An Ultimate Beginner Guide to DynamoDB

The emergence of cloud business has changed the way we develop our applications. So do the responsibilities of developers.

Kisan Tamang Follow
Oct 21 · 6 min read

> *In this article, I've covered the important concepts you must know in order to get started with* DynamoDB. *You can use it as a cheatsheet.*

Image: AWS DyanamoDb

Introduction

Amazon DynamoDB is a fully managed NoSQL database service by AWS that provides fast and predictable performance with seamless scalability.

DynamoDb can handle any kind of OLTP transaction or real-time operations.

In DynamoDB you do not create any database server as you do it in traditional relational databases. It means you don't have to manage any database servers. You only create a table and start using it.

Core Components

The core components of DynamoDB include ***tables***, ***items***, and ***attributes***.

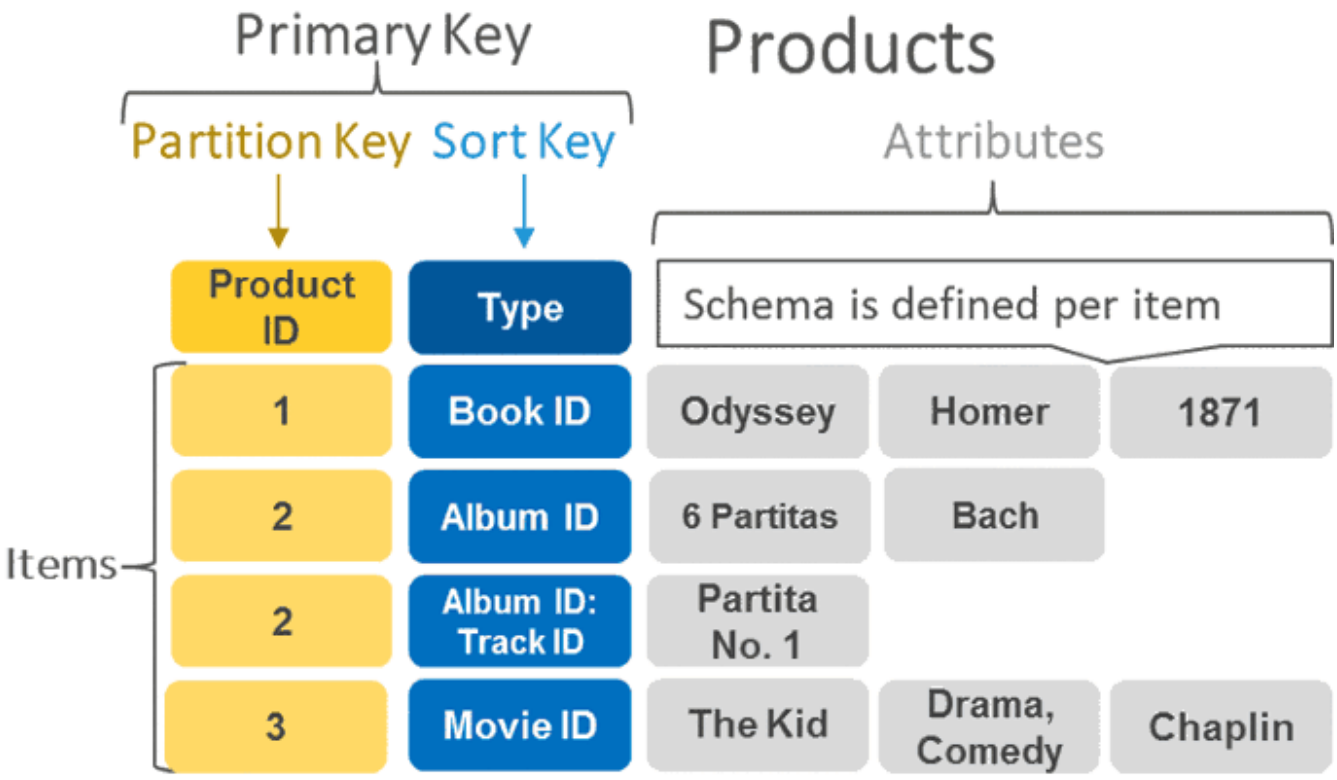


Image: DynamoDB Table structure

Table

Tables are a collection of *items* or data.

Items

An *item* is a group of attributes that is uniquely identifiable among all of the other items. Items in DynamoDB are similar in many ways to rows, records, or tuples in other database systems.

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.

Primary Key

While using DynamoDB, you specify the table name. In addition, you also specify the primary key for that table. The primary key uniquely identifies each *item* in DynamoDB.

How do you define Primary Key in DynamoDB?

1# Simple Primary Key

A simple primary key is composed of one attribute known as the ***partition key***.

Internally DynamoDB hash function takes that partition key as a value and initiates a hash function to generate a unique partition key which determines which partition data should be stored physically.

2# Composite Primary Key (Partition Key + Sort Key)

Another way of defining the primary key in DynamoDB is by using both the ***partition key*** and ***sort key,*** *also known as a* ***composite primary 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 two items to have the same partition key value. However, those two items must have different sorts of key values.

> Choosing the right key is always crucial while modeling your database. That is whole new topic for

next article.

Secondary Indexes

When the requirement of using a primary key limits the access patterns of a table, we use secondary indexes.

A *secondary index* is a data structure that contains a subset of attributes from a table, along with an *alternate key* to support `Query` operations.

You can retrieve data from the index using `Query` in much the same way as you use `Query` with a table. A table can have multiple secondary indexes, which give your applications access to many different query patterns.

Local Secondary Indexes (LSI)

Local secondary indexes can be used on a table with a composite primary key to specify an index with the same HASH key but a different RANGE key for a table.

Global Secondary Indexes (GSI)

Global Secondary Indexes allow you to query efficiently over any field (attribute) in your DynamoDB table. GSIs can treat any table attribute as a key, even attributes not present in all items.

Data Types

DynamoDB supports many data types for *items* in the table.

Scaler Types

Number, String, Boolean, Null.

Document Types

List, Map.

Set types

String set, Number set, Binary set.

Read Consistency

Eventually Consistent Reads

Eventual consistent reads data is returned immediately but data can be inconsistent. Copies of data will be generally consistent in 1 second.

Strongly Consistent Reads

Strongly Consistent Reads will always read from the leader partition since it always has an up-to-date copy. Data will never be inconsistent but latency may be higher. Copies of data will be consistent with a guarantee of 1 second.

Read/Write Capacity Mode

Depending on your application need, you can choose either of the following modes:

On-demand Mode

An *On-demand* mode is a good option if any of the following are true:

- You create new tables with unknown workloads.
- You have unpredictable application traffic.
- You prefer the ease of paying for only what you use.

Provisioned Mode

The *provisioned mode* is a good option if any of the following are true:

- You have predictable application traffic.
- You run applications whose traffic is consistent or ramps gradually.

- You can forecast capacity requirements to control costs.

Accessing DynamoDB

#1 Query

The `Query` operation in Amazon DynamoDB finds items based on primary key values.

```
aws dynamodb query --table-name Users --key-condition-expression
"UserName = :name" --expression-attribute-values '{":name":
{"S":"Customers"}}'
```

#2 Scan

A `Scan` operation in Amazon DynamoDB reads every item in a table or a secondary index. By default, a `Scan` operation returns all of the data attributes for every item in the table or index.

```
aws dynamodb scan --table-name Users --filter-expression "Users=
:name" --expression-attribute-values '{":name":{"S":"User A"}}'
```

#3 Using CLI

You can use AWS CLI to access dynamodb. For example, to create a table you can use:

```
aws dynamodb create-table --table-name Customers --attribute-
definitions AttributeName=id, AttributeType=S
AttributeName=CustomerName,AttributeType=S --key-schema
AttributeName=id,KeyType=HASH AttributeName=CustomerName
,KeyType=RANGE --provisioned-throughput
ReadCapacityUnits=1,WriteCapacityUnits=1
```

The above command will create a table named *Customers* with the provided configuration.

Some available commands:

👉 *list all the tables in dynamodb*

```
aws dynamodb list-tables
```

👉 writes a single item into the database.

```
aws dynamodb put-item
```

👉 reads the single item from the database

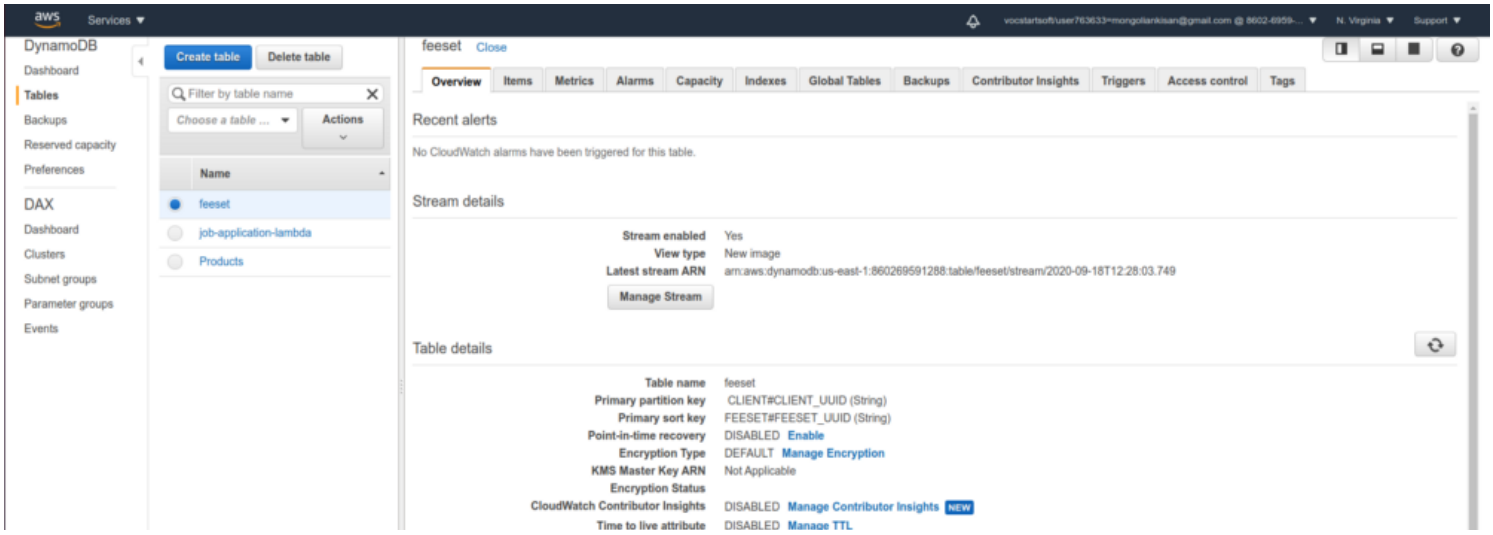
```
aws dynamodb get-item
```

Find all the APIs here.

> ***Note**: You need valid AWS access key and secret access key to work with AWS CLI.*

#4 Using Console

You can also the AWS management console for accessing dynamodb. But as a developer, working with a console is always a bad idea. The AWS dynamoDB console will look like this.



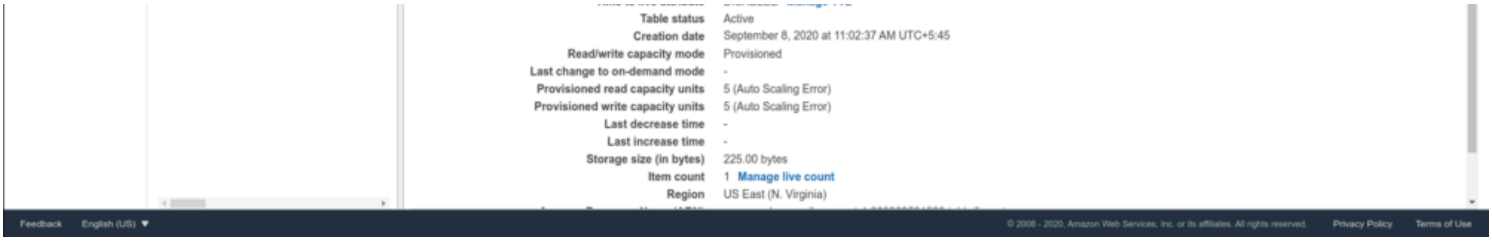


Image: AWS DynamoDB Console

#5 Using API

It is fine to work with CLI and Console. Actually, you can do much more with these tools. However, what if you want to use DynamoDB for your next application. In that case, AWS provides SDK for DynamoDB. For Node developer, visit here.

> *Note: To work with AWS SDK, you first need Access Key and Secret Access Key.*

#6 Using NoSQL workbench

You can also access your DynamoDB by using NoSQL Workbench provided by AWS. You can download it from here.

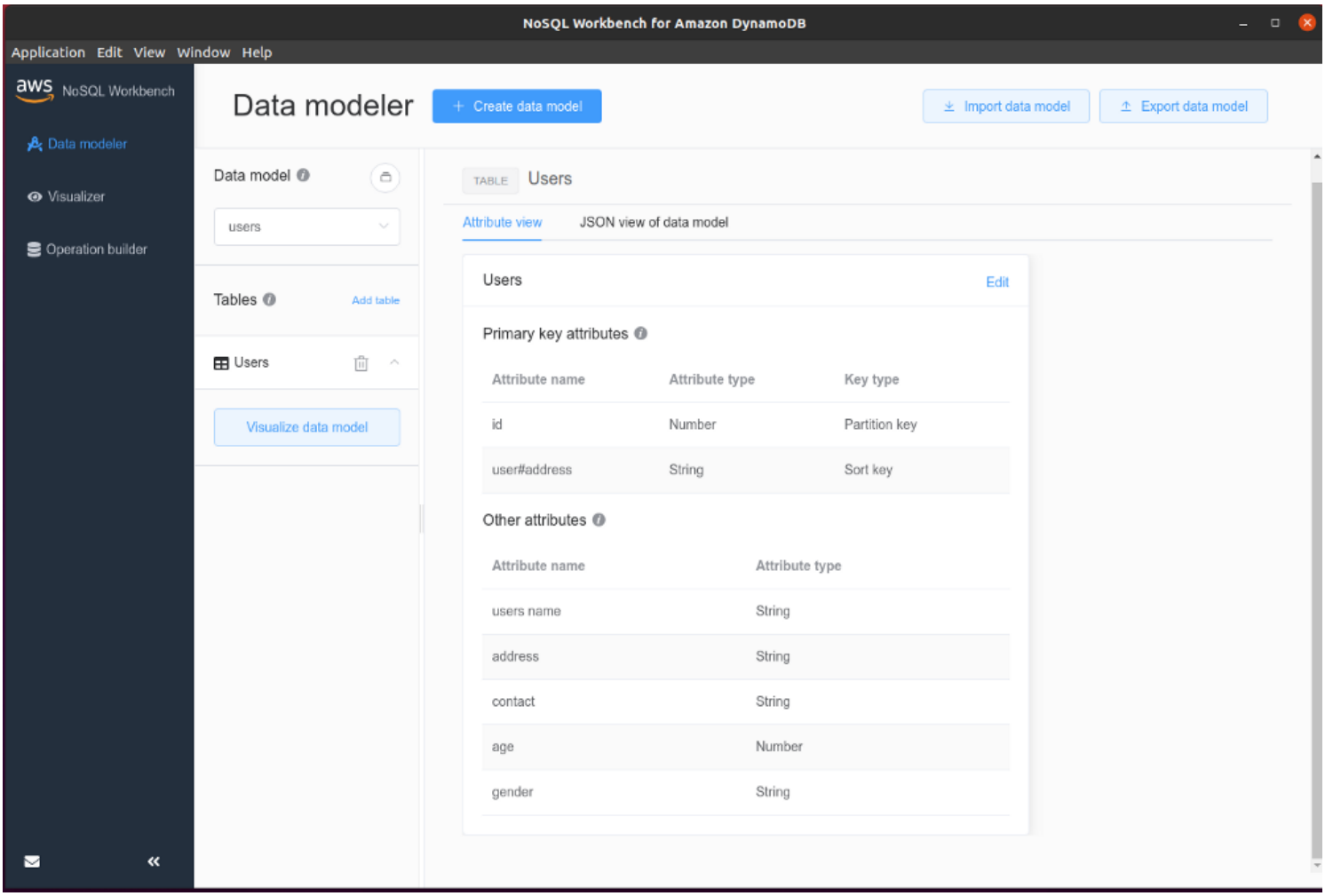


Image: AWS NoSQL workbench

DynamoDB Streams

Dynamodb stream features allow us to captures the changes made in the dynamodb table. Such, capture then can be used as an event for another operation.

For e.g, A new customer adds data to a DynamoDB table. This event invokes another application that sends a welcome email to the new customer.

DyanamoDB DAX

DynamoDB DAX is an in-memory caching mechanism for dynamodb for additional performance that enables you to benefit from fast in-memory performance for demanding applications.

DynamoDB Local

You can set up a DynamoDB a couple of ways in your local machine. You can find all the steps here. If you are a docker fan you can find a docker image here. AWS also provides NoSQL GUI workbench for visualizing your dynamodb. You can download it from here.

> *If you want to run dynamodb locally in your local machine, you can read my previous article here.*

Resources

1. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html
2. https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/Welcome.html
3. https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/DynamoDB.html
4. http://dynamodbguide.com/
5. https://www.dynamodbbook.com/

Wrapping Up

DynamoDB is an immensely cool technology when used in the right way. With the emergence of serverless technologies, DynamoDB is gaining so much popularity. It is hugely used in event-driven architectures. So, learning dynamodb might solve your next big problem in your next project.

Please follow me, I will keep posting about DynamoDB in the future, and let me know if you have any queries in the comment.

Keep Learning folks! 🙂

Serverless Dynamodb AWS Serverless Architecture 2020