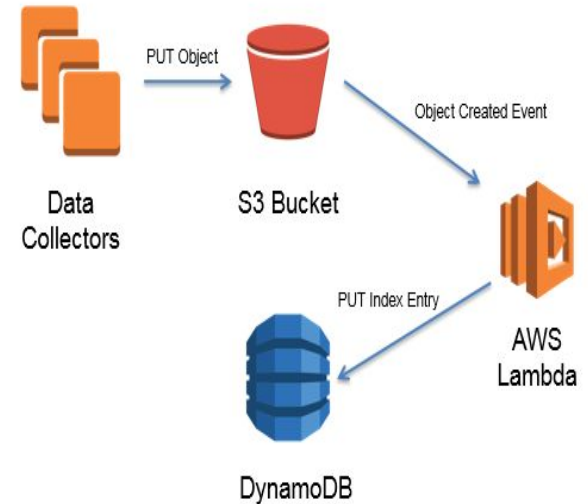
# Overview

- DynamoDB scales both horizontally and vertically, compared to traditional relational databases, which usually scale vertically.
- Relational databases do not work well with semi-structured or unstructured data, which is where a NoSQL database like DynamoDB shines.
- NoSQL ( non-only SQL) is an approach to database design that enables the storage and querying of data outside the traditional structures found in relational databases.
- DynamoDB operates on a pay-as-you-go pricing scheme, where costs are based on actual usage, making it cost-effective for both individual and business use. In contrast, traditional RDBMS systems typically do not follow this pricing model, as their costs are not directly tied to the amount of data stored or accessed.



# Data Storage and Indexing

- DynamoDB supports both document and key-value data schemas.
- The primary key structure includes a unique partition key and an optional composite key combining partition and sort keys.
- It partitions data across multiple servers, ensuring efficient handling of large datasets and preventing server overloads.
- Data replication across multiple AWS regions ensures high availability.
- It also easily scales horizontally and vertically to accommodate growing data storage needs.

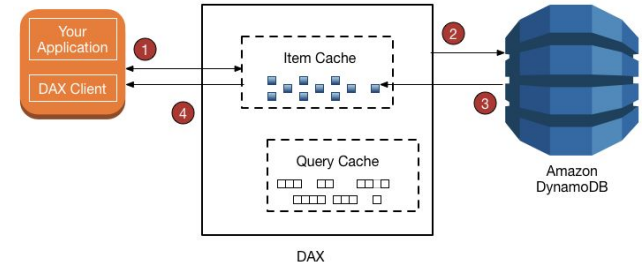


# Indexing

- Local Secondary Index (LSI) allows querying non-primary key attributes within the same partition, making range queries efficient.
- Global Secondary Index (GSI) enables querying on any attribute, facilitating complex queries and efficient data retrieval across partitions.
- Direct access to indexed attributes in DynamoDB speeds up data retrieval and reduces the need for full dataset scans, improving overall performance.
- The pay-as-you-go pricing model for indexes optimizes costs by allowing selective indexing based on application needs.

# Query Processing and Optimization

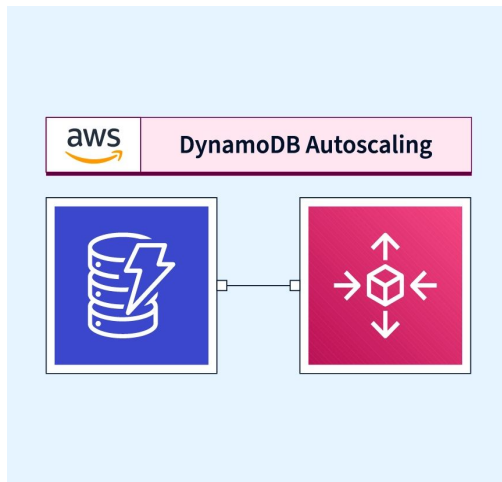
- DynamoDB Accelerator (DAX): Provides in-memory caching for fast data access speeds.
- On-demand Capacity Scaling: Automatically adjusts resources to handle varying loads seamlessly.
- Composite Keys: Simplifies queries across ranges and partitions with an efficient key management system.
- Cost-based Query Rewriting: Uses advanced algorithms to minimize query costs, optimizing financial efficiency.



# Query Processing and Optimization

## DynamoDB vs. Traditional RDBMS

- Partition & Sort Keys vs. Complex Joins  
Simplifies data retrieval, avoiding the complexity of RDBMS joins.
- Efficient Full Table Scans  
Less costly and disruptive compared to RDBMS, due to effective data partitioning.
- Post-Query Filtering  
Reduces overhead by filtering data after retrieval, unlike upfront processing in RDBMS.
- Dynamic vs. Manual Scaling  
Enhances performance without manual interventions required in RDBMS.
- Access Speed  
Non-reliance on disk-based storage allows faster data access under high loads.

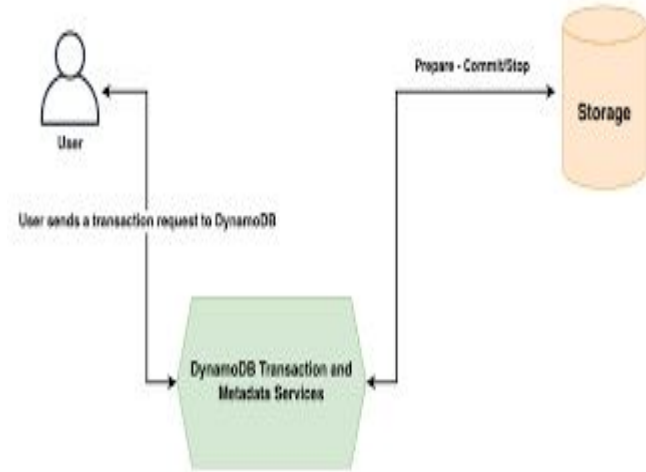


# Transaction management and security support

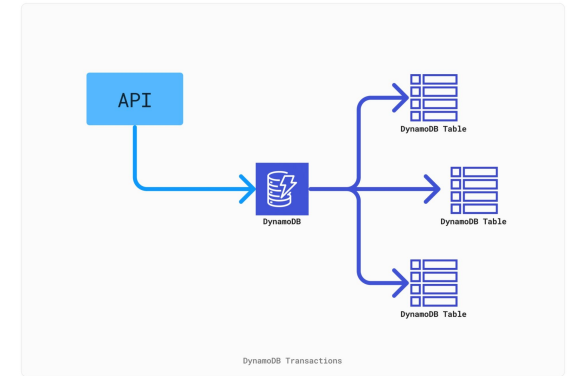
- DynamoDB supports ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring transactions are processed reliably and maintain data integrity during errors or failures.
- Transactions in DynamoDB are supported by APIs like TransactWriteItems and TransactGetItems, enabling grouped operations such as Put, Update, Delete, and ConditionCheck within a single atomic transaction.
- It guarantees serializability and isolation in concurrent environments through two-phase commit protocols and timestamp-based ordering.
- This robust transaction management is crucial for applications requiring high data consistency, such as banking, finance, and e-commerce platforms.
- Compared to RDBMS, DynamoDB offers greater flexibility for managing unstructured data and horizontal scaling, making it ideal for modern, scalable applications.



- DynamoDB's security framework includes data encryption for both data in-transit and at-rest. Identity and Access Management (IAM) in DynamoDB allows fine-grained access control, enabling precise control over user permissions and data access.
- DynamoDB complies with international security standards like SOC, PCI DSS, and HIPAA, ensuring that it meets stringent security and compliance requirements.
- AWS CloudTrail and Amazon CloudWatch provide real-time auditing and monitoring capabilities, enhancing security oversight and allowing quick detection of potential threats.



- RDBMSs such as MySQL also support ACID transactions using techniques like locking and isolation levels to ensure data consistency. They enable complex transactions by using joins. DynamoDB employs local and global secondary indexes to improve query efficiency, avoiding complex joins across multiple tables.
- DynamoDB excels in environments requiring high scalability and flexible data models, whereas RDBMSs are better suited for applications needing complex transactional operations and relational data integrity.



# Database Application

## WHY DynamoDB??

### Scalability and Performance:

- Handles high traffic and scales automatically.

### Flexible Schema:

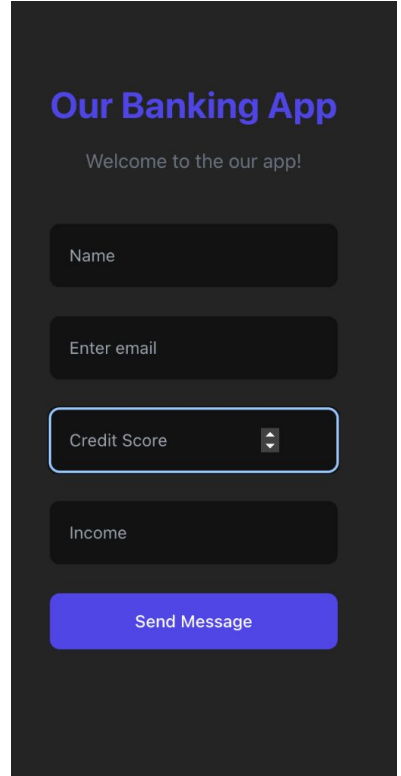
- No predefined schemas; allows for optional fields and evolving attributes.

### High Availability & Fault Tolerance:

- Reliable uptime and fault tolerance.

### Cost-Effective and Serverless:

- AWS manages infrastructure, reducing costs and development time.



The image shows a dark-themed mobile application interface. At the top, the title 'Our Banking App' is displayed in a purple font. Below the title, a welcome message 'Welcome to the our app!' is shown in a smaller, light gray font. The interface features four input fields: 'Name', 'Enter email', 'Credit Score' (which includes a dropdown arrow), and 'Income'. Each input field is a dark gray rectangle with light gray text. At the bottom of the form is a prominent purple button labeled 'Send Message' in white text.

# Database Application

These technologies are used to

- **AWS Amplify:** Hosts React App and CI/CD pipeline.

## Data Processing:

- **AWS Lambda:** Handles form data processing and stores it in DynamoDB.

## API Management:

- **AWS API Gateway:** Routes HTTP requests to Lambda.

## Database:

- **Amazon DynamoDB:** Stores processed data with high availability and scalability.

## Security:

- **AWS IAM:** Manages permissions and enhances security for DynamoDB operations.

Completed. Read capacity units consumed: 0.5

Items returned (7)

Actions Create item

< 1 >

<input type="checkbox"/>	id (String)	creditScore	email	income	message	name
<input type="checkbox"/>	<a href="#">chamanchaman@gm...</a>	0	chaman@g...	11000		cha
<input type="checkbox"/>	<a href="#">KochKochKochKoch2...</a>	800	KochKoch2...			Koc
<input type="checkbox"/>	<a href="#">k1111k23</a>	800	k23			k11
<input type="checkbox"/>	<a href="#">Maitreya Bhupesh Koc...</a>	<empty>		<empty>		Mai
<input type="checkbox"/>	<a href="#">kkkkk@reiffmail.com</a>	5000		10000		kkk
<input type="checkbox"/>	<a href="#">kochkoch@gmail.com</a>		koch@gmail...		kochKoch	koc
<input type="checkbox"/>	<a href="#">kochKkochK@gmail.c...</a>	800	kochK@gm...	0		koc

# Closing remarks

1. **High Efficiency and Scalability:** DynamoDB stands out as a highly efficient and scalable NoSQL database, perfect for modern applications that require rapid data access and flexibility with unstructured data.
2. **Advanced Features:** With support for both horizontal and vertical scaling, comprehensive data management capabilities, and robust security measures, DynamoDB offers a strong alternative to traditional relational database management systems (RDBMS).
3. **Performance and Reliability:** DynamoDB efficiently handles large volumes of data with low latency and supports ACID transactions, ensuring reliable performance even under high demand.

**4. Case Study Success:** The e-commerce case study demonstrates DynamoDB's effectiveness in real-world scenarios, highlighting its flexible data models, efficient indexing, and powerful query processing, which are essential for industries requiring quick data retrieval.

**5. AWS Integration:** The integration with AWS services provides additional benefits, offering a complete ecosystem that supports developers in creating and deploying scalable applications with enhanced capabilities.

**6. Future Prospects:** As cloud-based technologies continue to evolve and the demand for high-performance databases increases, DynamoDB's role in the market is expected to strengthen due to its adaptability and robust features.

**7. Dependable Choice:** DynamoDB's ability to manage dynamic data efficiently makes it a dependable choice for organizations leveraging NoSQL databases, ensuring effective data management in various applications.

# References

1. S. S. Chawathe, "Data Modeling for a NoSQL Database Service," 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}.
2. W. Yin Mok, "A Feasible Schema Design Strategy for Amazon DynamoDB: A Nested Normal Form Approach," 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}.
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"," 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. 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," 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}.
5. S. S. Chawathe, "Cost-Based Query-Rewriting for DynamoDB: Work in Progress," 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}.
6. B. Sirish Shetty and K. Akshay, "Performance Analysis of Queries in RDBMS vs NoSQL," 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICT), Kannur, India, 2019, pp. 1283-1286, doi: 10.1109/ICICT46008.2019.8993394. keywords: {Relational databases;Non-relational databases;Oracle;MySQL;NoSQL;MongoDB;RDBMS}.

THANK YOU!