AWS DynamoDB

What is DynamoDB ?

- Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale.
- DynamoDB allows you to create a database table that can store and retrieve any amount of data and serve any level of request traffic.
- It is fully managed database and supports both document and key-value data models.
- Its flexible data model and reliable performance make it s great fit for mobile, web, gaming, ad-tech, IoT, and many other applications.

Basics of DynamoDB

- Stored on SSD storage
- Spread across 3 geographically distinct data centers
- Eventual Consistent Reads(Default)
- Strongly Consistent Reads

Dynamo DB Components

DynamoDB comprises of core basic components that include *Tables, Items* and *Attributes.* A Table is a collection of Items, each of which is a collection of one or more attributes. DynamoDB uses primary keys to uniquely identify each item in a table and secondary indexes to provide more querying flexibility.

- **Tables –** DynamoDB stores data in tables which are a collection of data; for example Students Table to store student information data
- **Items –** Each table contains multiple items, where an item is a group of attributes that uniquely identifies it when compared to other items. Items can be considered similar to rows or records in a relational database system
- **Attributes –** Each item is composed of one or more attributes. Attributes are similar to fields or columns. For example, an item in the Students table could be a student's record, that has attributes including StudentID, FirstName, LastName etc.

Dynamo DB Components

Important Note:

Whereas in a Relational Databases System, you must pre-define the table name, primary key, a list of columns, the data type for each column, etc., in DynamoDB, you only need to ensure that the table has a primary key.

Individual items in DynamoDB can have any number of attributes but you are limited to each item not exceeding 400KB

Read Consistency

When reading an item from DynamoDB, the response may not reflect the results of recently completed writes. This is because DynamoDB maintains multiple copies of the data across multiple availability zones and so it offers eventual consistent reads which are cheaper than strongly consistent reads.

- **Eventual Consistent Reads –** With Event Consistent Read operations, the response might not reflect the results of a recently completed write operation. The response might include some stale data. However, generally, the lag is no more than one second. If you repeat your read request after a short time, the response should return the latest data.
- **Strongly Consistent Reads –** You can request strongly consistent reads and DynamoDB will respond with the most up-to-date data, reflecting the updates from all prior write operations that were successful.

Data Types

Unlike traditional relational databases, where you need to specify the columns, their names as well as data types that will be contained in them,

DynamoDB only requires specifying a Primary Key field to start with.

You do not need to specify all attributes ahead of time of an item; you can add columns on the fly. This gives you the flexibility to expand the schema as required over time.

Data Types

There are three categories of data types:

- Scalar
- Set Data Types
- Document Data Types

Scalar Data Types

- Scalar – Represents one value and the following five scalar types are
  - String – up to a maximum of 400KB
  - Number – positive or negative up to 38 digits
  - Binary – up to 400KB in size
  - Null
  - Boolean

Set Data Types

- Set Data Types – These are unique lists of one or more Scalar Value. Each value is unique in a set and must be of the same data type. There is no guarantee of order:
  - String Set – Unique list of string attributes
  - Number Set – Unique list of number attributes
  - Binary Set – Unique list of Binary attributes

Document Data Types

- Document Data Types – used to represent multiple nested attributes and is like a JSON file in structure. Data types can be nested within each other up to 32 levels deep 35. You can have Lists and Map
  - List – used to store an ordered list of attributes of different data types
  - Map used to store unordered list of key/value pairs

Primary Key

- When you create a table, you need to specify a Primary key which uniquely identifies every item in a database. DynamoDB support two types of private keys:
- **Partition Key**
- **Partition Key and Sort Key (Composite key)**

Partition Key

- **Partition Key** – The primary key is defined with a **single attribute** and is known as the partition key. DynamoDB uses the partition key's value to build an unordered hash index which is used to identify the partition in the which the item will be stored.
- **Note** that if you are only using only a partition key, you cannot have two items on the same table using the same partition key

Partition Key and Sort Key

- **Partition Key and Sort Key –** This is known as a **composite primary** key and is made up of two attributes, namely the primary (partition) key and the sort (range) key.
- You can uniquely identify and item if you provide both the partition key and sort key. Note that you can have multiple items with the same partition key if they have different sort keys.

Important Note

- The partition key of an item is also known as its hash attribute. The term is because DynamoDB uses an internal hash function to evenly distribute data items across partitions, based on their partition key values.
- The sort key of an item is known as a range attribute. DynamoDB uses sort keys to stores items with the same partition key physically close together, in sorted order by the sort key value.

Secondary Indexes

- Amazon DynamoDB enables you to query the data in a table using an optionally defined alternative key know as a Secondary Index.
- **Local Secondary Index**
- **Global Secondary Index**

Local & Global Secondary Index

- **Local Secondary Index –** This is an index that has the same partition key as the table, but a different sort key. **These can only be created when the table is created.** Furthermore, you cannot modify or delete a Local Secondary Index once created.
- **Global Secondary Index –** this is an index with a partition key and sort key that can both be different from those on the table. **Global secondary indexes can be created or deleted on a table at any time**

Amazon DynamoDB Streams

- Applications can be designed to keep track of recent changes and they perform some action on the changed record sets.
- This method of streaming data is a feature available on DynamoDB and enables you to get a list of item changes for a **24-hour** period.
- The stream is essentially ordered flow of information about changes to items in an Amazon DynamoDB table. Once enabled, DynamoDB captures information about every modification to data items in the table.

Provisioning Capacity

- DynamoDB enables you to write and read from the tables you create a Database. You can create, update and delete individual items. In addition, you can use multiple querying options to search for data in your tables.
- To ensure high availability and low latency responses, you are required to specify you read and write throughput values when you create a table.
- DynamoDB uses this information to reserve sufficient hardware resources and appropriately partitions your data over multiple servers to meet your throughput requirements. When you create a table, you need to specify the following capacity units

Provisioning Capacity

- **Read Capacity Units**
  - Readers are rounded to increments of 4KB in size
  - In Eventual Consistent Reads, one read capacity unit is 2 reads per second for items up to 4KB
  - For Strongly Consistent Reads, one read capacity unit consist of 1 read per second of up to 4KB in size
- **Write Capacity Units**
  - Number of 1KB writes per second

- ***Remember*** *– One read capacity unit represents one strongly consistent read per second, or two eventually consistent reads per second, for items up to 4 KB in size*

- ***Remember*** *– One write capacity unit represents one write per second for items up to 1 KB in size.*

Pricing

- DynamoDB simply asks you to specify the target utilization rate and minimum to maximum capacity that you want for your table. DynamoDB handles the provisioning of resources to achieve your target utilization of read and write capacity, then auto scales your capacity based on usage.
- Optionally you can directly specify read and write capacity if you prefer manual.
  - Provisioned Throughput(Write) – One write capacity unit(WCU) provides up to one write per second, enough for 2.5 million writes per month. As low as $0.47 per WCU
  - Provisioned Throughput(Read) – One read capacity unit (RCU) provides up to two reads per second, enough for 5.2 million reads per month as low as $0.09 per RCU
  - Indexed Data Storage – DynamoDB charges an hourly rate per GB of disk space that your table consumes table throughput

Provisioning Capacity

Example
Read Capacity Requirements

- If you have a table and you want to read 100 items per second with strongly consistent reads and your items are 8KB in size, you would calculate the required provisioned capacity as follows;
- 8KB/4KB = 2 capacity units
- 2 read capacity units per item x 100 reads per second = 200 read capacity units
- **Note:** Eventual Consistent Reads would require 100 read capacity units

Write Capacity Requirements

- If you have a table and you want to write 50 items per second and your items are 4KB in size, you would calculate the required provisioned capacity as follows;
- 4KB/1KB = 4 capacity units
- 4 write capacity units per item x 50 writes per second = 200 write capacity units

Searching Items

You can use a Query or a Scan to search for items in a DynamoDB table.

Queries are primary search operations to help you search for items in a table or a secondary index using the primary key attribute values.

You need to provide partition key name and value to search for and you can also provide a sort key name and value and use a comparison operator to refine the search results.

Features

- Amazon DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache that can reduce Amazon DynamoDB response times from milliseconds to microseconds, even at millions of request per second.
- Document Data Model Support- DynamoDB supports storing, querying and updating documents
- Key-Value Data Model Support
- Seamless scaling
- High availability
- Develop locally, scale Globally
- Cross-region replication

Pricing

- A unit of write capacity enables you to perform one write per seconds for items of up to 1KB in size
- Similarly , a unit of Read Capacity enables you to perform one strongly consistent read per second(or two eventually consistent reads per second) of item up to 4KB in size.

You can do the following on Dynamodb

- Monitor recent alerts, total capacity, service health, and the latest DynamoDB news on the DynamoDB dashboard.
- Create, update, and delete tables. The capacity calculator provides estimates of how many capacity units to request based on the usage information you provide.
- Manage streams.
- View, add, update, and delete items that are stored in tables. Manage Time To Live (TTL) to define when items in a table expire so that they can be automatically deleted from the database.
- Query and scan a table.
- Set up and view alarms to monitor your table's capacity usage. View your table's top monitoring metrics on real-time graphs from CloudWatch.
- Modify a table's provisioned capacity.
- Create and delete global secondary indexes.
- Create triggers to connect DynamoDB streams to AWS Lambda functions.
- Apply tags to your resources to help organize and identify them.
- Purchase reserved capacity.

Table Overview

- **Overview** – View stream and table details, and manage streams and Time To Live (TTL).
- **Items** – Manage items and perform queries and scans.
- **Metrics** – Monitor CloudWatch metrics.
- **Alarms** – Manage CloudWatch alarms.
- **Capacity** – Modify a table's provisioned capacity.
- **Indexes** – Manage global secondary indexes.
- **Triggers** – Manage triggers to connect DynamoDB streams to Lambda functions.
- **Access control** – Set up fine-grained access control with web identity federation.
- **Tags** – Apply tags to your resources to help organize and identify them.