TCSS 562 - Cloud Technology Sharing Presentation (11/30/2023)

DynamoDB

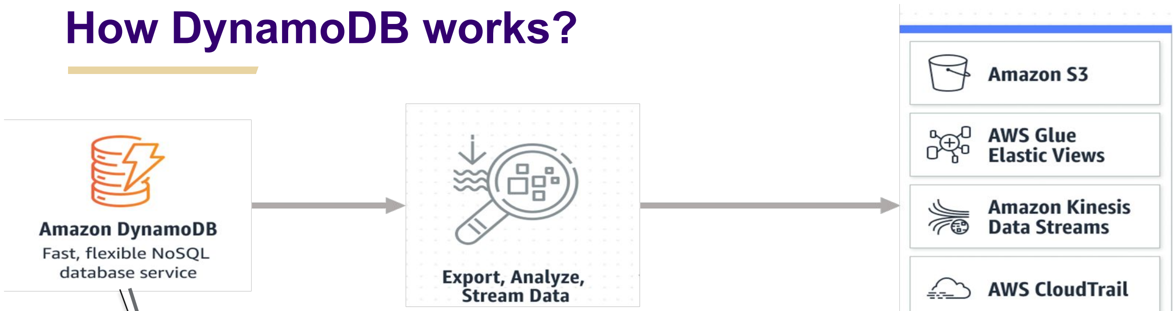
Vishnu Priya
Malavika Suresh
Alekhya Parisha

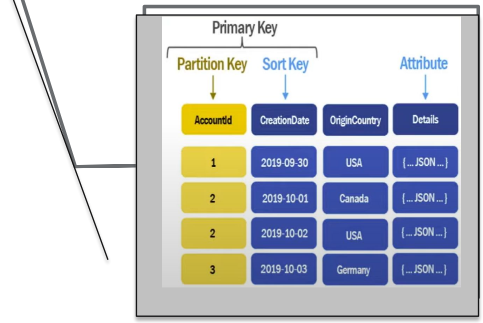
UNIVERSITY *of* WASHINGTON

Introduction

- DynamoDB is a fully managed, serverless, NoSQL database provided by Amazon Web Services.
- It is a NoSQL key-value store database.
- It offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data import and export tools.
- Provide seamless and predictable performance with the ability to scale both in terms of throughput and storage.

How DynamoDB works?


Amazon DynamoDB
Fast, flexible NoSQL database service
Export, Analyze, Stream Data
Amazon S3
AWS Glue Elastic Views
Amazon Kinesis Data Streams
AWS CloudTrail
Amazon CloudWatch


Primary Key
Partition Key
Sort Key
Attribute

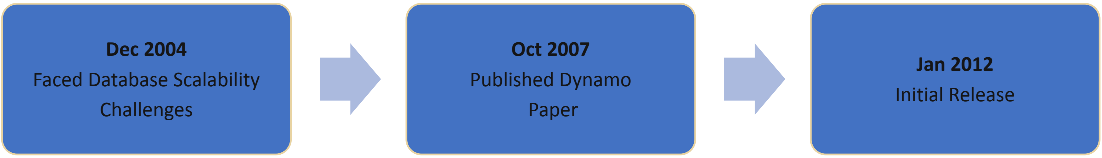
AccountId	CreationDate	OriginCountry	Details
1	2019-09-30	USA	{... JSON ...}
2	2019-10-01	Canada	{... JSON ...}
2	2019-10-02	USA	{... JSON ...}
3	2019-10-03	Germany	{... JSON ...}

DynamoDB table is a collection of items, and each item is a collection of attributes. Each item has a primary key

Competitors

Competitors	Data Storage Type	Managed/Unmanaged
MongoDB	Document	Fully Managed
Google BigTable	Wide-Column Key-Value	Fully Managed
Azure Cosmos DB	Document	Fully Managed
Apache Cassandra	Wide-Column	Depends on the service provider

History of DynamoDB



Dec 2004
Faced Database Scalability Challenges

Oct 2007
Published Dynamo Paper

Jan 2012
Initial Release

DynamoDB was introduced by **Werner Vogels** and his team.

Why NoSQL database?

- In traditional RDMS, all data stored on a single server or box which leads to scalability issues.
- Users turned to sharding, but this introduced additional complexities in data managing.
- The limitations of the single-box model led to the exploration of NoSQL database.

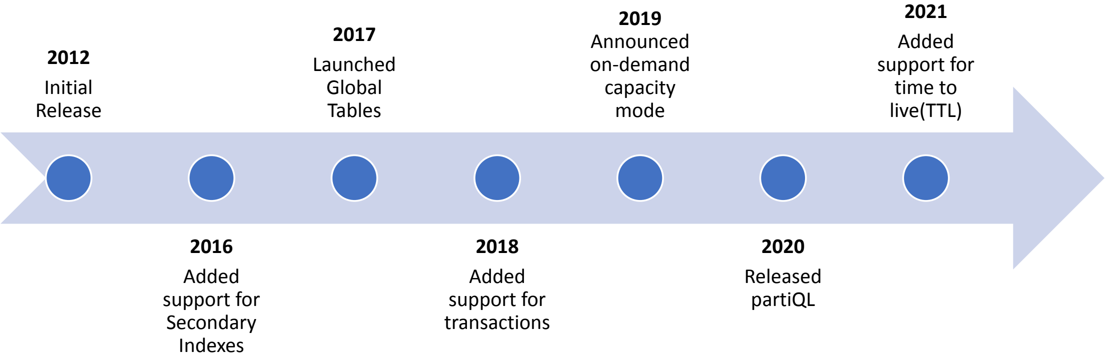
Why DynamoDB was invented?

During the 2004 holiday season, there were outages on the Amazon shopping website because of heavy traffic. This led them towards looking into an efficient way of storing data.

Motivation:

- Address scale and operational limitations of relational databases.
- Break free from expensive relational database licenses.
- Provide consistent performance at any scale.
- Build a fully managed, serverless NoSQL database.

Evolution of DynamoDB



2012
Initial Release

2016
Added support for Secondary Indexes

2017
Launched Global Tables

2018
Added support for transactions

2019
Announced on-demand capacity mode

2020
Released partiQL

2021
Added support for time to live(TTL)

Features

- Fully managed NoSQL database
- Auto-scalable and serverless
- Atomicity, Consistency, Isolation, And Durability (ACID) transactions
- Active-active replication with global tables
- Secondary indexes
- Fine grained access control
- Point-in-time recovery
- Backup and restore
- Integration with AWS ecosystem

Use Cases by industry

Duolingo

- Popular online language learning platform.
- The application requires higher read and write unit capacity
- They use DynamoDB to store and handle over 30 billion data objects on its web server.

Netflix

- Popular online streaming service.
- The application has 125+ million subscribers across the globe.
- They use DynamoDB to handle this subscriber’s big data.

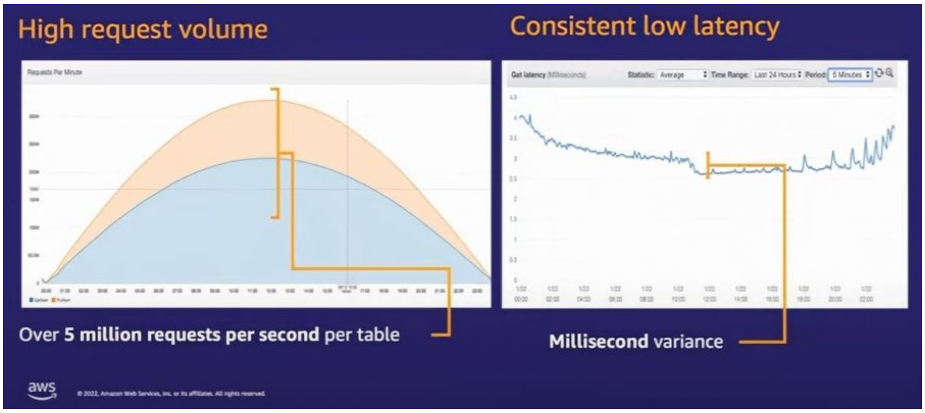
DynamoDB use cases (continued)

GE Healthcare

- Renowned company in the healthcare sector.
- The company is using Amazon’s DynamoDB service to manage the customer base.
- With easy cloud access, the service offers a better environment to the remote workforce who can share information and can stay up to date.

Advantages

- Performance and scalability - performance at any scale
- Fully Managed (Serverless)
- Seamless Data Replication and global distribution
- Secure
- A NoSQL DB with support for SQL interface (PartiQL)



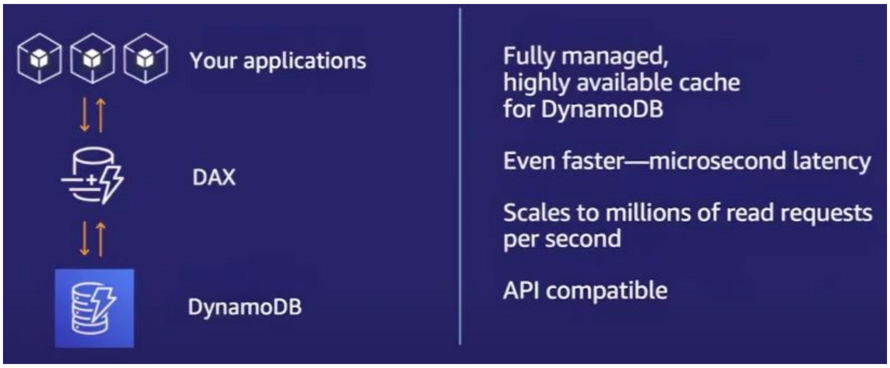
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Disadvantages

- Limited Querying Options
  - High reliance on indexing for querying tasks does not allow for querying if no indexes are available
- Difficult To Predict Costs
  - On-demand model could lead to unpredictable and expensive costs
- No support for On-Premise Deployments
  - Cannot support applications that require an on-premise database
- Limited Storage Capacities For Items
  - A limit of 400KB that includes the data as well as the attribute names used
- Unable to Use Table Joins
  - Difficult to query information from multiple tables

Usability

- DynamoDB is user-friendly in terms of its managed service nature and quick setup.
- However, the learning curve may be influenced by the NoSQL paradigm.
- Availability of extensive documentation, examples, and community support.
- DynamoDB Accelerator (DAX) makes usability very easy
- Faster adoption if prior experience with AWS services.



Fully managed, highly available cache for DynamoDB

Even faster—microsecond latency

Scales to millions of read requests per second

API compatible

Cost Discussion

WCU: Number of item writes per second x item size in 1KB blocks
RCU: Number of item reads per second x item size in 4KB blocks

On-Demand

Pay for WRU (Write Request Units) and RRU (Read Request Units).

Priced $1.25 per million operations and $0.25 per million operations respectively.

For strongly consistent operations:

One WRU = 1 write operation with item size up to 1KB

One RRU = 1 read operation with item size up to 4KB per second

For eventually consistent divide by 2, for transactional multiply by 2

Provisioned Capacity

Pay for provisioned number of RCU (Read Capacity Units) and WCU (Write Capacity Units).

Billed $0.00013 per RCU/h and $0.00065 per WCU/h.

For strongly consistent operations:

One WRU = 1 write operation with item size up to 1KB

One RRU = 1 read operation with item size up to 4KB

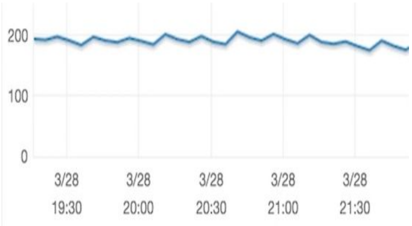
For eventually consistent divide by 2, for transactional multiply by 2

Pricing shown is for us-east-1, us-east-2 and us-west-2, which are among the cheapest regions

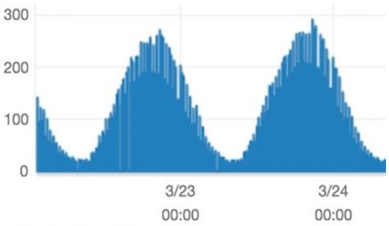
Cost Discussion (continued)

Provisioned	On-demand
● Pay per hour ● Capacity can be wasted ● Autoscaling can minimize waste, but could still lead to request throttling ● Best suited for predictable or consistent traffic patterns	● Pay per request ● No risk of throttling ● No need to monitor scaling or request traffic ● Best suited for unpredictable or random traffic patterns

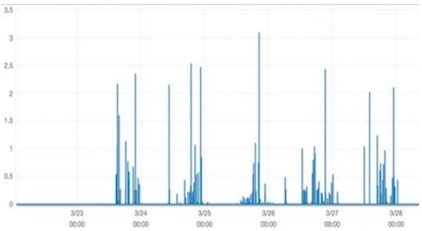
Use Provisioned



Use Provisioned + Autoscaling



Use On-Demand



Cost example

Consider a medium sized task management application where users can create, update, and complete tasks -

Throughput:

- You expect an average of 1,000 write requests per second (WCUs) for adding, updating, and completing tasks.
- Read requests are moderate, with an expected 500 read requests per second (RCUs) to display tasks to users.

Data Size:

- The total data size is estimated to be around 50 GB.

Provisioned Throughput:

- Given the moderate read and write demands, you provision 1,000 WCUs and 500 RCUs.

Now, estimating the monthly cost with these parameters-

- WCUs = 1000 WCUs * $0.00065 per WCU/hr * 720 hrs/month = $468/month,
- RCUs = 500 RCUs * $0.00013 per RCU/hr * 720 hrs/month = $47/month,
- Storage cost = 50 GB * $0.25 per month = $12.50 per month
- Total estimated monthly cost = $468 + $47 + $13 ~ $530
- This excludes the AWS free tier of 25 WCUs and 25 RCUs of provisioned capacity, 25 GB of data storage

This is only a simplified estimate but additional charges related to Data Transfer, Backups, DAX and Global Tables might apply depending on usage.

DynamoDB vs S3 price comparison

- An application receives on average 100 requests per seconds, with 2Kb payload and is required to persist it entirely. Also, each object saved is read 2 times. (200 WCUs and 200 RCUs)
- We can estimate the monthly storage cost for DynamoDB and S3 using the AWS calculator (us-east-1 region).
- S3 costs 1514$/month, whereas DynamoDB costs 830$/month using on-demand capacity, but it could drop to 173$/month (+360$ paid upfront) with reserved capacity.
- In the S3 case, the biggest cost part comes from accessing the service, the storage itself being under 100$/month.
- Now considering requests with different payloads - 5Kb, 10Kb, 20Kb and 50Kb
- Result - Once item size exceeds 5Kb and without using provisioned DynamoDB capacity, it becomes more convenient to store data in S3 than in DynamoDb.
- Therefore, understanding the requests pattern we'll have is crucial for using DynamoDB in a cost effective manner.

Conclusion

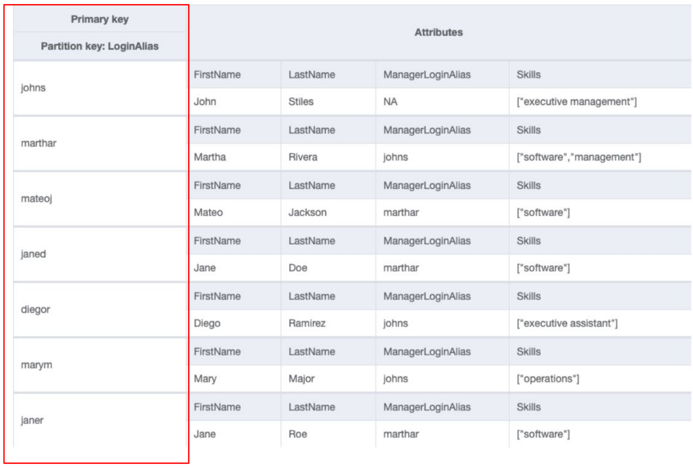
- DynamoDB emerges as a robust choice for database management in specific scenarios.
- It is particularly advantageous for those who have encountered scalability challenges with traditional databases, require high-performance handling of online transaction processing workloads.
- The database is well-suited for mission-critical applications demanding constant high availability without manual intervention.
- Cost-effective in certain use cases.
- Additionally, DynamoDB offers a compelling solution for users seeking a high level of data durability and facing challenges in forecasting peaks and valleys in database performance.

DynamoDB Employee Data Model Demo

- Designed to efficiently store and retrieve employee details
- Attributes:
  - LoginUsername
  - FirstNameand LastName
  - ManagerUsername
  - Skills
- Key Components:
  - Main table (Employee)
  - Global secondary index (Name)
  - Global secondary index (DirectReports)
- AccessPatterns:
  - Retrieval by login username
  - Search by name
  - Direct reports

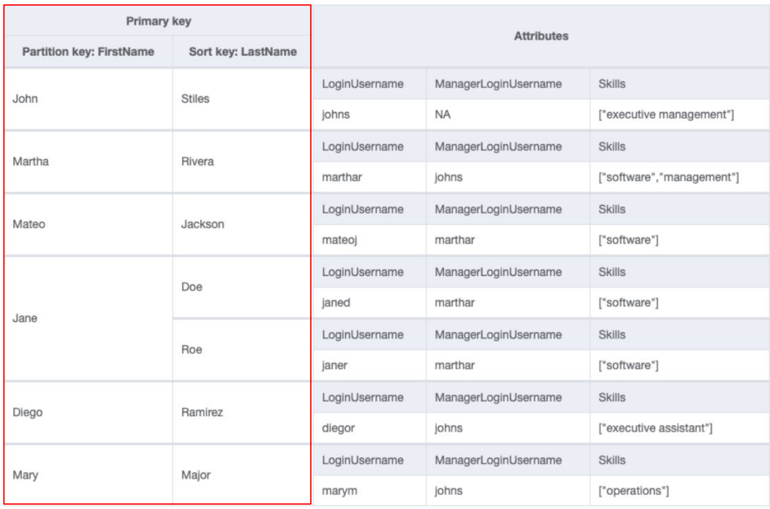
Employee Table

Primary keys:
- Simple primary key (Partition key)
- Composite primary key (Partition key and sort key)



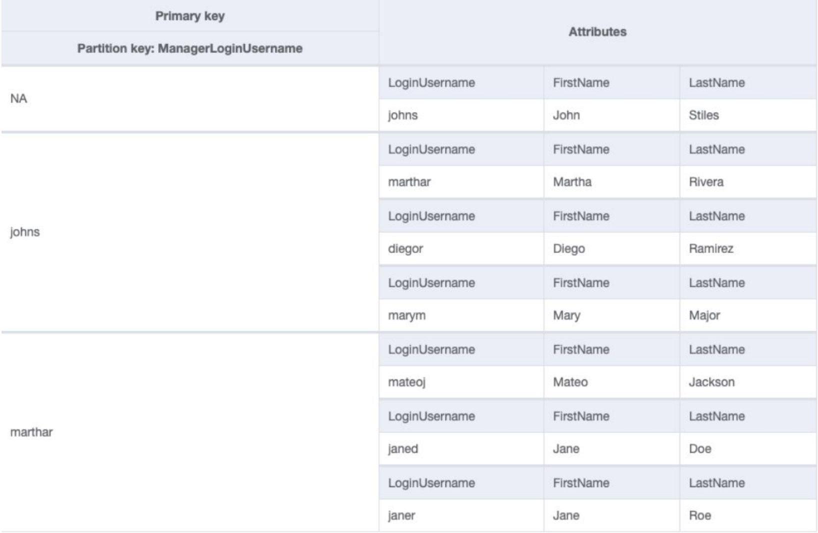
Primary key	Attributes			
Partition key: LoginAlias				
johns	FirstName	LastName	ManagerLoginAlias	Skills
	John	Stiles	NA	["executive management"]
marthar	FirstName	LastName	ManagerLoginAlias	Skills
	Martha	Rivera	johns	["software","management"]
mateoj	FirstName	LastName	ManagerLoginAlias	Skills
	Mateo	Jackson	marthar	["software"]
janed	FirstName	LastName	ManagerLoginAlias	Skills
	Jane	Doe	marthar	["software"]
diegor	FirstName	LastName	ManagerLoginAlias	Skills
	Diego	Ramirez	johns	["executive assistant"]
marym	FirstName	LastName	ManagerLoginAlias	Skills
	Mary	Major	johns	["operations"]
janer	FirstName	LastName	ManagerLoginAlias	Skills
	Jane	Roe	marthar	["software"]

Global Secondary Index - Name



Primary key		Attributes		
Partition key: FirstName	Sort key: LastName			
John	Stiles	LoginUsername	ManagerLoginUsername	Skills
		johns	NA	["executive management"]
Martha	Rivera	LoginUsername	ManagerLoginUsername	Skills
		marthar	johns	["software","management"]
Mateo	Jackson	LoginUsername	ManagerLoginUsername	Skills
		mateoj	marthar	["software"]
Jane	Doe	LoginUsername	ManagerLoginUsername	Skills
		janed	marthar	["software"]
	Roe	LoginUsername	ManagerLoginUsername	Skills
		janer	marthar	["software"]
Diego	Ramirez	LoginUsername	ManagerLoginUsername	Skills
		diegor	johns	["executive assistant"]
Mary	Major	LoginUsername	ManagerLoginUsername	Skills
		marym	johns	["operations"]

Global Secondary Index - DirectReports



Primary key	Attributes		
Partition key: ManagerLoginUsername			
NA	LoginUsername	FirstName	LastName
	johns	John	Stiles
johns	LoginUsername	FirstName	LastName
	marthar	Martha	Rivera
	LoginUsername	FirstName	LastName
	diegor	Diego	Ramirez
	LoginUsername	FirstName	LastName
	marym	Mary	Major
marthar	LoginUsername	FirstName	LastName
	mateoj	Mateo	Jackson
	LoginUsername	FirstName	LastName
	janed	Jane	Doe
	LoginUsername	FirstName	LastName
	janer	Jane	Roe

References

- https://ionutignatescu.medium.com/s3-vs-dynamodb-price-comparison-6e4edba7c40e
- https://aws.amazon.com/dynamodb/pricing/
- https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.html
- https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/workbench.SampleModels.html

Questions

Thank you!