DynamoDB

Amar More

What is Amazon DynamoDB?

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability
- DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling
- DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data

What is Amazon DynamoDB?

- With DynamoDB, you can create database tables that can store and retrieve any amount of data and serve any level of request traffic
- You can scale up or scale down your tables' throughput capacity without downtime or performance degradation
- DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs
- You can create on-demand backups and enable point-in-time recovery for your Amazon DynamoDB tables. Point-in-time recovery helps protect your tables from accidental write or delete operations. With point-in-time recovery, you can restore a table to any point in time during the last 35 days

What is Amazon DynamoDB?

- DynamoDB allows you to delete expired items from tables automatically to help you reduce storage usage and the cost of storing data that is no longer relevant

High Availability and Durability in DynamoDB?

- DynamoDB automatically spreads the data and traffic for your tables over a sufficient number of servers to handle your throughput and storage requirements, while maintaining consistent and fast performance
- All of your data is stored on solid-state disks (SSDs) and is automatically replicated across multiple Availability Zones in an AWS Region, providing built-in high availability and data durability
- You can use global tables to keep DynamoDB tables in sync across AWS Regions

Core Components of DynamoDB

- **Tables** – Similar to other database systems, DynamoDB stores data in tables. A *table* is a collection of data
- **Items** – Each table contains zero or more items. An *item* is a group of attributes that is uniquely identifiable among all of the other items. In DynamoDB, there is no limit to the number of items you can store in a table
- **Attributes** – Each item is composed of one or more attributes. An *attribute* is a fundamental data element, something that does not need to be broken down any further

Core Components of DynamoDB

- Each item in the table has a unique identifier, or primary key, that distinguishes the item from all of the others in the table
- Other than the primary key, the table is schemaless, which means that neither the attributes nor their data types need to be defined beforehand. Each item can have its own distinct attributes
- Some of the items have a nested attribute (*Address*). DynamoDB supports nested attributes up to 32 levels deep

Example Table in DynamoDB

Music


```
{
   "Artist": "No One You Know",
   "SongTitle": "My Dog Spot",
   "AlbumTitle": "Hey Now",
   "Price": 1.98,
   "Genre": "Country",
   "CriticRating": 8.4
}
{
   "Artist": "No One You Know",
   "SongTitle": "Somewhere Down The Road",
   "AlbumTitle": "Somewhat Famous",
   "Genre": "Country",
   "CriticRating": 8.4,
   "Year": 1984
}
```



```
{
   "Artist": "The Acme Band",
   "SongTitle": "Still in Love",
   "AlbumTitle": "The Buck Starts Here",
   "Price": 2.47,
   "Genre": "Rock",
   "PromotionInfo": {
      "RadioStationsPlaying": {
         "KHCR",
         "KQBX",
         "WTNR",
         "WJJH"
      },
      "TourDates": {
         "Seattle": "20150622",
         "Cleveland": "20150630"
      },
      "Rotation": "Heavy"
   }
```

Primary Key

- When you create a table, in addition to the table name, you must specify the primary key of the table
- The primary key uniquely identifies each item in the table, so that no two items can have the same key
- DynamoDB supports two different kinds of primary keys:
- **Partition key** – A simple primary key, composed of one attribute known as the *partition key*. DynamoDB uses the partition key's value as input to an internal hash function. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored

Primary Key

- **Partition key and sort key** – Referred to as a *composite primary key*, this type of key is composed of two attributes. The first attribute is the *partition key*, and the second attribute is the *sort key*.
- DynamoDB uses the partition key value as input to an internal hash function. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored. All items with the same partition key value are stored together, in sorted order by sort key value.
- In a table that has a partition key and a sort key, it's possible for multiple items to have the same partition key value. However, those items must have different sort key values
- Each primary key attribute must be a scalar (meaning that it can hold only a single value). The only data types allowed for primary key attributes are string, number, or binary. There are no such restrictions for other, non-key attributes

Secondary Indexes

- A *secondary index* lets you query the data in the table using an alternate key, in addition to queries against the primary key
- You can create one or more secondary indexes on a table
- DynamoDB supports two kinds of indexes:
- **Global secondary index** – An index with a partition key and sort key that can be different from those on the table.
- **Local secondary index** – An index that has the same partition key as the table, but a different sort key.
- Each table in DynamoDB has a quota of 20 global secondary indexes (default quota) and 5 local secondary indexes
- DynamoDB maintains indexes automatically. When you add, update, or delete an item in the base table, DynamoDB adds, updates, or deletes the corresponding item in any indexes that belong to that table

RCU and WCU

- For provisioned mode tables, you specify throughput capacity in terms of read capacity units (RCUs) and write capacity units (WCUs):
- One *read capacity unit* represents one strongly consistent read per second, or two eventually consistent reads per second, for an item up to 4 KB in size
- Transactional read requests require two read capacity units to perform one read per second for items up to 4 KB
- If you need to read an item that is larger than 4 KB, DynamoDB must consume additional read capacity units

RCU and WCU

- One *write capacity unit* represents one write per second for an item up to 1 KB in size
- If you need to write an item that is larger than 1 KB, DynamoDB must consume additional write capacity units
- Transactional write requests require 2 write capacity units to perform one write per second for items up to 1 KB

Table Classes

- DynamoDB offers two table classes designed to help you optimize for cost
- The DynamoDB Standard table class is the default, and is recommended for the vast majority of workloads
- The DynamoDB Standard-Infrequent Access (DynamoDB Standard-IA) table class is optimized for tables where storage is the dominant cost
- For example, tables that store infrequently accessed data, such as application logs, old social media posts, e-commerce order history, and past gaming achievements, are good candidates for the Standard-IA table class

Table Classes

- Every DynamoDB table is associated with a table class (DynamoDB Standard by default)
- All secondary indexes associated with the table use the same table class
- Each table class offers different pricing for data storage as well as for read and write requests
- The choice of a table class is not permanent—you can change this setting using the AWS Management Console, AWS CLI, or AWS SDK