Practical No. 5

Title: Implementation of DynamoDB queries

Aim: Design a Database Schema for an application using DynamoDB (DQL)..

Software required: DynamoDB software

Theory:

- Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that require consistent single-digit millisecond latency at any scale.
- It is a fully managed database that supports both document and key-value data models.
- Its flexible data model and performance makes it a great fit for mobile, web, gaming, ad-tech, IOT, and many other applications.
- It is stored in SSD storage.
- It is spread across three geographically data centres.
- DynamoDB allows users to create databases capable of storing and retrieving any amount of data, and serving any amount of traffic.
- It automatically distributes data and traffic over servers to dynamically manage each customer's requests, and also maintains fast performance.
- The DynamoDB Environment only consists of using your Amazon Web Services account to access the DynamoDB GUI console, however, you can also perform a local install.
- The AWS (Amazon Web Service) provides a version of DynamoDB for local installations.
- It supports creating applications without the web service or a connection.
- It also reduces provisioned throughput, data storage, and transfer fees by allowing a local database.

Working Environment -

You can use a JavaScript shell, a GUI console, and multiple languages to work with DynamoDB. The languages available include Ruby, Java, Python, C#, Erlang, PHP, and Perl.

Core Concepts of AWS DynamoDB

- Tables

    - In Amazon DynamoDB, the collection of items is known as a table. A table in AWS DynamoDB is not a structured table with a fixed number of cells or columns.
- Items
    - Each table in Amazon DynamoDB contains one or more items. Items are made up of a group of attributes that are uniquely identifiable.
- Attributes
    - Attributes in AWS DynamoDB are fundamental data elements or values that reside in an item. Equivalent to data values that reside in a particular cell of a table in a relational database.

Amazon DynamoDB Features

- Using DynamoDB, developers can easily develop scalable cloud-based applications
- AWS can easily achieve data retrieval in single-digit milliseconds
- DevOps need not worry about managing the high availability and durability of data because DynamoDB automatically replicates it synchronously across multiple AWS Availability Zones (AZs)
- DynamoDB can be provisioned according to the number of write units and several read units allocated
- The user's database table always remain available based on provisioned throughput requirements like read-write units per second
- DynamoDB utilizes JSON as a transport protocol
- Hashkeys are used for the data partitioning in DynamoDB
- NoSQL and Big Data are the technologies that work together because they both share the same allocated and scalable side-to-side structure of the database

The classification of DynamoDB is as follows: -

- **Control Plane** (It is responsible for creating and managing DynamoDB table)
    - Create Table
    - Describe Table
    - List Table
    - Delete Table

- **Data Plane** (It consists of 'CRUD' operation, i.e. Create, Read, Update & Delete)
    - Creating Data
        - PutItem
        - BatchWriteItem
    - Reading Data
        - GetItem
        - BatchGetItem
        - Query
        - Scan

- Updating Data
  - UpdateItem
- Deleting Data
  - DeleteItem
  - BatchWriteItem

- **DynamoDB Stream**
  - ListStream
  - DescribeStream
  - GetSharedIterator
  - GetRecords

Amazon DynamoDB Primary Key -

When you create a table, in addition to the table name, you have to 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 that is 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. An important rule to implementing a Partition key is that in A table that has only a partition key, no two items can have the same partition key value. The People table described in Tables, Items, and Attributes is an example of a table with a simple primary key (PersonID). You can access any item in the People table directly by providing the Person Id value for that item.

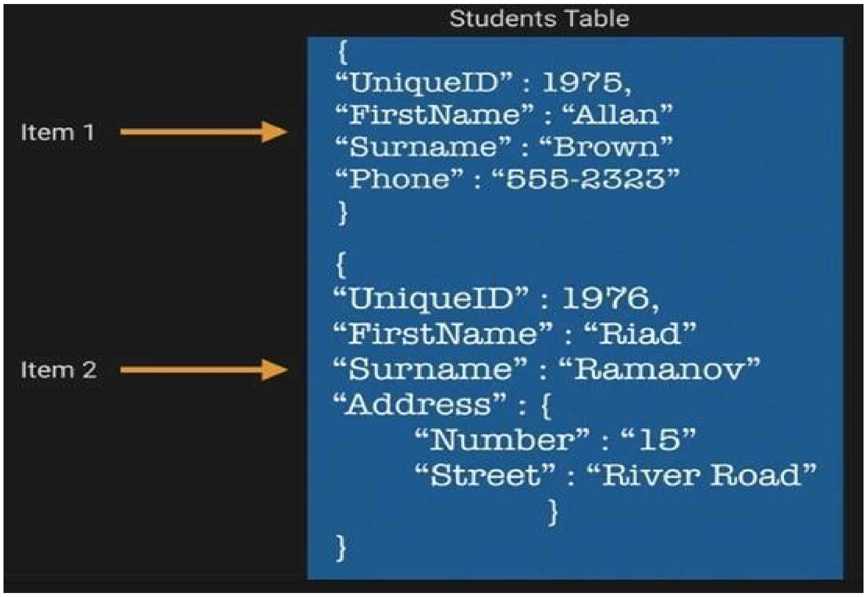
- **Partition key and sort key**– It is referred to as a composite primary key, this type of key is composed of two attributes. The first attribute is the partition key, and the second attribute is the sort key. DynamoDB uses the partition key value as input to an internal hash function. The output from 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.

How does DynamoDB work?

A DynamoDB database can be broken down into three theorems:

1. Tables: A collection of things that you want to store together
2. Items: An item is just like a row in a normal database

Attributes: A column or field in a normal database



Students Table

```
{
"UniqueID" : 1975,
"FirstName" : "Allan"
"Surname" : "Brown"
"Phone" : "555-2323"
}

{
"UniqueID" : 1976,
"FirstName" : "Riad"
"Surname" : "Ramanov"
"Address" : {
     "Number" : "15"
     "Street" : "River Road"
            }
}
```

.

. A DynamoDB table must have a primary key. There are two possible types to choose from:

1. Partition Key — Single Attribute —which will just be a field in your data source that uniquely represents the row (e.g., an auto-generated, unique product ID).
2. Partition Key & Sort Key — Composite Key — which will be a combo of two attributes that will uniquely identify the row, and how the data should naturally be sorted (e.g., Unique product ID and purchase date timestamp)

- Your DynamoDB partition key must be unique and sparse. As this key is hashed internally and used to distribute that data for storage.
- This is a similar technique to Redshift and HBase that prevents hot-spotting of data.

- If using a composite key, then two items can have the same Partition Key, but the Sort Key must be unique.

- This will mean all items with the same Partition key will be stored together but sorted in ascending order using the Sort Key.

AWS DynamoDB – Working with Queries –

- Amazon DynamoDB is a NoSQL managed database service provided by Amazon that stores semi-structured data like key-value pairs.

- A DynamoDB table consists of items. Each item consists of one partition key and one or more attributes. An example of an item is given below:

Example:

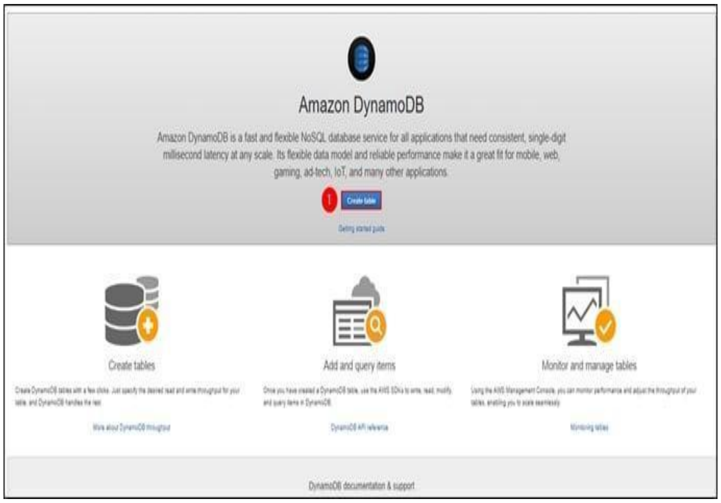
```
{
        "MovieID": 101,
         "Name": "The Shawshank Redemption",
          "Rating": 9.2,
         "Year": 1994
}
```

In the above example, MovieID is the partition key.

- A partition key is used to differentiate between items. A query operation in DynamoDB finds items based on primary key values.
- The name of the partition key attribute and a single value for that attribute must be provided. The query returns all items searched against that partition key value.

How to Create a table in DynamoDB

Step 1: Navigate to the DynamoDB section in AWS. Select "*Create Table*".



Amazon DynamoDB

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, IoT, and many other applications.

Create tables

Create DynamoDB tables with a few clicks. Just specify the desired read and write throughput for your table, and DynamoDB handles the rest.

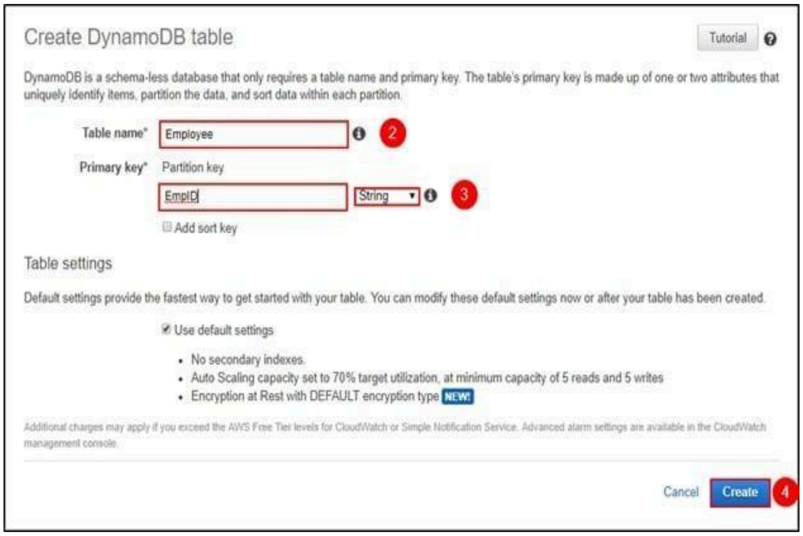
Add and query items

Once you have created a DynamoDB table, use the AWS SDKs to write, read, modify, and query items in DynamoDB.

Monitor and manage tables

Using the AWS Management Console, you can monitor performance and adjust the throughput of your tables, enabling you to scale seamlessly.

Step 2: Fill in with the necessary details and click on "*Create*."



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* Employee

Primary key* Partition key

EmpID String

□ Add sort key

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

- No secondary indexes.
- Auto Scaling capacity set to 70% target utilization, at minimum capacity of 5 reads and 5 writes
- Encryption at Rest with DEFAULT encryption type NEW!

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Step 3: You can view your table being created. Click on "*Overview*" to understand your table, click on "*Items*" to edit, insert and query on the table. There are many more options you can use to understand your table better.

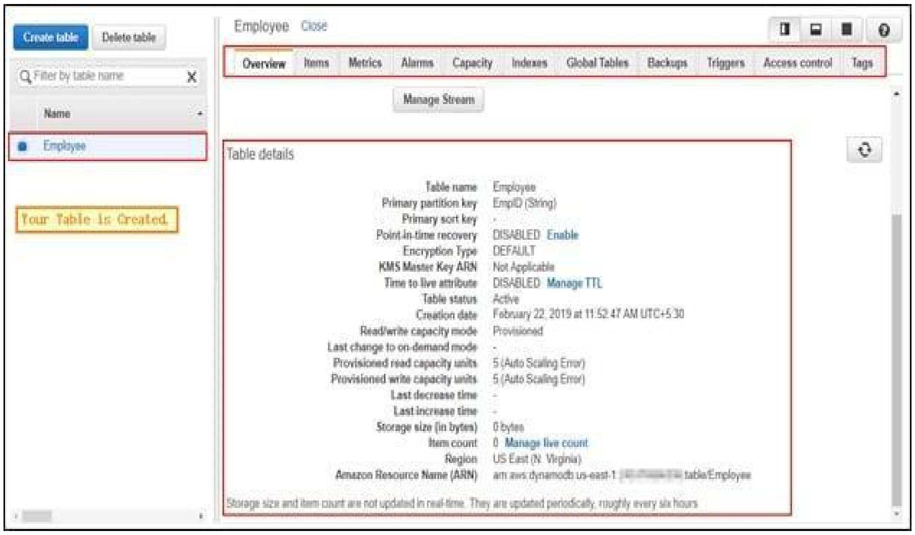


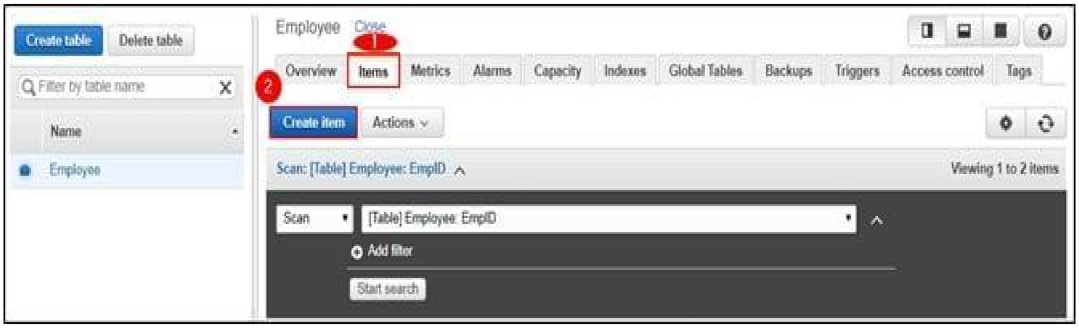
Table details

Table name	Employee
Primary partition key	EmpID (String)
Primary sort key	-
Point-in-time recovery	DISABLED Enable
Encryption Type	DEFAULT
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	February 22, 2019 at 11:52:47 AM UTC+5:30
Read/write capacity mode	Provisioned
Last change to on-demand mode	-
Provisioned read capacity units	5 (Auto Scaling Error)
Provisioned write capacity units	5 (Auto Scaling Error)
Last decrease time	-
Last increase time	-
Storage size (in bytes)	0 bytes
Item count	0 Manage live count
Region	US East (N. Virginia)
Amazon Resource Name (ARN)	arn:aws:dynamodb:us-east-1:[redacted]:table/Employee

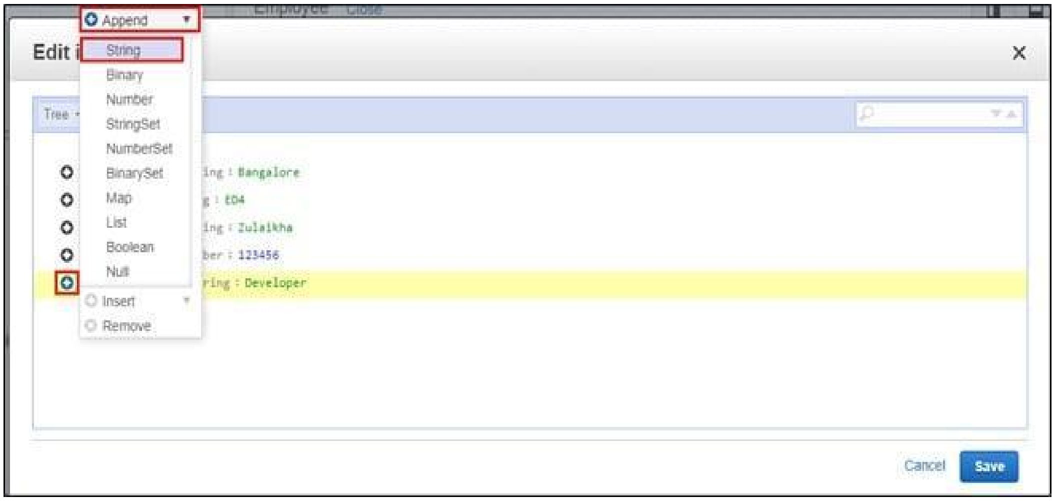
Storage size and item count are not updated in real-time. They are updated periodically, roughly every six hours.

How to Insert a table in DynamoDB?

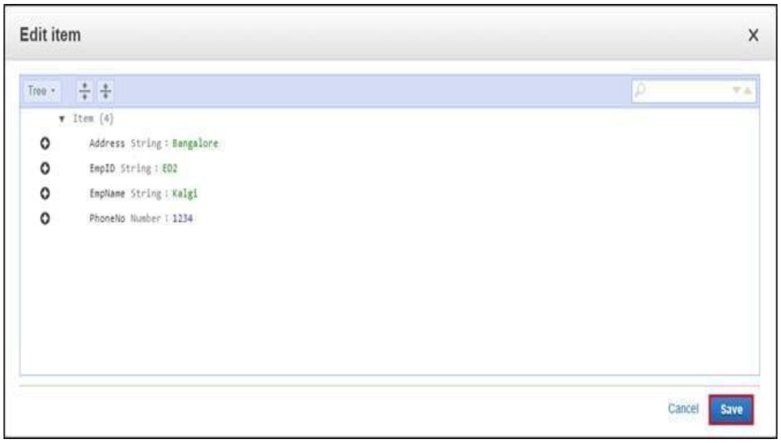
Step 1: Navigate to "*Items*" and click on "*Create item*."



Step 2: It will open a JSON file where you can add different items. Click on the "+" symbol and select "*Append*" and select what type of data you want to enter.

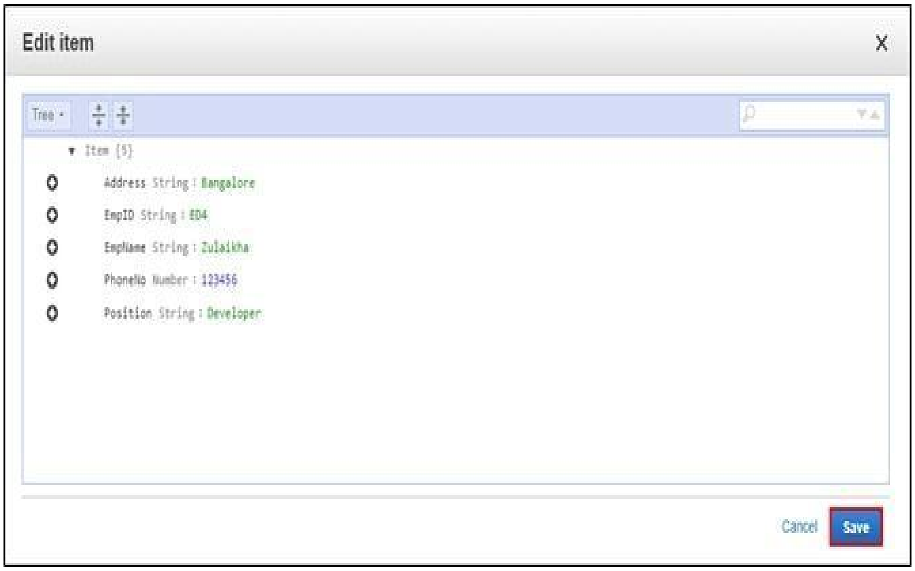


Step 3: This is what it looks like after adding multiple columns to your table. Click on "*Save*".



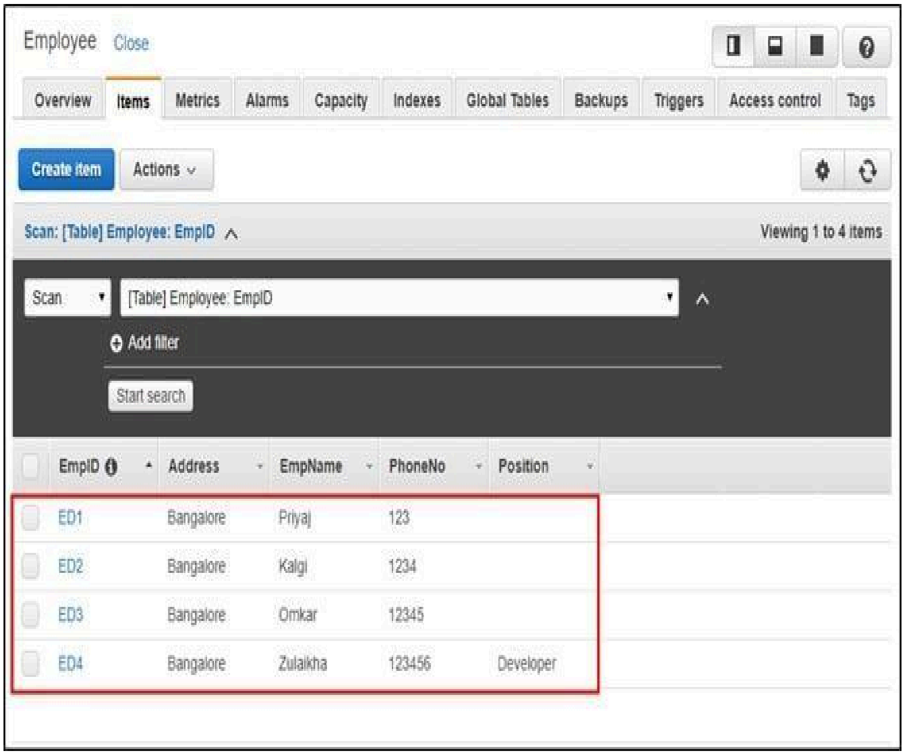
```
▾ Item {4}
    Address String : Bangalore
    EmpID String : E02
    EmpName String : Kalgi
    PhoneNo Number : 1234
```

Step 4: Since it is a NoSQL architecture, you can play around with the columns you add to the table. E.g., "*Position*."



```
▾ Item {5}
  Address String : Bangalore
  EmpID String : ED4
  EmpName String : Zulaikha
  PhoneNo Number : 123456
  Position String : Developer
```

Step 5: This is how your table will look like once you have inserted the data.



Scan: [Table] Employee: EmpID

Viewing 1 to 4 items

Scan ▾ [Table] Employee: EmpID ▾

EmpID	Address	EmpName	PhoneNo	Position
ED1	Bangalore	Priyaj	123	
ED2	Bangalore	Kalgi	1234	
ED3	Bangalore	Omkar	12345	
ED4	Bangalore	Zulaikha	123456	Developer

Source Code –

Let's assume we're building a simple application for managing books and authors. The application needs to support queries for retrieving books by title, author, and publication year.

# Define the table for books

```
CREATE TABLE Book (
  # Primary key
  BookID UUID PRIMARY KEY,

  # Attributes
  Title STRING,
  AuthorID UUID,
  PublicationYear INT,
  Genre STRING,
  Summary STRING,

  # Secondary Indexes
  GSI1 (
    # Global Secondary Index for querying by AuthorID
    AuthorID UUID
  ),
  GSI2 (
    # Global Secondary Index for querying by Title
    Title STRING
  ),
  GSI3 (
    # Global Secondary Index for querying by PublicationYear
    PublicationYear INT
  )
);

# Define the table for authors
CREATE TABLE Author (
  # Primary key
  AuthorID UUID PRIMARY KEY,

  # Attributes
  FirstName STRING,
  LastName STRING,
  BirthYear INT,

  # Secondary Index
  GSI (
    # Global Secondary Index for querying by LastName
```

```
   LastName STRING
  )
);
```

Explanation:

1. Book Table:
    - Primary Key: BookID (UUID) - Unique identifier for each book.
    - Attributes: Title, AuthorID, PublicationYear, Genre, Summary.
    - Global Secondary Indexes (GSIs) for querying by AuthorID, Title, and PublicationYear.
2. Author Table:
    - Primary Key: AuthorID (UUID) - Unique identifier for each author.
    - Attributes: FirstName, LastName, BirthYear.
    - Global Secondary Index (GSI) for querying by LastName.

Conclusion –

We have implemented Dynomo DB queries on single database schema.

FAQ:-

1) Give the name of 4 use cases of AWS DynamoDB.
2) Explain two types of primary keys used by DynamoDB.
3) Explain the differences between Amazon DynamoDB and Amazon SimpleDB.
4) Name different data types supported by DynamoDB.
5) Give 5 ways to get data from DynamoDB.
6) How can we generate Access Key and Secret Key in DynamoDB?