IDS 521- Term Paper
DynamoDB and its ecosystem

Jigyasa Sachdeva
jsachd3@uic.edu
Student, UIC, MIS

Varun Maheshwari
vmahes3@uic.edu
Student, UIC, MIS

Shruti Jain
sjain95@uic.edu
Student, UIC, MIS

Sanjana Kalani
skalan7@uic.edu
Student, UIC, MIS

Abstract- Relational databases are not scalable, not suitable for cloud environments and cannot handle big data. (Mohammed, IM. (05/01/2014) Relational Vs. NoSQL databases: A survey [1]. To overcome these challenges, NoSQL databases were introduced and have gained popularity. To handle the velocity, veracity and volume of data, DynamoDB was introduced which is a key-value NoSQL database. This database is licensed by Amazon's AWS. It is a serverless database, it scales vertically and automatically and can process more than 10 trillion requests per day and peaks of more than 20 million requests per second. It is a pay-per request model which provides financial flexibility for clients. This paper provides a broad view of the comparison between NoSQL and Relational databases, types of NoSQL databases, DynamoDB and its overview, pricing structure of DynamoDB, companies using DynamoDB, an example of performing CRUD operations on the free tier model, and DynamoDB's comparison with other famous NoSQL databases.

Table of Contents

1. Relational Database:

RDBMS is a set of data elements with pre-existing relationships. These elements are represented in form of tables having rows and columns. Tables provides knowledge about the objects that are characterized in the database. Column describes the data and the field stores a real value of the object. The rows represents a set of connected values of an entity. Each row in a table has a unique identity known as a primary key, and rows can be related by foreign keys. This data can be made available in many ways without organizing the database tables itself **[9].**

1.1 Features of RDBMS:

- **SQL:** SQL (Structured Query Language) is used to facilitate communications with Relational Databases. The well-known ANSI SQL is supported through all famous relational database engines, and some of these engines additionally have an extension to ANSI SQL to guide capability which is specific to that engine. SQL is used to add, replace or delete rows of statistics, retrieving subsets of statistics for transaction processing and analytics programs, and to manage all factors of the database.
- **Data Integrity:** Data integrity represents particular, accurate and constant records. Relational databases makes use of many constraints to perform facts integrity in the database. These are primary Keys, Foreign Keys, 'Not NULL', 'Unique', 'Default' and 'Check' constraints. These constraints help provide enterprise policies on statistics to ensure the precision and safety of the information. Moreover, many relation databases permit custom code; embedded in triggers that run based totally on an action on the database.
- **Transactions:** Transactions are SQL statements that execute in a chain to form a unique unit of operation. Transactions offer an "all-or-nothing" proposition, meaning that the complete transaction ought to entire as a single unit and be written to the database or not one of the components of the transaction must go through. In the relational database terminology, a transaction consequence in a COMMIT or a ROLLBACK. Each transaction is dealt with in a coherent and reliable way independent of different transactions.
- **ACID Compliance:** The database transaction should be ACID compliant this is Atomic, Consistent, Isolated and Durable to enforce facts integrity.

Atomicity: It states that both transactions be correctly accomplished absolutely or if any part of the transaction fails, then the entire transaction be invalidated.
Consistency: It mandates the information that is written to the database as part of the transaction must adhere to all described regulations, and restrictions consisting of constraints, cascades, and triggers.
Isolation: It is critical to reaching concurrency manage and makes certain every transaction is independent unto itself.
Durability: It calls for that all the modifications made to the database be everlasting once a transaction is efficiently finished.

1.2. Advantages and Disadvantages of RDBMS

Advantages of RDBMS
- Structure of data provides ease of understanding
- Network Accessibility
- SQL support
- Speed and Maintenance
- Multiple User Access

Disadvantages of RDBMS:
- Cost
- Complex information is not supported
- Limit on the Field Length
- Data sharing is difficult between large systems
- Managing large volumes of data

2. NoSQL Database and its Emergence

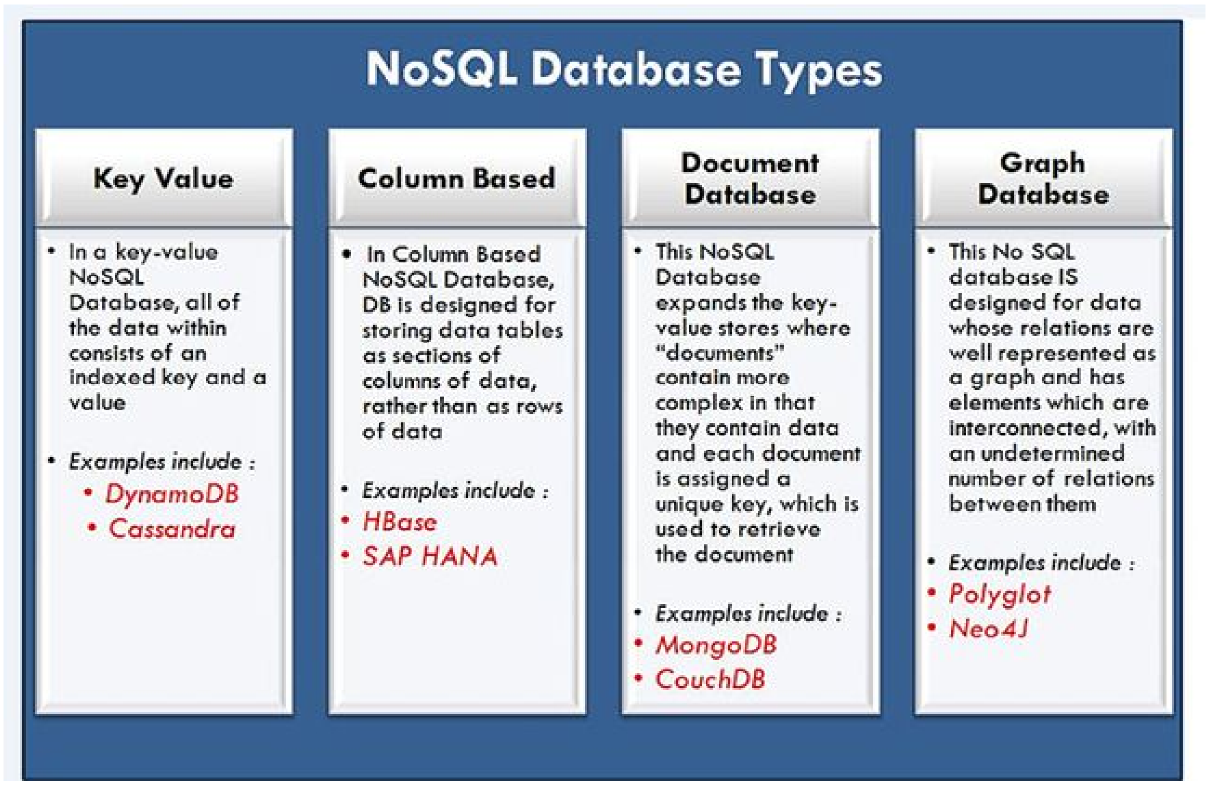
2.1. History:

The term NoSQL become used by Carlo Strozzi in 1998 to call his light-weight Strozzi NoSQL open-supply relational database that did no longer divulge the same old Structured Query Language (SQL) interface but become nonetheless relational. His NoSQL RDBMS is awesome from the circa-2009 fashionable idea of NoSQL databases. Strozzi shows that, because the present-day NoSQL movement "departs from the relational version altogether, it should, consequently, were referred to as more appropriately 'NoREL', relating to 'No Relational'.
Johan Oskarsson, then a developer at Last. Fm, reintroduced the term NoSQL in early 2009 while he organized an occasion to talk about "open source distributed, non relational databases". The call attempted to label the emergence of increasingly non-relational, allotted records stores, such as open supply clones of Google's Bigtable/MapReduce and Amazon's DynamoDB. **[10]**

2.2. Definition

NoSQL is a class of database management systems (DBMS) that does not follow all the hints of a relational DBMS and cannot use traditional SQL to query records.The word is especially deceptive while interpreted as "No SQL," and most translates it as "Not Only SQL," as this kind of database isn't always commonly a replacement however, as an alternative, complementary addition to RDBMSs and SQL. NoSQL-based systems are normally applied in very massive databases, which may be especially prone to performance issues due to the limitations of SQL and the relational version of databases. Many think of NoSQL due to the fact the modern-day database of choice that scales with Web necessities. Some excellent implementations of NoSQL are Facebook's Cassandra database, Google's Bigtable and Amazon's SimpleDB and DynamoDB. **[11]**

2.3 Types of NOSQL databases


NoSQL Database Types
Key Value
In a key-value NoSQL Database, all of the data within consists of an indexed key and a value
Examples include :
DynamoDB
Cassandra
Column Based
In Column Based NoSQL Database, DB is designed for storing data tables as sections of columns of data, rather than as rows of data
Examples include :
HBase
SAP HANA
Document Database
This NoSQL Database expands the key-value stores where "documents" contain more complex in that they contain data and each document is assigned a unique key, which is used to retrieve the document
Examples include :
MongoDB
CouchDB
Graph Database
This No SQL database IS designed for data whose relations are well represented as a graph and has elements which are interconnected, with an undetermined number of relations between them
Examples include :
Polyglot
Neo4J
[12]

2.3.1. Key Value:

- Key-value databases keep records as a hash desk of keys.**[2]** Every key map to an opaque binary item.
- Typical use cases are applications that require massive amounts of simple data (like web records/ sensor), caching etc.

2.3.2. Column Based:

- In column-based NoSQL dB; data is stored in columns of data and not as rows of data. Columns are grouped together to form column families.
- These families contain a virtually huge number of columns that can be created dynamically or at the definition of the schema. Read and write is implemented using columns.

2.3.3. Document Store NoSQL Databases

- The data which is a set of key value pair is compressed as document database which resembles the key-value store but the difference is reflected when the value stored provides structure and encoding of managed data.

2.3.4. Graph Databases

- In a Graph Based NoSQL Database, you won't find a rigid SQL format or a representation of tables and columns but use a flexible graphical representation that is perfect to address scalability issues
- Graph structures that provide index-free adjacency are used with edges, nodes and properties.

2.4. Advantages and Disadvantages of NoSQL Database:

Advantages:

- Flexible scalability
- Stores real time massive data
- Less expensive compared to RDBMS
- Requires less management with distribution of data and administrative requirements **[3]**
- Streaming Support
- Performance

Disadvantages:

- Many features are not yet implemented
- There is no strong customer support provided unlike RDBMS.
- The expertise in NoSQL databases is still not advanced
- Becomes difficult in selecting the right database for the application given the variety of databases available. **[13]**

3. DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service offered by Amazon Web Services. It requires a minimal amount of setup and maintenance on the part of a developer while offering great performance and scalability. It is a NoSQL key-value store and stores JSON objects in a simple key-value format.

3.1. Characteristics:

- Key-Value Store- Stores data as a collection of registers, each identified by a key [6]. A data retrieval hierarchy is used to obtain to index by unique keys [7].
- Highly Scalable- As the volume of data increases, NoSQL can manage it efficiently.
- Fully Managed- It is serverless, so users do not need to pay operational costs that come with server maintenances; everything is managed by Amazon
- Enterprise Ready- DynamoDB is built for mission-critical workloads, including support for broad set of applications that require complex business logic. DynamoDB helps secure your data with encryption

and continuously backs up your data for protection, with guaranteed reliability through a service level agreement. **[5]**

- ACID Transactions- DynamoDB provides native, server-side support for transactions, simplifying the developer experience of making coordinated, all-or-nothing changes to multiple items both within and across tables. With support for transactions, developers can extend the scale, performance, and enterprise benefits of DynamoDB to a broader set of mission-critical workloads.
- High Performance- On-demand backup and restore allows you to create full backups of your DynamoDB tables' data for data archiving, which can help you meet your corporate and governmental regulatory requirements. You can back up tables from a few megabytes to hundreds of terabytes of data and not affect performance or availability to your production applications.
- Security- DynamoDB encrypts all customer data at rest by default. Encryption at rest enhances the security of your data by using encryption keys stored in AWS Key Management Service. With encryption at rest, you can build security-sensitive applications that meet strict encryption compliance and regulatory requirements. The default encryption using AWS owned customer master keys is provided at no additional charge.
- Easy to use- Tutorials are provided by Amazon to help organisations and individuals create their own database.

3.2. Core components:

- Like other databases, DynamoDB stores its data in **tables**. Each table contains a set of items, and each item has a set of fields or **attributes**.
- Each table must have a **primary key**, present in all items within the table, and this primary key can be either a single attribute or a combination of two attributes: a **partition key** and a **sort key**.
- A *secondary index* lets you query the data in the table using an alternate key, in addition to queries against the primary key. DynamoDB doesn't require that you use indexes, but they give your applications more flexibility when querying your data. After you create a secondary index on a table, you can read data from the index in much the same way as you do from the table. [8]
- Stream: Each event is represented by a *stream record*. If you enable a stream on a table, DynamoDB Streams writes a stream record whenever one of the following events occurs
  - A new item is added to the table
  - An item is updated: The stream captures the "before" and "after" image of any attributes that were modified in the item.
  - An item is deleted from the table: The stream captures an image of the entire item before it was deleted.
- The data is **distributed** across many machines— this guarantees scalability and high performance
- To write and read items to and from a DynamoDB table, you'll need to use the DynamoDB HTTP API, either directly or by using the AWS SDK or the AWS CLI.
- Also, you can **batch** your reads and writes to DynamoDB tables, even across different tables at once.

DynamoDB supports transactions, automated backups, and cross-region replication.

3.3. Advantages of DynamoDB

- **Fully managed:** DynamoDB is a fully managed solution—you need perform no operational tasks at all to keep the database running. This means no servers to update, no kernel patches to roll out, no SSDs to replace. Using a fully managed solution reduces the amount of time your team spends on operations, allowing you to focus instead on developing your product
- **Autoscaling:** Nor must you do anything to scale the performance of the DynamoDB tables or their sizes as your application's load increases. DynamoDB takes care of all this automatically.
- **Automatic replication:** DynamoDB Global Tables allow you to have a multi-master, multi-region database with very little setup and no ongoing maintenance. This can help speed up your application's performance when you have customers across different regions of the world who use locally deployed instances of your application.
- **Streaming support:** DynamoDB allows you to create streams of updates to your data tables.You can then use these streams to trigger other work in other AWS services, including Lambda functions.

3.4. Disadvantages of DynamoDB:

- **Strong vendor lock-in:** DynamoDB is a proprietary solution and doesn't have an open-source version, so you'll be looking at a significant amount of work migrating to a different database solution. Some functionality, such as DynamoDB Streams, might be particularly hard to rebuild in a different database.
- **The cost structure can backfire with large datasets:** There are two pricing options for DynamoDB: the on-demand option and the provisioned throughput option. While the on-demand pricing is a good fit for applications with "spiky" usage and relatively low average traffic, as average usage increases the on-demand pricing structure can become quite expensive.
- **No built-in caching:** DynamoDB requires access to your data to be mostly uniform. If you access a certain segment of your data (like the most recent entries) much more than others, you won't have an out-of-the-box way to cache these recent items—except of course the DynamoDB Accelerator, which is proprietary and adds to your costs.
- **No support for JOIN operations:** DynamoDB design doesn't allow you to join data from across multiple tables. So if you need the data from multiple tables for a single operation, you'll find DynamoDB to be more expensive, slower and more complicated for implementing JOINs than with a relational datastore.

4. Companies using DynamoDB

Nike Case Study

Nike is a sports giant which migrated their cloud from Cassandra DB to DynamoDB over a period of years at a massive scale. DynamoDB enables them to achieve their goals which focuses on 'Innovation and Inspiration'. Their backend work like inventory, product data, checkout and other broad services were independently deployed as microservices and then independently handled by DynamoDB. Their launch team (which handles influx of traffic), product data team (which handles the requests to get the information on the product) and other teams like checkout migrated to DynamoDB. This helped them relieve cluster maintenance and achieve the scalability they need. ("Nike Case Study", n.d.)

Lyft Case Study

Lyft is a ridesharing app connecting drivers and passengers in 200 cities, arranging 14 million ride shares per month ("Lyft Case Study", n.d.). Lyft's challenge was the big users, a large number of concurrent users via web application, requesting ride shares from both drivers and passengers. The amount of requests fluctuates significantly depending on several factors that the company cannot possibly foresee beforehand. Auto-scaling feature of DynamoDB came in handy for Lyft; without having to predict the traffic, Lyft can scale flexibly during the peaks with auto-scaling feature that turns on automatically. Lyft utilizes multiple databases to store GPS information of all rides.

AirBnb Case study

Online travel marketplace supports hundreds of critical services on its platform, making it essential to maintain a reliable source control infrastructure.Airbnb has more than 1,000 engineers, who execute more than 100,000 continuous integration jobs on an average working day. GitHub Enterprise provides the engineers with a single source of truth for all code repositories. However, source control infrastructure had become an operational headache due to the system's scaling issues.The system did not scale with Airbnb's increasing Git traffic and hindered the team from focusing on higher-level problem solving and implementing new features.Airbnb turned to Amazon Web Services (AWS) to help achieve its goals.While exploring how to solve its challenges, Airbnb realized it could utilize Amazon Elastic File System (Amazon EFS), a simple, scalable system for Linux-based workloads for use with AWS Cloud services and on-premises resources.

5. Pricing Model

5.1. Key Terms

- **Read capacity unit (RCU):** Each API call to read data from your table is a read request. Read requests can be strongly consistent, eventually consistent, or transactional. For items up to 4 KB in size, one RCU can perform one *strongly consistent* read request per second. Items larger than 4 KB require additional RCUs. For items up to 4 KB in size, one RCU can perform two *eventually consistent* read requests per second. *Transactional* read requests require two RCUs to perform one read per second for items up to 4 KB

- **Write capacity unit (WCU):** Each API call to write data to your table is a write request. For items up to 1 KB in size, one WCU can perform one *standard* write request per second. Items larger than 1 KB require additional WCUs. *Transactional* write requests require two WCUs to perform one write per second for items up to 1 KB.

- **Replicated write capacity unit (rWCU):** When using DynamoDB global tables, your data is written automatically to multiple AWS Regions of your choice. Each write occurs in the local Region as well as the replicated Regions.

- **Streams read request unit:** Each GetRecords API call to DynamoDB Streams is a streams read request unit. Each stream read request unit can return up to 1 MB of data.

- **Backup and Restore**:DynamoDB offers two methods to back up table data. Continuous backups with point-in-time recovery (PITR) provide an ongoing backup of your table for the preceding 35 days. Customers can restore table to the state of any specified second in the preceding five weeks. On-demand backups create snapshots of the table to archive for extended periods to help meet corporate and governmental regulatory requirements.
- **Restoring a Table:**Restoring a table from on-demand backups or PITR is charged based on the total size of data restored (table data, local secondary indexes, and global secondary indexes) for each request.

- **Global Tables:**When customers select on-demand capacity mode for their DynamoDB global tables, they pay only for the resources your application uses on each replica table. Write requests for global tables are measured in replicated write request units instead of standard write request units. The number of write request units consumed for replication depends on the version of global tables customer is using. Read requests and data storage are billed consistently with standard tables (tables that are not global tables). If customers add a table replica to create or extend a global table in new Regions, DynamoDB charges for a table restore in the added regions per gigabytes of data restored

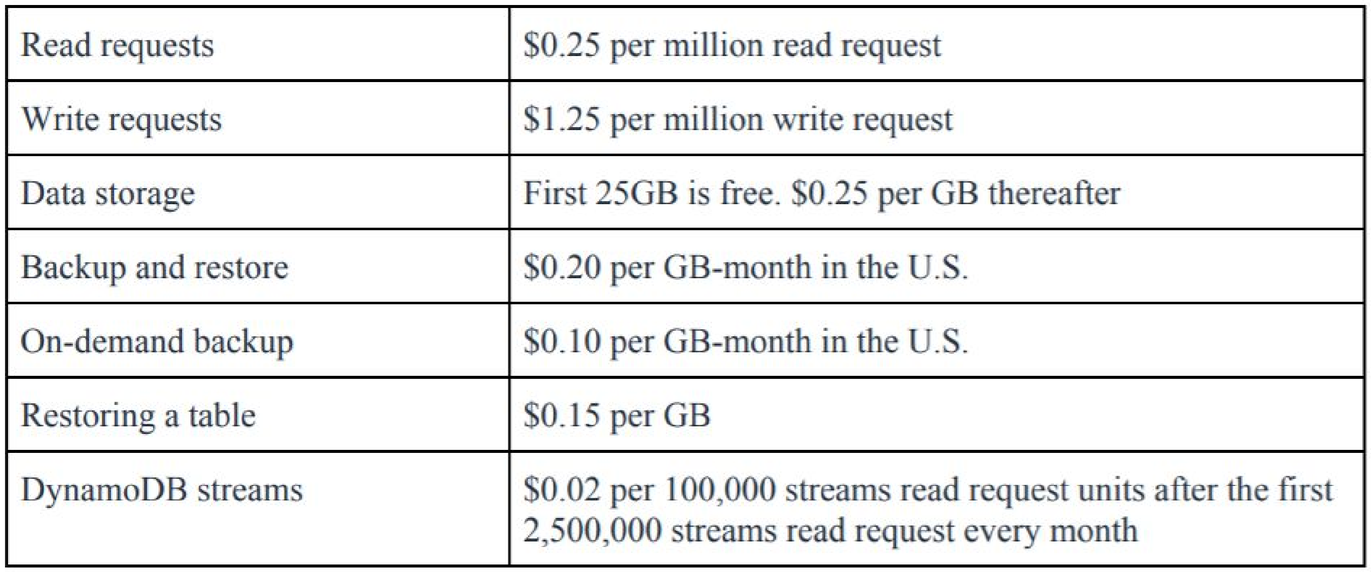
5.2. Pricing Model

5.2.1 Pricing for Dynamic Capacity Mode

With on-demand capacity mode, DynamoDB charges customers for the data reads and writes application performs on the tables. Customers do not need to specify how much read and write throughput they expect their application to perform because DynamoDB instantly accommodates workloads as they ramp up or down. **[4]**

On-demand capacity mode might be best if you:

- Create new tables with unknown workloads.
- Have unpredictable application traffic.
- Prefer the ease of paying for only what you use.



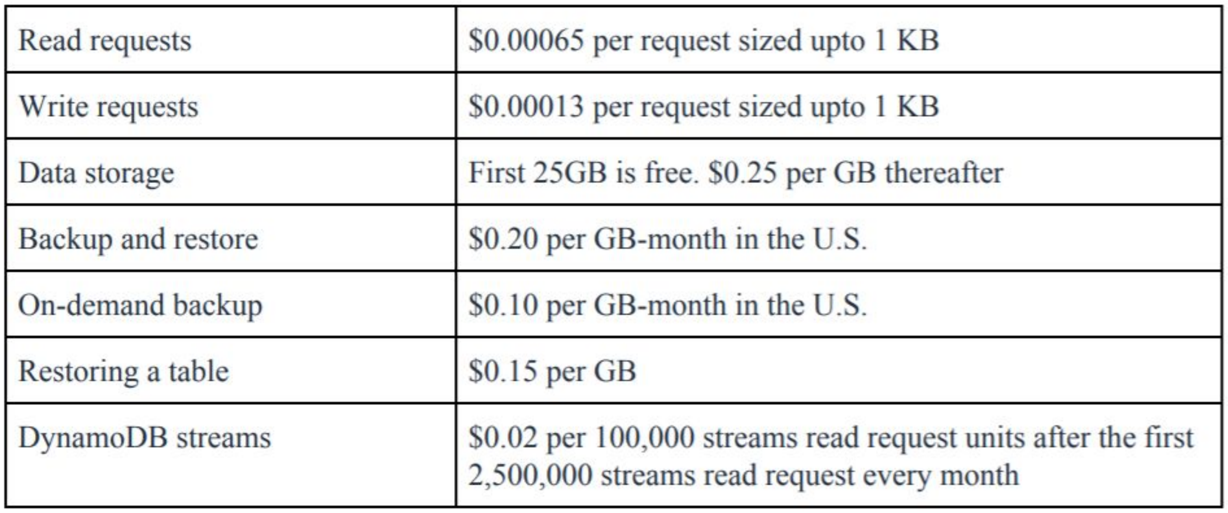
Read requests	$0.25 per million read request
Write requests	$1.25 per million write request
Data storage	First 25GB is free. $0.25 per GB thereafter
Backup and restore	$0.20 per GB-month in the U.S.
On-demand backup	$0.10 per GB-month in the U.S.
Restoring a table	$0.15 per GB
DynamoDB streams	$0.02 per 100,000 streams read request units after the first 2,500,000 streams read request every month

5.2.2. Pricing for Provisioned Capacity

With provisioned capacity mode, customers specify the number of reads and writes per second that they expect their application will require. They can use auto scaling to automatically adjust table's capacity based on the specified utilization rate to ensure application performance while reducing costs.

Provisioned capacity mode might be best if customers have the following preferences:

- Have predictable application traffic.
- Run applications whose traffic is consistent or ramps gradually.
- Can forecast capacity requirements to control costs.



Read requests	$0.00065 per request sized upto 1 KB
Write requests	$0.00013 per request sized upto 1 KB
Data storage	First 25GB is free. $0.25 per GB thereafter
Backup and restore	$0.20 per GB-month in the U.S.
On-demand backup	$0.10 per GB-month in the U.S.
Restoring a table	$0.15 per GB
DynamoDB streams	$0.02 per 100,000 streams read request units after the first 2,500,000 streams read request every month

5.3. DynamoDB free tier

The AWS Free Tier enables you to gain free, hands-on experience with AWS services. The following DynamoDB benefits are included as part of the AWS Free Tier. Each benefit is calculated monthly on a per-region, per-payer account basis.

- 25 Write Capacity Units and 25 Read Capacity Units of provisioned capacity (1 capacity unit = upto 1kb)
- 25 GB of data storage
- 25 replicated Write Capacity Units for global tables deployed in two AWS Regions (data in local storage replicated globally)
- 2.5 million stream read requests from DynamoDB Streams
- 1 GB of data transfer out (15 GB for your first 12 months), aggregated across AWS services

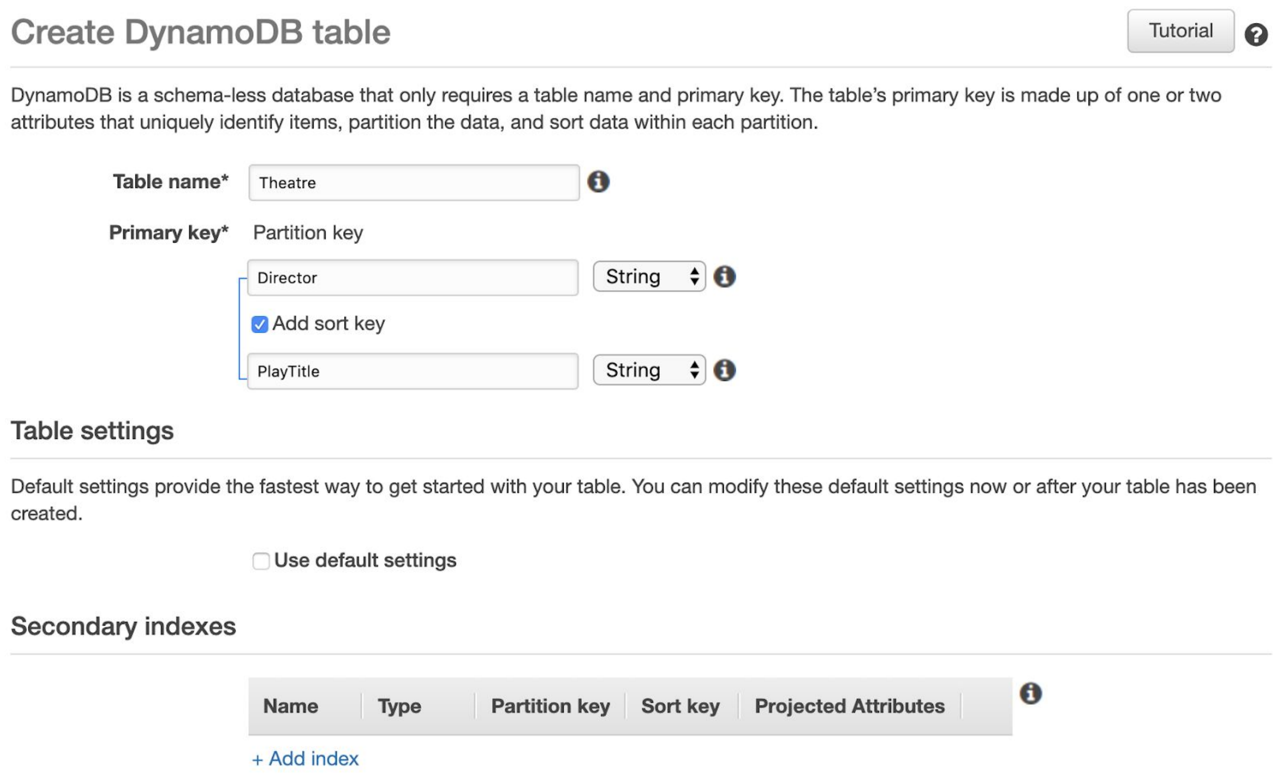
6. DynamoDB CRUD Implementation

6.1. Create

The first step for creating a table in DynamoDB is to give the table a name. In our example: we titled the table as 'Theatre'.

The second step is to determine the primary key of the table. The primary key in DynamoDB is either a single attribute (partition key) or a composite key (a combination of partition key and sort key). A partition key is the primary lookup to find a set of rows and DynamoDB uses the partition key's value as input to an internal hash function. The output from the hash function determines the physical storage internal to DynamoDB in which the item will be stored. [8] All items with the same partition key value are stored together, in sorted order by sort key value. In our example: 'Director' is our partition key as one director would have directed multiple plays and sort key is 'PlayTitle' which will help us reach the exact play we are searching for.

Other than primary key, DynamoDB helps helps query the data in the table using an alternate key called the secondary index. in addition to queries against the primary key. DynamoDB doesn't require that you use indexes, but they give your applications more flexibility when querying your data.



Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name*: Theatre

Primary key*: Partition key

Director — String

[✓] Add sort key

PlayTitle — String

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

[ ] Use default settings

Secondary indexes

Name	Type	Partition key	Sort key	Projected Attributes

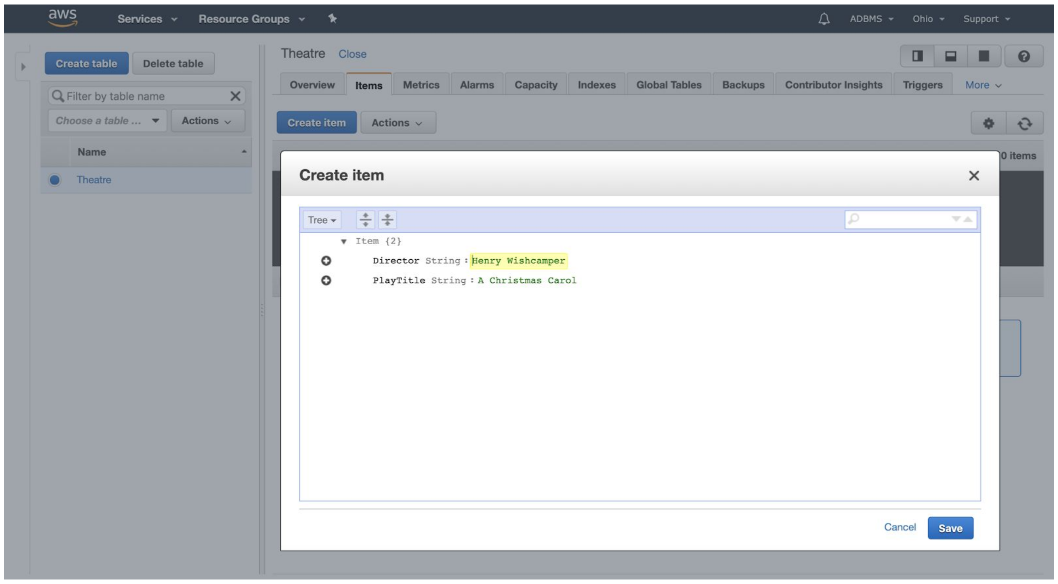
+ Add index

The table setting that we used were set to default which included:

- Provisioned read/write capacity mode. This could be on-demand with respect to pricing.
- The provisioned capacity included 5 read and write capacity units each per month.

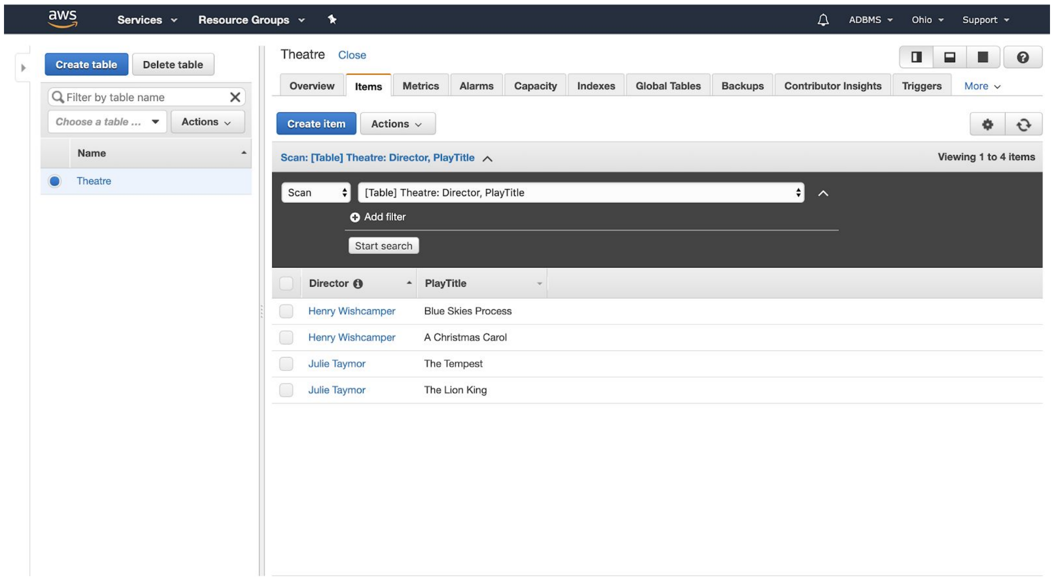
- The encryption was default where the key is owned by Amazon. It could also be KMS: where either the customer or AWS manages the Customer Master Key.

After the table structure is created; the next step is to add values in the table. As our table requires a play name directed by a director: we considered adding the following items: (Henry Wishcamper, Blue Skies Process); (Henry Wishcamper, A Christmas Carol); (Julie Taymor, The Lion King), (Julie Taymor, The Tempest). To create these items: click on 'Items' and then 'Create Item'. A pop up appears with the column name and data type, the data type can be changed here and a value is added. On clicking the 'Save' button, one successful row is created.



```
▾ Item {2}
    Director String : Henry Wishcamper
    PlayTitle String : A Christmas Carol
```

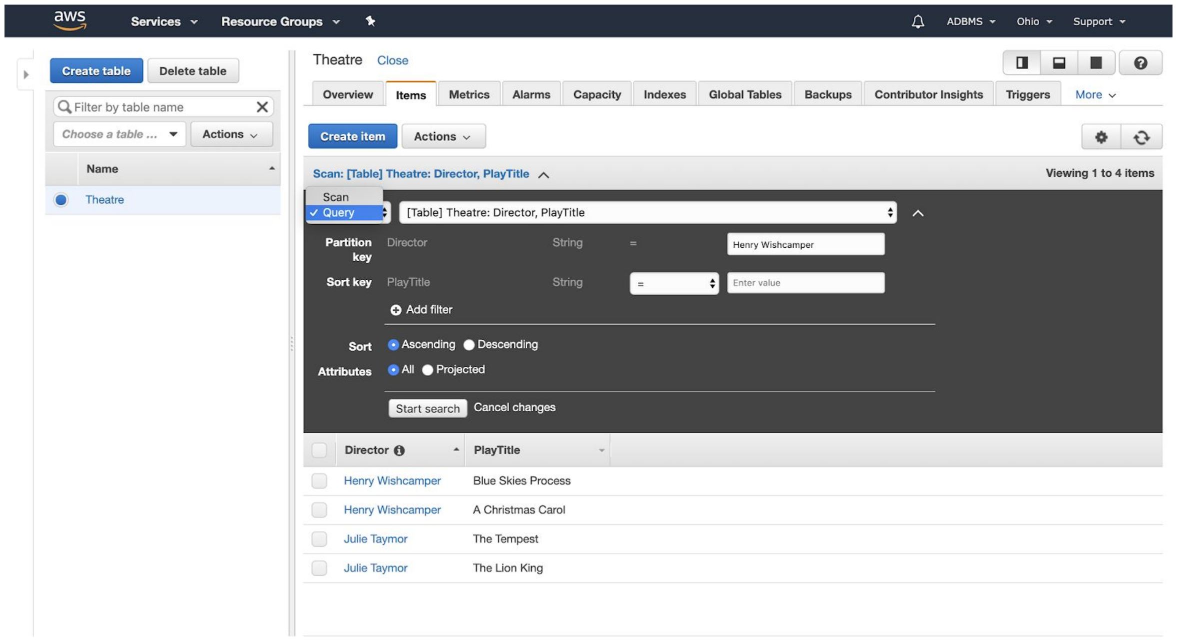
The final table looked as follows:



Director	PlayTitle
Henry Wishcamper	Blue Skies Process
Henry Wishcamper	A Christmas Carol
Julie Taymor	The Tempest
Julie Taymor	The Lion King

6.2. Read

To read or query the table: the table should be in ‘Scan’ mode as shown. The query will have a search key which could be either (partition key) or (partition key, sort key). On selecting either and putting the desired value; either column can be compared. For example: Partition Key: Director = ‘Henry Wishcamper’ matches all items in the column Director and checks if the value returned is ‘Henry Wishcamper’.



Scan: [Table] Theatre: Director, PlayTitle — Viewing 1 to 4 items

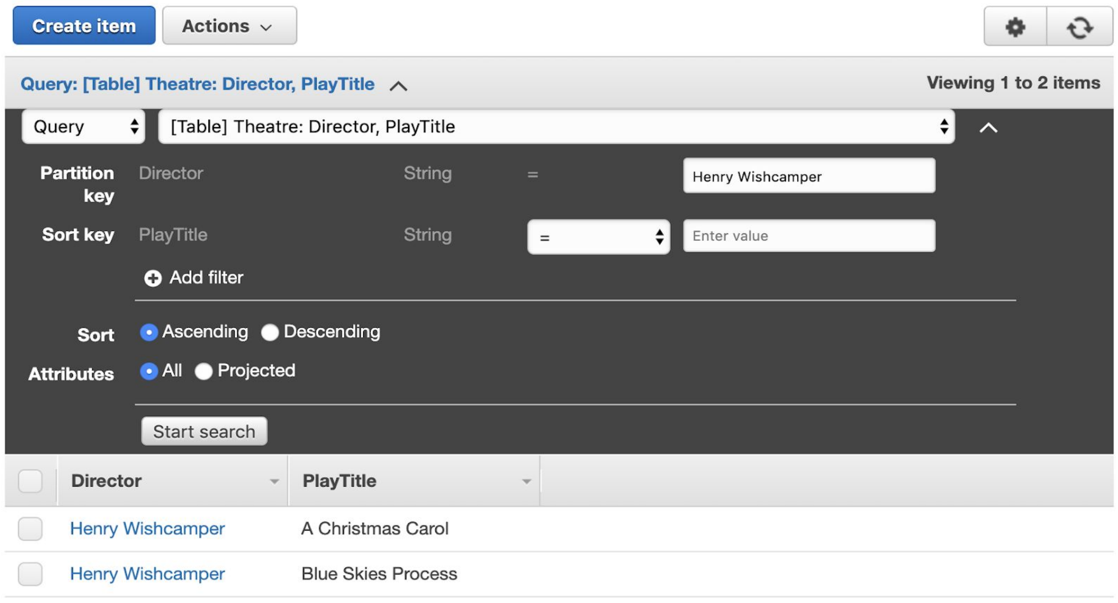
Query — [Table] Theatre: Director, PlayTitle

Partition key	Director	String	=	Henry Wishcamper
Sort key	PlayTitle	String	=	Enter value

Sort: (●) Ascending ( ) Descending
Attributes: (●) All ( ) Projected

Director	PlayTitle
Henry Wishcamper	Blue Skies Process
Henry Wishcamper	A Christmas Carol
Julie Taymor	The Tempest
Julie Taymor	The Lion King

The results to the query are as follows:



Query: [Table] Theatre: Director, PlayTitle — Viewing 1 to 2 items

Query — [Table] Theatre: Director, PlayTitle

Partition key	Director	String	=	Henry Wishcamper
Sort key	PlayTitle	String	=	Enter value

Sort: (●) Ascending ( ) Descending
Attributes: (●) All ( ) Projected

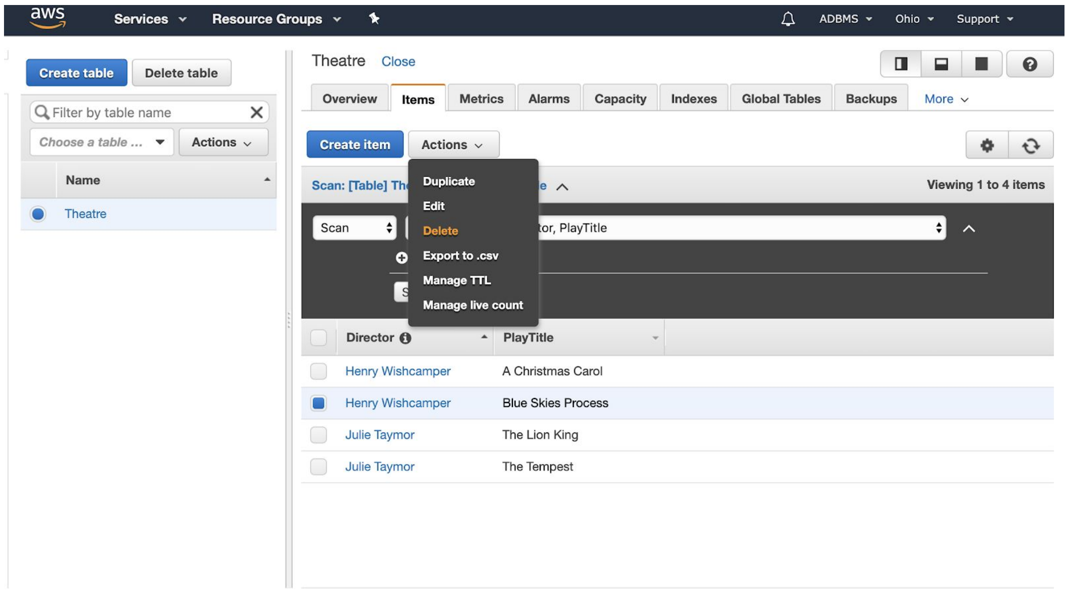
Director	PlayTitle
Henry Wishcamper	A Christmas Carol
Henry Wishcamper	Blue Skies Process

6.3. Update

The table can be updated by using a match case to find a record, and then on selecting: changing the record.

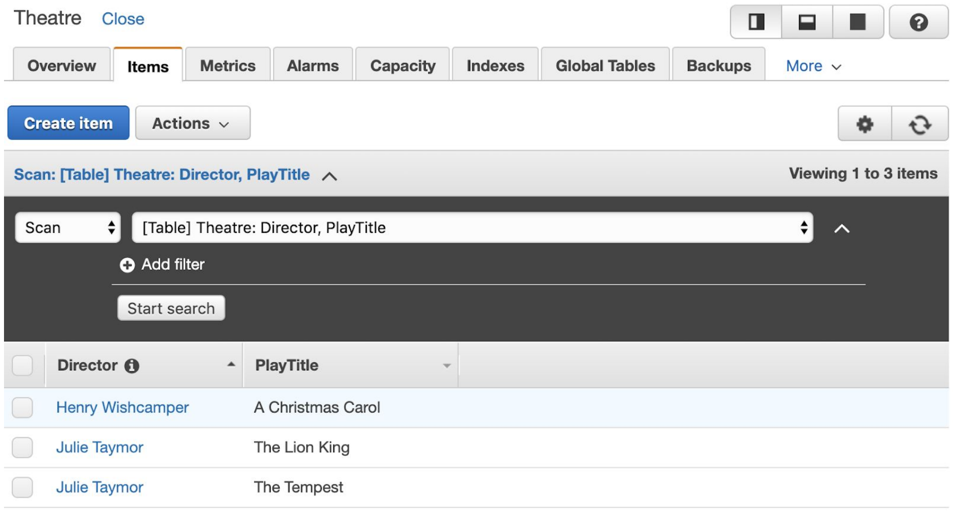
6.4. Delete

To delete a particular row, select the particular row by either searching it with a match case or manually finding it, then click on actions and find a button called ‘Delete’. On reaching this button, click ‘Delete’, say ‘Ok’ to the popup that shows up and the row gets deleted.



	Director	PlayTitle
☐	Henry Wischcamper	A Christmas Carol
☑	Henry Wischcamper	Blue Skies Process
☐	Julie Taymor	The Lion King
☐	Julie Taymor	The Tempest

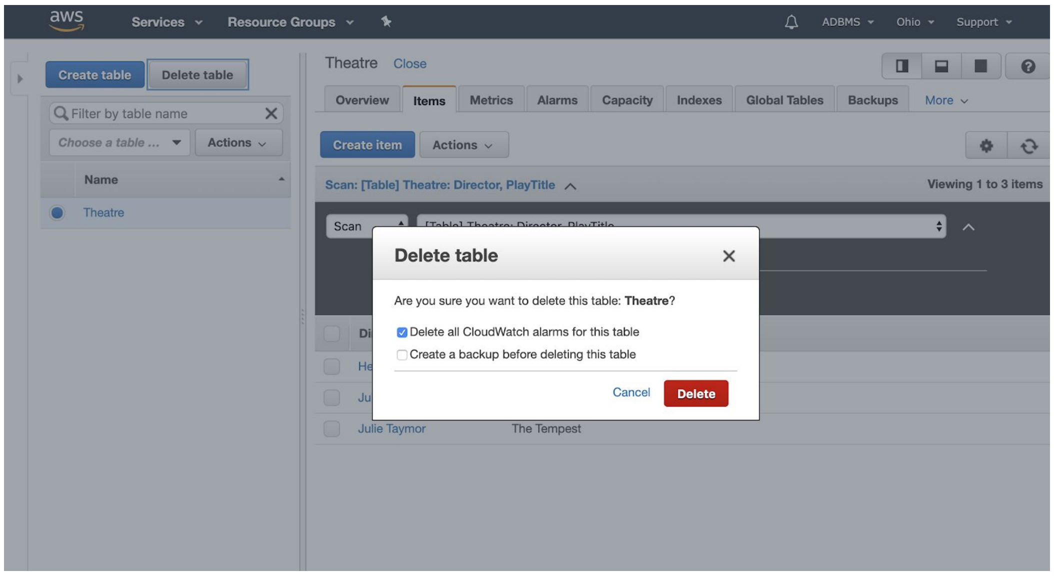
The row selected are deleted:



	Director	PlayTitle
☐	Henry Wischcamper	A Christmas Carol
☐	Julie Taymor	The Lion King
☐	Julie Taymor	The Tempest

Deleting a table:

Select the table from the left sidebar which contains the names of table. Click on Actions and select 'Delete'. The pop-up shows up asking either deleting all cloudwatch alarms for the table or creating a backup before deleting the table. Choose either and proceed to deleting the table.



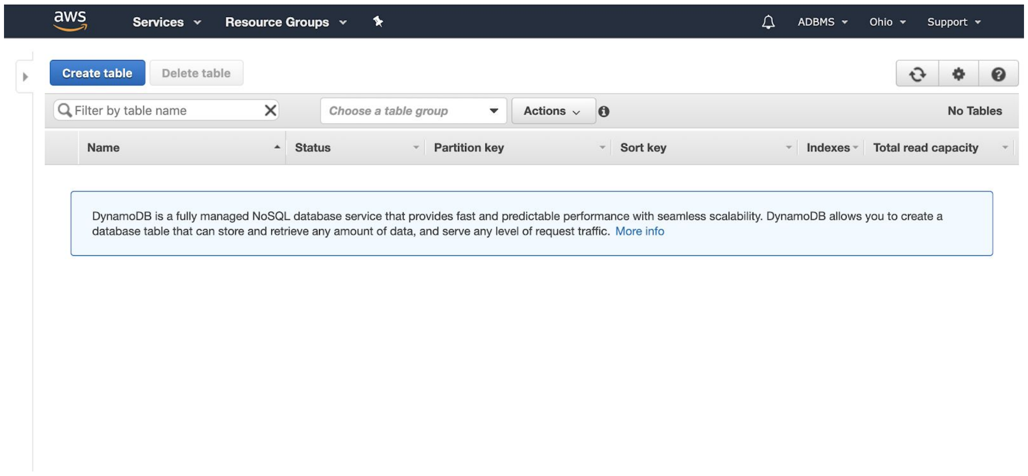
Delete table

Are you sure you want to delete this table: **Theatre**?

- [✓] Delete all CloudWatch alarms for this table
- [ ] Create a backup before deleting this table

Julie Taymor	The Tempest

The table gets deleted:



Name	Status	Partition key	Sort key	Indexes	Total read capacity

DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB allows you to create a database table that can store and retrieve any amount of data, and serve any level of request traffic. More info

7. Comparison with other databases

7.1. RDS VS DynamoDB:

This could be a good mental template to choose between RDS and DynamoDB: if we rely heavily on relational data structures, RDS might be a better fit. If each depicted dataobject is relatively self-contained, the best choice could be DynamoDB.**[14]**

RDS	DynamoDB
RDS is the relational database service of Amazon.	DynamoDB is a NoSQL key-value datastore and does not support traditional database system.
Amazon RDS performs routine server activities such as distribution, patching, backup, restoration, identification of errors, and repair.	Amazon DynamoDB is also a service that has been hosted and controlled. This means that users do not need to manage any databases, networking or tolerance to faults to preserve their decentralized existence.
RDS supports several database instances. It offers, among others, host versions of many common relational databases such as MySQL, PostgreSQL and Microsoft SQL Server.	DynamoDB provides single digit millisecond performance.
Storage size is 64TB (Aurora Engine),16TB (MySQL, PostgreSQL)	It supports tables of any size.
Number of tables per unit depends on the DB engine.	The number of tables per unit is 256.
It is optimized for OLTP database workload. Cannot perform better in case of dynamic scenarios.	The read and write throughput can be specified for each table. It has a DAX (DynamoDB Accelerator) capability to increase the performance.

Security is provided when we configure firewall settings and control network access to the database instances.	Security is provided by integrating with IAM.

7.2. Google Cloud Bigtable vs DynamoDB

Google Cloud Bigtable	**DynamoDB**
Google Cloud Bigtable is a NoSQL wide column datastore by Google.	DynamoDB is a key-value store NoSQL database provided by Amazon.
Bigtable stores an item's attributes in columns; group columns in column families. The columns can be extracted without having to collect whole objects.	DynamoDB stores an item as a single data blob. If primary key or a secondary key index is used to search an item in DynamoDB the whole item is obtained. Only the whole blob can be changed at a time and secondary index requires additional storage space.
Column values can be filtered in Bigtable without creating additional indices. Bigtable doesn't support secondary indices.	The DynamoDB consist of primary as well as secondary indices and using the index filtration of the data is done.
Bigtable stores the value in a single format.	DynamoDB supports predefined data types.
Bigtable does not provide trigger and does not support any such mechanisms.	DynamoDB contains lambda functions and offers complex automations on the data present in its table.

Due to its column structure, you can achieve much better performance in Bigtable compared to DynamoDB for certain use cases. However, Bigtable is a little perceptive to get started; DynamoDB provides more functionalities.**[14]**

7.3. Cassandra vs DynamoDB

Cassandra	**DynamoDB**
Cassandra is an alternative to DynamoDB and is an open-source NoSQL storage solution. It is a wide-column store database. Cassandra stores different fields in its own columns instead of storing key values as a single blob of data.	DynamoDB is a key-value store database and is not an open-source technology instead it is provided by Amazon. It uses JSON to store data.
Cassandra supports addition of schema in the database. It offers out of the box option for the schema controlling.	DynamoDB provides software defined schemas in which the schema checking would live in the application code.
Query language is modified version of SQL.	Query language is JSON-based.
Fine tuning of replication and cache settings can be done in Cassandra on demand.	AWS manages the replication and cache settings in case of DynamoDB.
Cassandra being open source is free, but the charges are for the machines that run it. E.g.-AWS EC2 and disk + networking costs.	DynamoDB is priced as per read/write operation and at provisioned read/write capacity unit.

As a rule, with smaller datasets, DynamoDB is cheaper and easier to use. Cassandra is a more customizable solution, so if you dedicate the right resources to manage and optimize it, it performs better on scale and is more cost-effective in the long term.

7.4. MongoDB vs DynamoDB

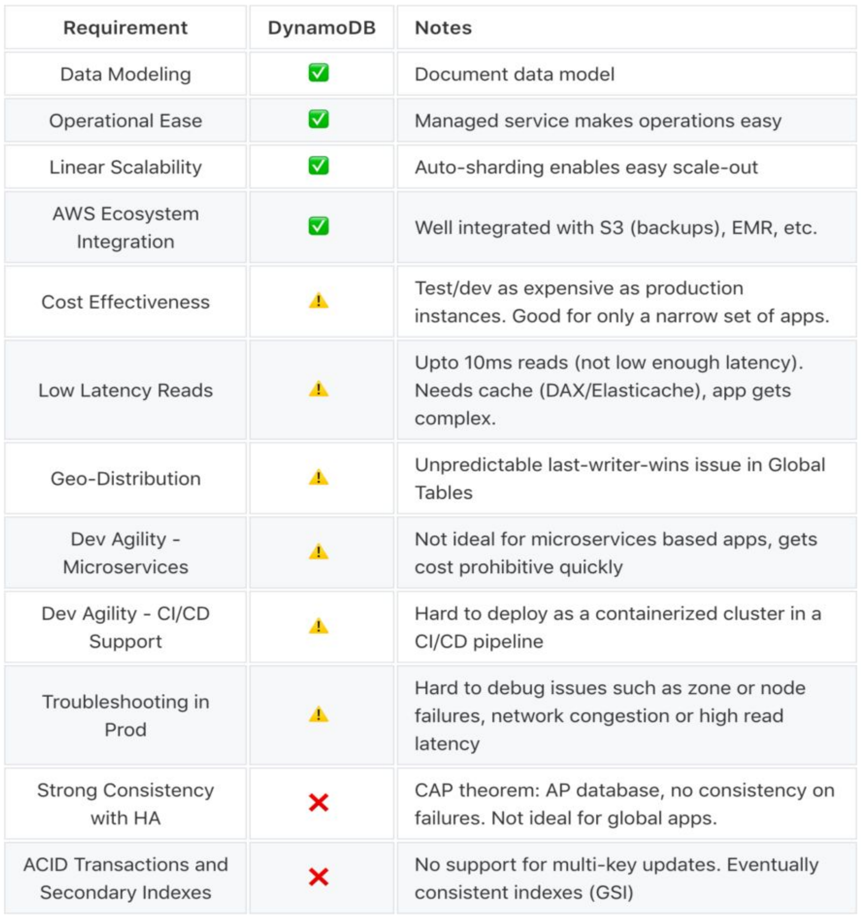
MongoDB	**DynamoDB**
MongoDB is an open source database that can run itself using hardware, cloud provider or MongoDB Atlas. MongoDB requires substantial resources for operation but offers performance tuning.	DynamoDB is proprietary of Amazon and is used in AWS. DynamoDB does not require substantial resources for operation. DynamoDB lacks performance tuning.
Pricing of MongoDB is at flat rate per hour depending on the specifications of the cluster.	DynamoDB follows pay as per use and provides option of scaling the provisioned capacity automatically.
MongoDB provides built-in data types like timestamps, floating-point numbers, and geospatial data.	DynamoDB provides less in built functions compared to MongoDB in terms of data processing.
MongoDB provides better granular controls over indexing, specifically when replicas and multiple data centers are present.	DynamoDB's settings are maintained and configured by AWS.
MongoDB is not tightly integrated with AWS services.	DynamoDB is integrated with other AWS services which makes it an easier option to build complex automation.

Overall, for most large applications, MongoDB generally scales better — but it has a higher entry barrier and can be expensive to operate if you host it on your own.

8. Conclusion

Amazon DynamoDB is key-value pair information repository that provides output at any level of single-digit millisecond. It is a fully managed, multi-regional, multimedia, robust server for internet-scale applications with built-in security, backup and restore, and in-memory caching.

Challenges and Future improvements DynamoDB lacks using multiple joins and single column fetching which could lead to an increase in its performance and accuracy. DynamoDB is not ACID compliant and to achieve the characteristics above the architecture makes the structure more complex to use.Thus improvements can be seen in inculcating ACID properties within the data model.There are no client controlled transactions,hence development can be done in this area to improve features and capabilities. Below diagram shows DynamoDB's strengths and weaknesses.**[15]**



Requirement	DynamoDB	Notes
Data Modeling	✅	Document data model
Operational Ease	✅	Managed service makes operations easy
Linear Scalability	✅	Auto-sharding enables easy scale-out
AWS Ecosystem Integration	✅	Well integrated with S3 (backups), EMR, etc.
Cost Effectiveness	⚠️	Test/dev as expensive as production instances. Good for only a narrow set of apps.
Low Latency Reads	⚠️	Upto 10ms reads (not low enough latency). Needs cache (DAX/Elasticache), app gets complex.
Geo-Distribution	⚠️	Unpredictable last-writer-wins issue in Global Tables
Dev Agility - Microservices	⚠️	Not ideal for microservices based apps, gets cost prohibitive quickly
Dev Agility - CI/CD Support	⚠️	Hard to deploy as a containerized cluster in a CI/CD pipeline
Troubleshooting in Prod	⚠️	Hard to debug issues such as zone or node failures, network congestion or high read latency
Strong Consistency with HA	❌	CAP theorem: AP database, no consistency on failures. Not ideal for global apps.
ACID Transactions and Secondary Indexes	❌	No support for multi-key updates. Eventually consistent indexes (GSI)

The paper provided detailed analysis on the pricing, competitors and use of DynamoDB. We can conclude by mentioning that DynamoDB can be used for self contained data structures where client does not prefer autotuning and has smaller datasets to work on. Also working on the above challenges can increase the performance and improve future developments and growth for AWS DynamoDB.

9. References:

[1] Mohammed, IM. (05/01/2014) Relational Vs. NoSQL databases: A survey, International Journal of Computer and Information Technology (ISSN: 2279 – 0764) Volume 03 – Issue 03, May 2014

[2] E. Anderson, X. Li, M. A. Shah, J. Tucek, and J. J. Wylie, "What consistency does your key-value store actually provide?" HotDep, vol. 10, pp. 1–16, 2010.

[3] Freire, S. M., Teodoro, D., Wei-Kleiner, F., Sundvall, E., Karlsson, D., & Lambrix, P. (2016). Comparing the performance of NoSQL approaches for managing archetype-based electronic health record data. PloS one, 11(3), e0150069

[4] Yoder, D., & Shriver, S. (February 28, 2019). Amazon DynamoDB auto scaling: Performance and cost optimization at any scale, AWS Blog. Retrieved from: https://aws.amazon.com/blogs/database/amazon-dynamodb-auto-scaling-performance-and-cost-optimization-at-any-scale/

[5] Zaki, A. K. (2014). NoSQL databases: new millennium database for big data, big users, cloud computing and its security challenges. International Journal of Research in Engineering and Technology (IJRET), 3(15), 403-409.

[6] Li, T., Zhou, X., Brandstatter, K., & Raicu, I. (2013). Distributed key-value store on hpc and cloud systems. In 2nd Greater Chicago Area System Research Workshop (GCASR) (Vol. 238)

[7] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny. Workload analysis of a large-scale key-value store.In Proceedings of the SIGMETRICS'12, June 2012.

[8]https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html

[9]https://aws.amazon.com/relational-database/

[10]https://en.wikipedia.org/wiki/NoSQL

[11]https://www.techopedia.com/definition/27689/nosql-database

[12]https://www.clariontech.com/blog/applications-that-work-best-with-nosql-database

[13]https://www.hadoop360.datasciencecentral.com/blog/advantages-and-disadvantages-of-nosql-databases-what-you-should-k

[14]https://serverless.com/dynamodb/

[15]https://blog.yugabyte.com/11-things-you-wish-you-knew-before-starting-with-dynamodb/