



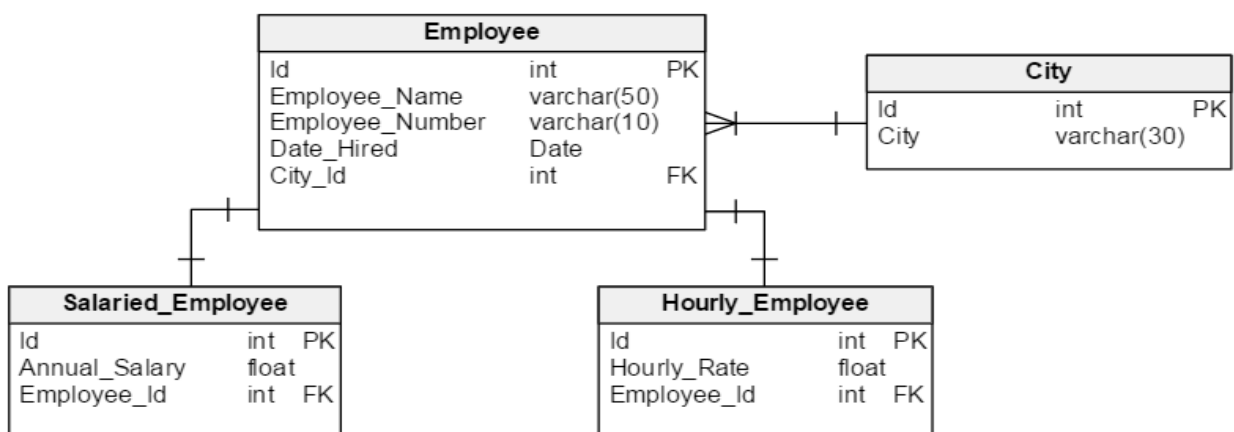
Amazon DynamoDB

DynamoDB is a fully managed NoSQL database service provided by Amazon. It provides fast and consistent performance, highly scalable architecture, flexible data structure, event driven programmability, and fine-grained access control.

Before we get into details about DynamoDB, let us understand fundamental characteristics about RDBMS/SQL and NoSQL databases.

What is an RDBMS?

RDBMS enables you to create databases in the form of tables which store related data. As you can see we have 4 tables here and are connected using a unique primary key for each table. Inside the table, data is stored here in rows and columns.



Examples of RDBMS:



What is SQL?

SQL is a standardized language to interact with relational databases. It can execute query against a database and retrieve data from one or more tables. It can insert, update, delete and retrieve data and can perform many database related activities.

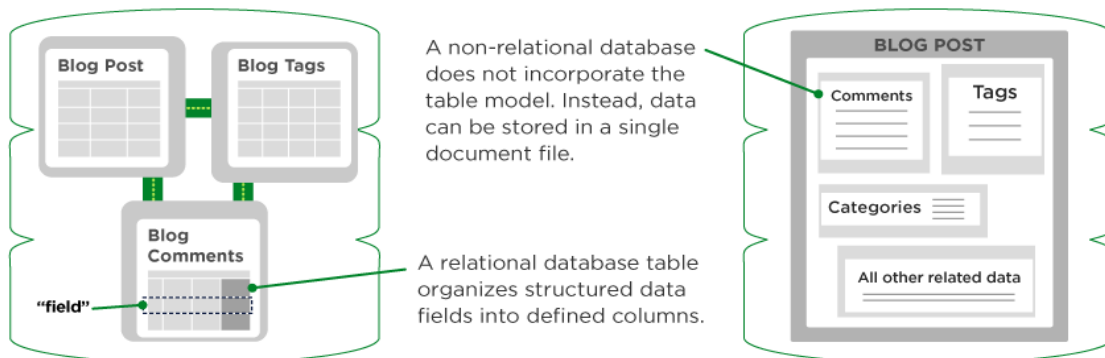
What is NoSQL?

A NoSQL database provides a way to store and retrieve data that is in a non-tabular format. It is also referred to as NonSQL, Non-Relational or Not only SQL database.

NoSQL databases are used for managing large sets of data that are frequently updated in a distributed system. It eliminates the need for rigid schema associated with an RDBMS.

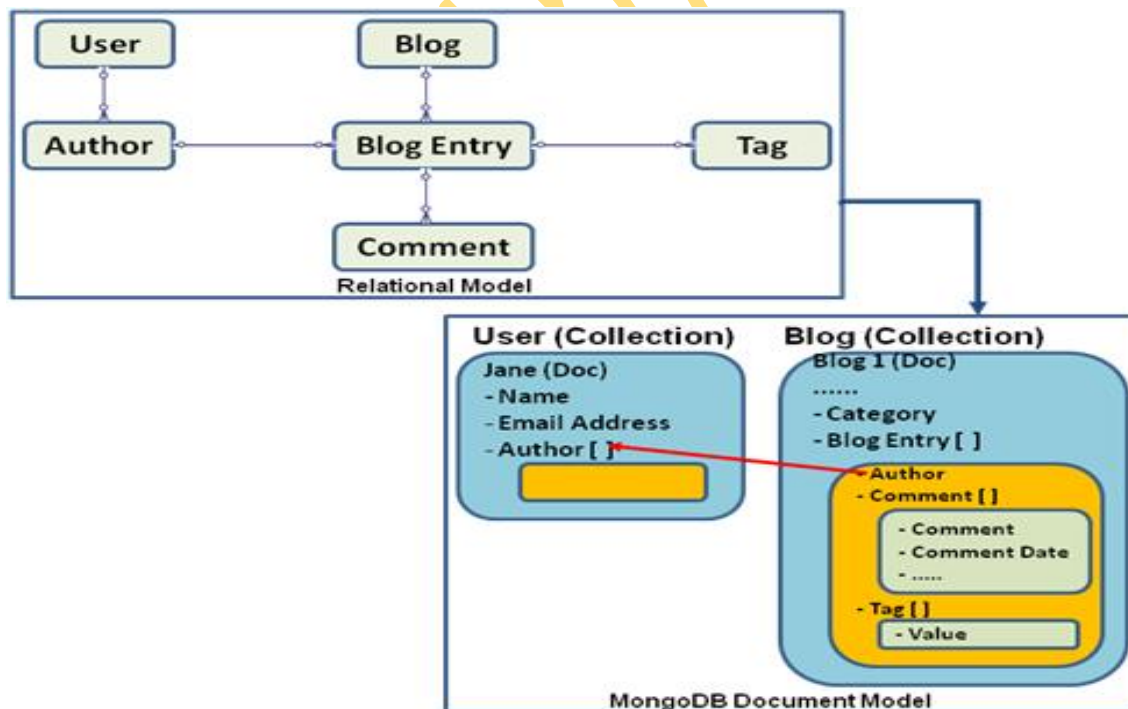
RELATIONAL VS. NON-RELATIONAL DATABASES

upwork™



If you notice the above diagram for a Blogpost in RDBMS we need to write separate tables for Blog Post, Tags and Comments but where as in NoSQL all data can be written in one document.

The below diagram explains you the difference between a relational model and NoSQL(MongoDB) model.



So, in sum:

SQL deals with structured data and NoSQL deals with unstructured data.

SQL has a static schema and NoSQL has a dynamic Schema, which is why is very flexible.

Types of NoSQL databases:

- 1.Key-value pair database
- 2.Document Databases
- 3.Graph Databases
- 4.Wide column stores

1.Key-Value pair databases:

It uses a very simple data model that stores data in a pair of unique keys and the associated value. Commonly, it is used for storing time series data, click stream data and application logs.

Example of key value pairs: DynamoDB,Redis,and Aerospike.

key	value
firstName	Bugs
lastName	Bunny
location	Earth

2.Document databases:

It stores data in a structure that represents a document-like format such as JSON, XML and YAML. Document databases are used for content management and monitoring applications.

Examples of document databases: MongoDB, CouchDB, MarkLogic, and so on.

MongoDB schema written in JSON:

Relational

Customer ID	First Name	Last Name	City
0	John	Doe	New York
1	Mark	Smith	San Francisco
2	Jay	Black	Newark
3	Meagan	White	London
4	Edward	Daniels	Boston

Phone Number	Type	DNC	Customer ID
1-212-555-1212	home	T	0
1-212-555-1213	home	T	0
1-212-555-1214	cell	F	0
1-212-777-1212	home	T	1
1-212-777-1213	cell	(null)	1
1-212-888-1212	home	F	2



MongoDB

```
{  customer_id : 1,
  first_name  : "Mark",
  last_name   : "Smith",
  city        : "San Francisco",
  phones: [ {
    number : "1-212-777-1212",
    dnc    : true,
    type   : "home"
  },
  {
    number : "1-212-777-1213",
    type   : "cell"
  }
]
```

Tables in MongoDB are referred as collections.

You can see how different tables in RDBMS is written as a single schema in MongoDB. In this way NoSQL offers flexibility in writing a schema.

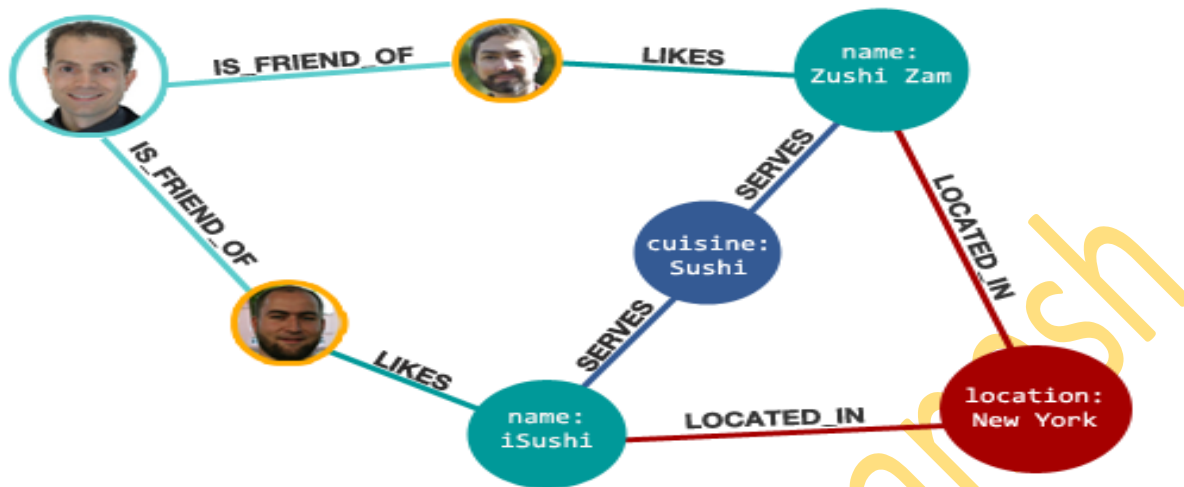
3. Graph Databases:

A graph database is a NoSQL database that uses graph structures and stores related data in nodes. It emphasizes on the connection between the data elements to accelerate query performance.

It is mainly used to store geographical data and recommendation search engines.

Examples of Graph databases: Allegrograph, IBM graph, Neo4J

Graph databases illustration:



The relationships allow data in the nodes to be linked together directly, and in many cases retrieved with one operation.

4. Wide column databases:

This database stores data using a column oriented model. It is also called table-like database or column store database. It stores data in a table-like structure and it can store large number of columns.

Wide column databases are generally used for storing data related to internet search and other similar large-scale web applications

Column Families				
row key	personal data		professional data	
employee ID	Name	City	Title	Salary
1342	Joseph	New York City	Product Manager	\$150,000
1543	Frank	Boston	Software Engineer	\$160,000
7643	David	San Francisco	Data Scientist	\$130,000

Examples: HBase,Cassandra,SimpleDB etc

In Wide column database, each column is stored in a separate file as depicted here.

Introducing DynamoDB:

What is DynamoDB?

DynamoDB is a fully-managed NoSQL database provided by AWS. Fully-managed means:

- It requires no database admin
- It requires no servers to manage and no levers to tune
- It requires no manual backup.

Everything is taken care of AWS

All you have to do is :

- Setup tables and,
- configure the level of **provision throughput** that each table should have

Provision throughput refers to the level of read and write capacity that you want AWS to reserve for the table. You're charged for the total amount of throughput that you configure for tables and storage space you used for the table.

DynamoDB is a key value store NoSQL. A key value store is a collection of items. You can look up to the data using **primary keys or indexes**.

It also takes care of fault tolerance by replicating the data in DynamoDB in **3 regions**.

DynamoDB components:

There are basically 3 components in a DynamoDB table: tables, items, and attributes.

1.Tables:

DynamoDB stores data in tables. This tables consists of items, which forms a group of attributes

2.Item:

A table consists of multiple items .An item is just like a record or row in RDBMS.

3.Attributes:

Each item consists of group of attributes. It is similar to 'fields' in RDBMS.

If you see the table 'People' here:



An each Item consists of several attributes like 'Person ID' , 'Last Name' , 'First Name' and 'Phone', 'Address' and 'Favorite Color'

As you can see attribute '**Address**' is not in **first item** and '**Favorite Color**' is not in **second item**. This is what DynamoDB is famous for: **flexible schema**

Primary Key:

Primary key uniquely identifies each item in the table. There are two types of primary keys:

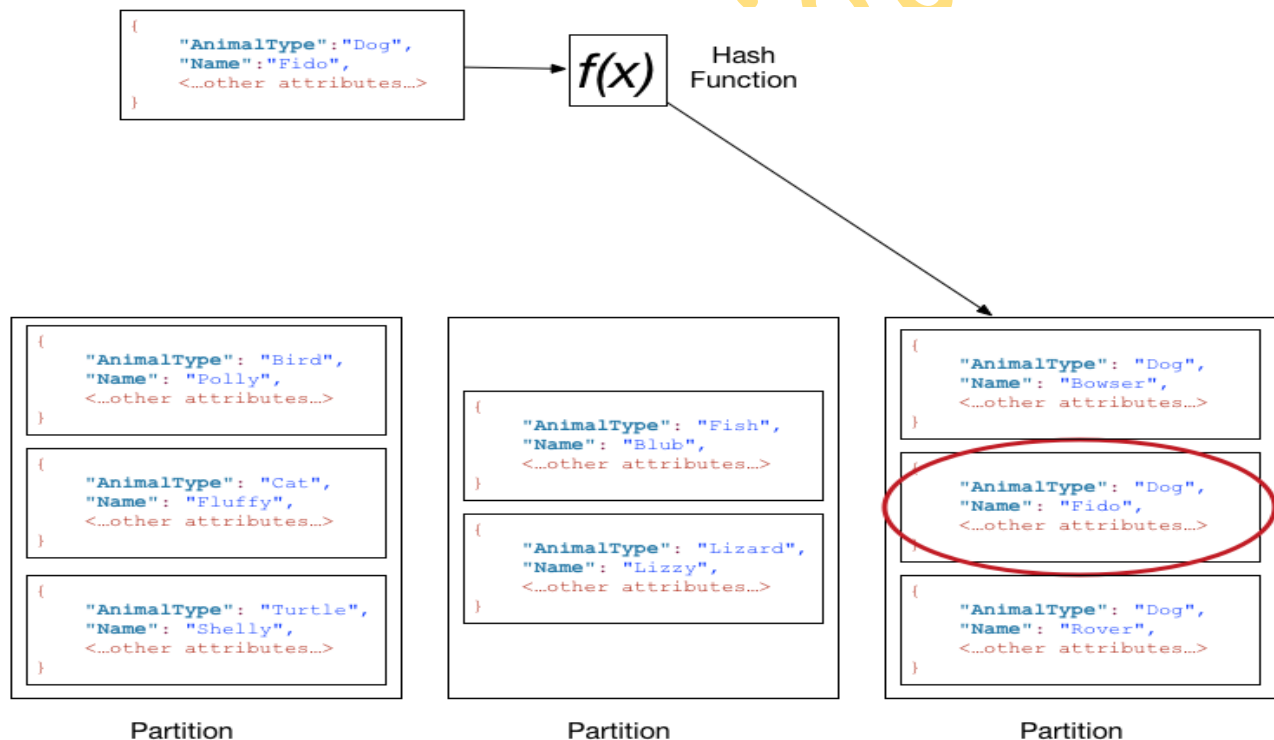
1.Partition key

2.Partition key and sort key

1.Partition key (also called as Hash key):

- a. DynamoDB partitions data in sections based on partition key value.
- b.Partition key value is used as an input to an internal hash function, which determines in which partition the data is stored.
- c. Which is why it is also called as hash key
- d.No two items in a table can have same partition key

illustration:



AnimalType is the partition key and the value is **'dog'** which is the input to the hash function $f(x)$ and this $f(x)$ determines in which partition **'AnimalType:dog'** is stored. So, retrieval of data is easy.

2.Partition key and sort key:

- a.It is composed of two attributes as its name says
- b.It is also called as composite key
- c.Just like the partition key,the composite key also uses partition key as input to internal hash function.This hash function determines the place of the item in partition.
- d.In partitions,the items are stored based on a sort key value.A sort key is also called as **range key**.No items can have same sort key but need not have an unique partition key.

Illustration:

```
{  
  Department_id: "D91007"  
  Employee_id : "20001"  
  Department_name : "R & D"  
  Employee_name : "Sampath"  
  Joiningdate: "2016-04-04"  
}
```

In the above table:

Partition key : Department_id

Sort key : Employee_id

As you can see,data retrieval is fast with the use of both keys.

A table with both partition key and sort key can have more than one item with the same partition key,but it must have a **different sort key**.

Here you can the department_id can be **same** for many employees but employment_id must be **unique**.

Indexes:

DynamoDb allows you to create secondary indexes(which are optional) on a table.It is an alternative way to query table data in addition to querying it using primary key.

DynamoDB supports two types of secondary indexes:

Local secondary index:

1. It has the same partition key but different sort key.
2. Can only be created when creating a table.
3. lets you query over a single partition, as specified by the partition key value in the query.
4. you can choose either eventual consistency or strong consistency.
5. For each partition key value, the total size of all indexed items must be 10 GB or less.

Global secondary index:

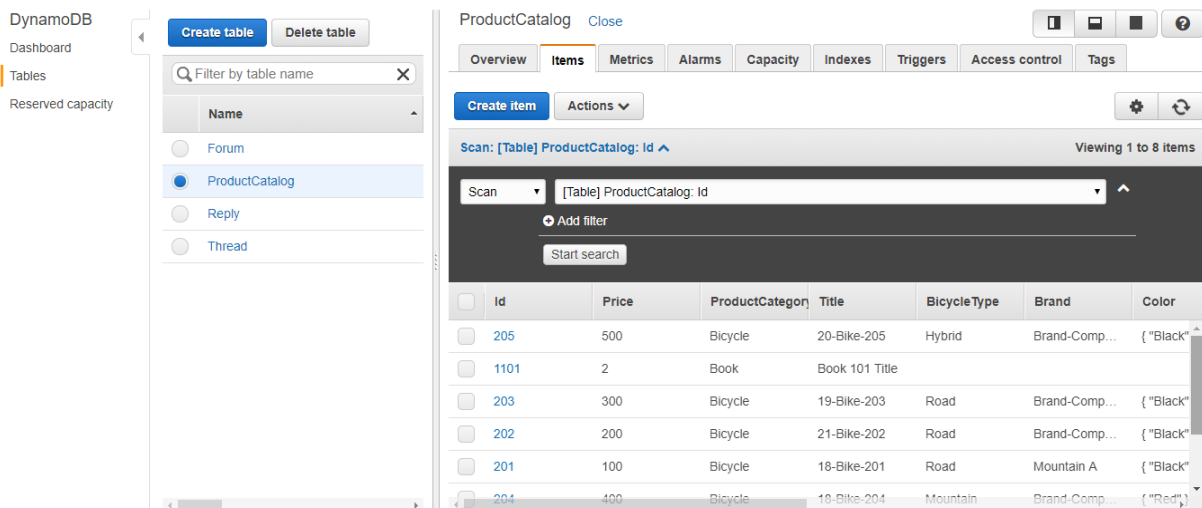
1. It has different partiton key and different sort key.
2. Can be created at table creation or later.
3. lets you query over the entire table, across all partitions.
4. support eventual consistency only
5. no size restrictions for global secondary indexes

You can create up to 5 global secondary indexes and up to 5 local secondary indexes per table.

Query :

A query operation finds items in a table using only primary key attribute values. You must provide a partition attribute name and a distinct value to search for.

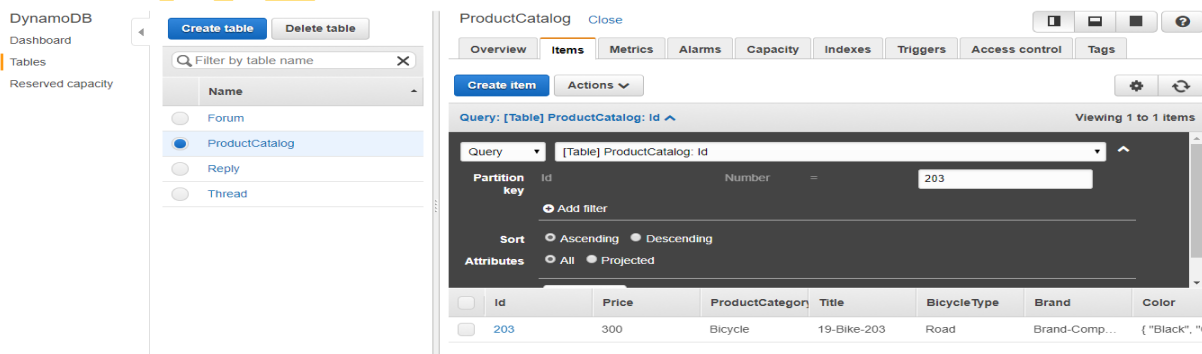
If you notice the DynamoDB table 'Product Catalog', you can search the items of the Product Catalog using the primary key 'id' and providing value to it.



The screenshot shows the AWS DynamoDB console interface. On the left, the 'DynamoDB' dashboard is visible with the 'Tables' section selected. The main panel displays the 'ProductCatalog' table. The 'Items' tab is active, showing a list of items. The table has the following columns: Id, Price, ProductCategory, Title, BicycleType, Brand, and Color. The items are listed as follows:

Id	Price	ProductCategory	Title	BicycleType	Brand	Color
205	500	Bicycle	20-Bike-205	Hybrid	Brand-Comp...	{ "Black"
1101	2	Book	Book 101 Title			
203	300	Bicycle	19-Bike-203	Road	Brand-Comp...	{ "Black"
202	200	Bicycle	21-Bike-202	Road	Brand-Comp...	{ "Black"
201	100	Bicycle	18-Bike-201	Road	Mountain A	{ "Black"
204	400	Bicycle	18-Bike-204	Mountain	Brand-Comp...	{ "Red"

And in this table I searched for the item 203 as shown below, the id is the partition key.



The screenshot shows the AWS DynamoDB console interface. On the left, the 'DynamoDB' dashboard is visible with the 'Tables' section selected. The main panel displays the 'ProductCatalog' table. The 'Query' tab is active, showing a query builder interface. The query is set to search for the item with Id 203. The results show the item with Id 203.

Id	Price	ProductCategory	Title	BicycleType	Brand	Color
203	300	Bicycle	19-Bike-203	Road	Brand-Comp...	{ "Black", "C

However, I can provide sort key value and use comparison operator to refine my search, but it's optional.

As you can see that query returned all the attributes in that item, however you can use **projection expression** to retrieve only few attributes.

By default, Query is eventually consistent but can be changed.

Scan:

A scan operation performs every item in the table. By default, a scan returns all of the data attributes for every item; however, you can use Projection Expression parameter so that scan only returns some of the attributes but not all of them.

Query vs Scan: which one is best

A scan operation scans entire table, then it produces the result. For large tables, a scan operation can use up the provisioned throughput for a single operation resulting in latency.

So, query is more efficient than scan.

Provisioned throughput:

DynamoDB allows us to set Read and Write Provisioned throughput. By default, AWS provisions 5 Read capacity and 5 write capacity units.

Read capacity:

1 read capacity = 1 strongly consistent read

1 read capacity = 2 eventual consistent reads

1 read operation process = item up to 4 KB in size

If an item is more than 4 KB, it requires additional write capacity

If an item is less than 1 KB, it still requires 1 read capacity units

Write capacity:

1 write capacity = 1 write operation up to 1 KB in size

If an item is larger than 1 KB, it requires additional write capacity units

If an item is less than 1 KB, it still requires 1 write capacity

So, if you create a table (by default it comes with a throughput of 5 Read capacity and 5 Write capacity units)

Strongly consistent reads:

Since 1 Read capacity = 1 strongly consistent read of 4KB

So, 5 read capacity units each of 4 KB = $5 \times 1 \times 4\text{KB} = 20\text{ KB per second}$

Hence, one table can perform a strongly consistent read of upto 20 KB per second

Eventual consistent reads:

Since 1 read capacity = 2 eventual consistent reads

So, 5 Read capacity units of 4 KB = $5 \times 2 \times 4\text{ KB} = 40\text{ KB per second}$

Hence, one table can perform an eventual consistent read of upto 40 KB per second

Write capacity:

Since, 1 write capacity = 1 write operation upto 1 KB in size

5 write capacity units can process = $5 \times 1 \times 1\text{KB} = 5\text{ KB per second}$

Examples:

1. You have an application that requires reading 15 items per second. Each item is of 3 KB in size. If the application requires strongly consistent reads, what is the read capacity required to address this need?

Sol: Since, 1 Read capacity = 1 strongly consistent read of 4KB

Here Item size = 3 KB, but still it requires 1 read capacity to process that read.

So item size is rounded-off to multiple of 4. So, item size = 4K

Read throughput for strongly consistent reads = (item size rounded-up in multiples of 4 KB) / 4KB x number of items

=> Read throughput = $4/4 \times 15$ (Item size is 3 B but rounded up to 4)

$$= 1 \times 15 = 15/1 = 15$$

The answer is 15 read capacity units.

2. You have an application that requires reading 80 items per second. Each item is 5 KB in size. If the application requires using strongly consistent read, what is the read capacity?

Sol:

Item size = 5KB. Now, Item size should be rounded to multiples of 4KB .

Since, item size is more than 4KB .So, nearest number is 8

Hence ,item size after rounding up = 8 KB

Since it is a **strongly consistent read**:

$$\text{Read throughput} = (\text{item size rounded-up in multiples of 4 KB}) / 4 \text{ KB} \times \text{number of items}$$

Therefore, for strongly consistent reads

$$\text{Read throughput} = 8 (\text{item size rounded-up}) / 4 \times 80 = 160$$

The answer is 160 read capacity units.

3. You have an application that requires reading 60 items per second .Each item is 3 KB size and an application requires eventual consistent read. What is the read capacity?

Sol:

item size = 3KB ,if rounded then item size = 4KB

therefore for **eventual consistency**,

$$\text{Read throughput} = [(\text{item size rounded-up in multiples of 4 KB}) / 4 \text{ KB} \times \text{no.of items}] / 2$$

$$= (4/4 \times 60) / 2 = 30$$

The answer is 30 read capacity units.

4. You have an application that writes 10 items per second with each item being 8 KB in size. How many write capacity units are required to address this need?

Ans:

1 write capacity = 1 write operation up to 1 KB in size

Hence write throughput = (Item size rounded in multiples of 1 KB) / 1KB x number of items

$$\text{So write throughput} = 8/1 \times 10 = 80$$

The answer is 80 write capacity units.

More examples:

5. You have an application that requires to read 5 items of 10 KB per second using

(i) What is the read throughput :

a) strong consistency b) eventual consistency

(ii) What is the write capacity?

What is the read throughput

Ans: Item size = 10 KB, if rounded up to nearest multiples of 4

Then, item size = 12 KB

(i) (a) read capacity for strongly consistent read = $12/4 \times 5 = 15$ read capacity units

so read capacity for strongly consistent = 15 units

(b) read capacity for eventual consistent read = $(12/4 \times 5) / 2 = 7.5$ read capacity
but read capacity units are always integers

So read capacity for eventual consistent= 8 units

(ii) since 1 write capacity unit = 1 write operation upto 1 KB

Therefore, write throughput = $10(\text{rounded upto nearest multiples of 1 KB})/1 \times 5 = 50$

So, write capacity = 50 units

6.You have an application that requires to write 12 items of 100 KB per item each second.What is the write throughput?

Ans: **since 1 write capacity unit = 1 write operation upto 1 KB**

Therefore,Write capacity = $100(\text{rounded to nearest multiples of 1 KB})/1 \times 12$

So,write capacity = 1200 units

400 HTTP Status code – ProvisionedThroughputExceededException

You'll get this error message when you exceed your maximum allowed provisioned throughput for a table or for one or more GSI

Lab 1 :Intro to DynamoDb

Step:1 Create a DynamoDB table named 'Music' with primary key 'Artist' and sort key 'Song' and select 'string' for both keys and click Create.

Create DynamoDB table Tutorial ?

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* Music ?

Primary key* Partition key

Artist String ?

☒ Add sort key

Song 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

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".

? You do not have the required role to enable Auto Scaling by default. Please refer to documentation.

[Feedback](#) [English \(US\)](#) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

The table will be created in less than a minute.

DynamoDB

[Create table](#) [Actions](#)

Filter by table name

Name

Music

Music [Close](#)

Overview [Items](#) [Metrics](#) [Alarms](#) [Capacity](#) [Indexes](#) [Triggers](#) [Access control](#) [Tags](#)

Table is being created

Recent alerts

No CloudWatch alarms have been triggered for this table.

Stream details

Stream enabled No

View type -

Latest stream ARN -

[Manage Stream](#)

Table details

Table name Music

Primary partition key Artist (String)

Primary sort key Song (String)

Time to live attribute [Manage TTL](#)

Table status Creating

Creation date November 9, 2017 at 12:31:50 AM UTC+5:30

Provisioned read capacity units 5 (Auto Scaling Disabled)

[Feedback](#) [English \(US\)](#) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

Step 2: Add data:

We have a table named 'Music' now we'll add **items**. Each table consists of multiple items. **Items** are collections of individual records called '**attributes**'.

Now click on items tab and click create item.

You will already notice Artist and Song value. Now fill the data as shown in below screenshot. To append **additional data** as shown below click on Append and keep 'string' as the setting and 'number' for the year and click on **save**.

Tree ▾

Item {4}

- Album String : Dil to Pagal hai
- Artist String : Lata Mangeshkar
- Song String : Dil to Pagal hai
- Year Number : 1996

Cancel Save

Now add three more items as shown:

Item 2:

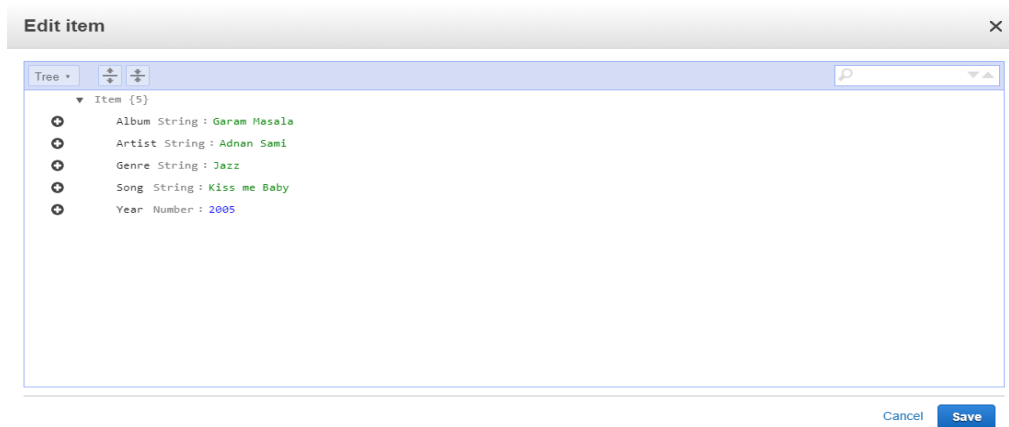
Tree ▾

Item {5}

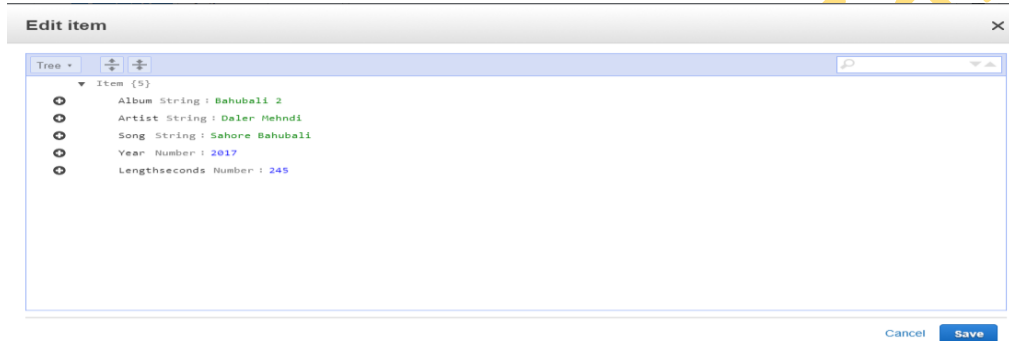
- Album String : Kal ho Na Ho
- Artist String : Sonu Nigam
- Genre String : semi classical
- Song String : Kal ho Na Ho
- Year Number : 2003

Cancel Save

Item 3:

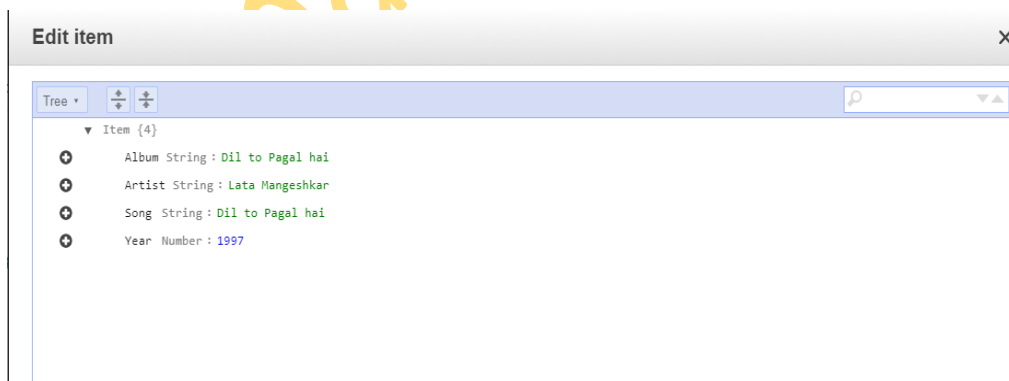


Item 4: New attribute 'Lengthseconds' with data type Number is created. This is the advantage of having a schemaless database design.



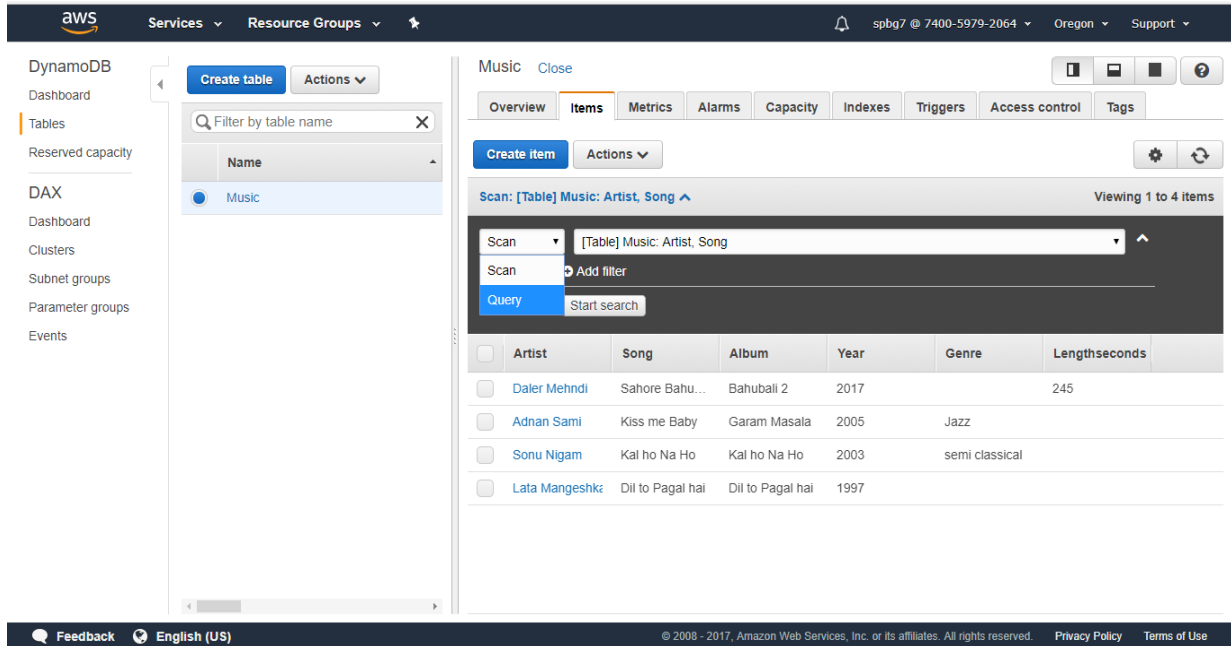
Step 3: Now modify the data

Modify the table Lata Mangeshkar and change the year from 1996 to 1997



Step 4: Now query the table

Now click on the scan below the create item tab and change it to 'query'



The screenshot shows the AWS DynamoDB console interface. On the left, the navigation menu includes DynamoDB, Dashboard, Tables, Reserved capacity, DAX, Dashboard, Clusters, Subnet groups, Parameter groups, and Events. The main panel shows the 'Music' table with tabs for Overview, Items, Metrics, Alarms, Capacity, Indexes, Triggers, Access control, and Tags. The 'Items' tab is active, displaying a 'Create item' button and a 'Scan' dropdown menu. The dropdown is set to 'Query' (previously 'Scan'). Below the dropdown, there is a search bar and a 'Start search' button. A table of items is displayed, showing columns for Artist, Song, Album, Year, Genre, and Lengthseconds. The items listed are:

Artist	Song	Album	Year	Genre	Lengthseconds
Daer Mehndi	Sahore Bahu...	Bahubali 2	2017		245
Adnan Sami	Kiss me Baby	Garam Masala	2005	Jazz	
Sonu Nigam	Kal ho Na Ho	Kal ho Na Ho	2003	semi classical	
Lata Mangeshkar	Dil to Pagal hai	Dil to Pagal hai	1997		

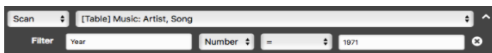
Now fields for **partiton key** and **sort key** will appear in that enter the details:

Partition key : Sonu Nigam

Sort key : Kal ho Na ho

And click start search the item you asked for is retrieved.

Alternatively, you can **scan** the element too. Now change the mode to scan and you will see this tab



The screenshot shows the 'Scan' dropdown menu with the 'Filter' tab selected. The filter is set to 'Year' with a value of '1997'.

Now for enter the attribute type 'year'

Change the string to 'number' and enter the value as '2017'

Click start 'search' and You will get the output.

Step5 :Delete the table

Now delete the table by clicking on **Action** menu beside Create table and click **Delete table**

Conclusion: So, you have an idea about CRUD operations in DynamoDB table

C -Create U-Update

R- Read D-Delete

Lab 2 – Creating DynamoDB table using AWS SDK(PHP)

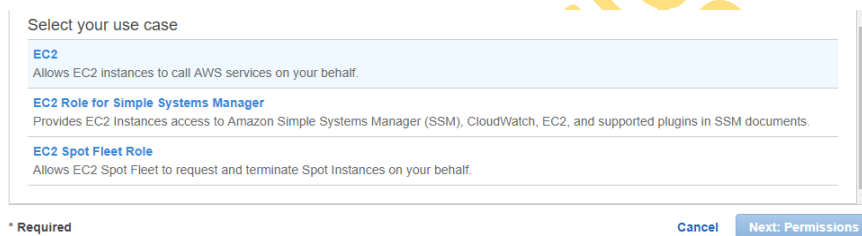
Step 1: Create an IAM role

Login to AWS console and navigate to IAM and create a role and assign DynamoDB fullaccesspolicy to the role.

Click on Roles in IAM and click on Create Role and chose the service as EC2



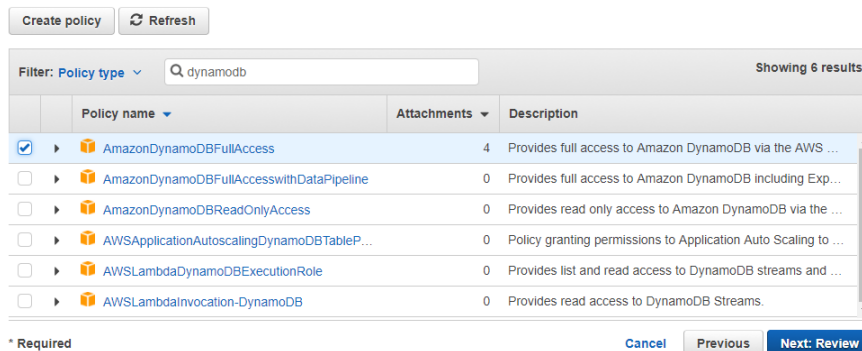
And select your usecase as EC2 and click Next permissions



Now in the permissions search 'dynamoDb' and attach AmazonDynamoDBFullAccess policy

Attach permissions policies

Choose one or more policies to attach to your new role.



Now enter the role name and click 'Create Role'

Create role



Review

Provide the required information below and review this role before you create it.

Role name*
Maximum 64 characters. Use alphanumeric and '+', '@', '_' characters.

Role description
Maximum 1000 characters. Use alphanumeric and '+', '@', '_' characters.

Trusted entities AWS service: ec2.amazonaws.com

Policies AmazonDynamoDBFullAccess [View](#)

* Required

[Cancel](#) [Previous](#) [Create role](#)

Step 2: Launch an EC2 instance which uses IAM role and bash script(installs LAMP and stores createtables.php and uploaddata.php)

In configure instance details chose IAM role as the one we create now

Number of instances [Launch into Auto Scaling Group](#)

Purchasing option ☐ Request Spot instances

Network [Create new VPC](#)

Subnet [Create new subnet](#)

Auto-assign Public IP

IAM role [Create new IAM role](#)

Shutdown behavior

Enable termination protection

Monitoring

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Storage](#)

[Feedback](#) [English \(US\)](#)

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

And click on Advanced details underneath Tenancy in the same page and paste the info given in .txt file

Prepared by Sriganesh

aws Services Resource Groups

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Subnet: No preference (default subnet in any Availability Zone) [Create new subnet](#)

Auto-assign Public IP: Use subnet setting (Enable)

IAM role: DynamoDBLab [Create new IAM role](#)

Shutdown behavior: Stop

Enable termination protection: ☐ Protect against accidental termination

Monitoring: ☐ Enable CloudWatch detailed monitoring
Additional charges apply.

Tenancy: Shared - Run a shared hardware instance
Additional charges will apply for dedicated tenancy.

▼ Advanced Details

User data: ☒ As text ☐ As file ☐ Input is already base64 encoded

```
#/bin/bash
yum update -y
yum install httpd24 php66 git -y
service httpd start
chkconfig httpd on
```

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Storage](#)

Now click on Next:Add Storage ,Next:Add tags and Next:Add security and click on Configure security group and enable HTTP on port 80.Then launch the instance

Launch Status

Your instances are now launching

The following instance launches have been initiated: [i-01d834eea07004056](#) [View launch log](#)

Step 3: SSH into the instance

Now connect to your instance using putty or ubuntu on VMWare

```
spbg7@ubuntu:~/Downloads$ chmod 400 a1.pem
spbg7@ubuntu:~/Downloads$ ssh -i "a1.pem" ec2-user@ec2-52-34-92-161.us-west-2.co
mpute.amazonaws.com
The authenticity of host 'ec2-52-34-92-161.us-west-2.compute.amazonaws.com (52.3
4.92.161)' can't be established.
ECDSA key fingerprint is SHA256:iCimt7LtqCTQbMx6bhwUnQBqo7K5PwQDQ5VXJDTrzuI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-52-34-92-161.us-west-2.compute.amazonaws.com,52.
34.92.161' (ECDSA) to the list of known hosts.

  _ _ | _ _ | _ )
 _ | ( _ _ /   Amazon Linux AMI
 _ _ | \ _ _ | _ _ |

https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/
[ec2-user@ip-172-31-20-66 ~]$
```

Now as you can see in /var/www/html/ I have two files one is test.php and other one is dynamodb


```
[ec2-user@ip-172-31-20-66 ~]$ cd /var/www/html/  
[ec2-user@ip-172-31-20-66 html]$ ls  
dynamodb test.php  
[ec2-user@ip-172-31-20-66 html]$
```

Now install and run php composer by pasting the following links on Command Line

```
Install composer : curl -sS https://getcomposer.org/installer | php
```

```
Run Composer: php composer.phar require aws/aws-sdk-php
```

```
[ec2-user@ip-172-31-20-66 html]$ sudo su  
[root@ip-172-31-20-66 html]# curl -sS https://getcomposer.org/installer | php  
All settings correct for using Composer  
Downloading...
```

```
[root@ip-172-31-20-66 html]# php composer.phar require aws/aws-sdk-php  
Do not run Composer as root/super user! See https://getcomposer.org/root for details
```

As you can see composer is installed and running

Step 4: Edit the DynamoDb files

Now goto dynamodb directory using cd command as shown and you will find 3 files like this:

```
[root@ip-172-31-20-66 html]# ls  
composer.json composer.lock composer.phar dynamodb test.php vendor  
[root@ip-172-31-20-66 html]# cd dynamodb  
[root@ip-172-31-20-66 dynamodb]# ls  
createtables.php README.md uploaddata.php  
[root@ip-172-31-20-66 dynamodb]# nano createtables.php  
[root@ip-172-31-20-66 dynamodb]#
```

As you can see from the screenshot there are 3 .php files:

1. createtables.php : php code for creating a table
2. README.md : Introduction file

3. uploaddata.php : php code for uploading data into those tables as created from createtables.php

Now edit createtables.php by using nano editor by typing command nano createtables.php as shown above and it will take you to that file

Now edit the region as per your requirement.

```
// Find out what the issues are:
ini_set('display_errors',1);
ini_set('display_startup_errors',1);
error_reporting(-1);

require '/var/www/html/vendor/autoload.php';
use Aws\DynamoDb\DynamoDbClient;

$client = DynamoDbClient::factory(array(
    'region' => 'eu-west-2', // replace with your desired region visit http
    'version' => '2012-08-10' // Now needs a version
));
```

After editing the php file save and exit from the file by typing ctrl o and ctrl x

Now edit the same for uploaddata.php by typing nano uploaddata.php. Make sure that regions in createtables.php and uploaddata.php reflects the same region.

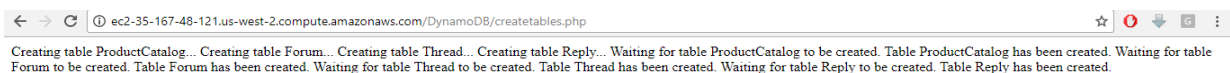
Step 5: Run those files using browser

Now copy the DNS of your EC2 instance and paste it in a different browser window and navigate the path of DynamoDB as shown:

<http://ec2-35-167-48-121.us-west-2.compute.amazonaws.com/DynamoDB/createtables.php>

in the above link DNS name should be replaced with yours but the path ec2-name/DynamoDB/createtables.php should be kept the same way and hit Enter.

You will see that tables are created as that script gets executed.



Now run the uploaddata.php script the same way as shown:

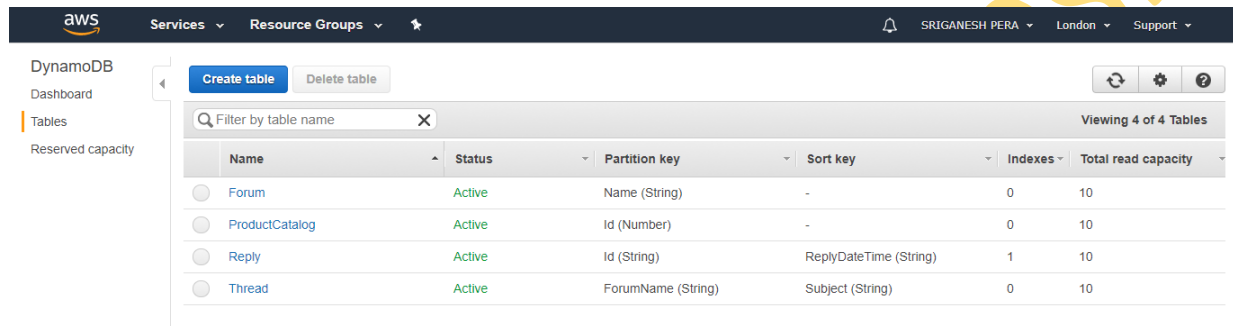
ec2-35-167-48-121.us-west-2.compute.amazonaws.com/DynamoDB/uploaddata.php

Adding data to the ProductCatalog table... done. Adding data to the Forum table... done. Adding data to the Reply table... done.

Now change the region to London and you will see your table created with data uploaded in it.

Mindcheck: Why London region? Why not US region?

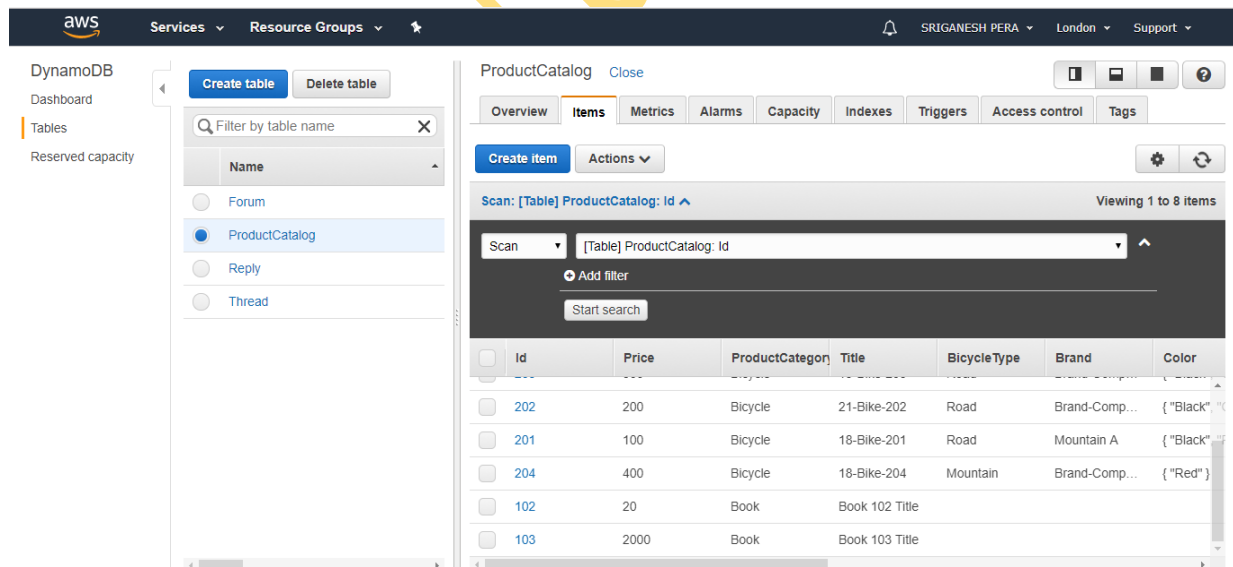
As you can see I've 4 tables created in London Region.



The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with 'DynamoDB', 'Dashboard', 'Tables', and 'Reserved capacity'. The main area has a 'Create table' button and a 'Delete table' button. Below them is a search bar 'Filter by table name'. A table lists 4 tables:

Name	Status	Partition key	Sort key	Indexes	Total read capacity
Forum	Active	Name (String)	-	0	10
ProductCatalog	Active	Id (Number)	-	0	10
Reply	Active	Id (String)	ReplyDateTime (String)	1	10
Thread	Active	ForumName (String)	Subject (String)	0	10

Feel free to check in the items of the tables, and try to create global indexes to REPLY table and also play on with Capacity Tab, Metrics Tab and Access Control tab. Also try to know the difference between Scan and Query.



The screenshot shows the AWS DynamoDB console interface with the 'ProductCatalog' table selected. The 'Items' tab is active, showing a list of items. The table has columns: Id, Price, ProductCategory, Title, BicycleType, Brand, and Color. The items are:

Id	Price	ProductCategory	Title	BicycleType	Brand	Color
202	200	Bicycle	21-Bike-202	Road	Brand-Comp...	{ "Black", "R"
201	100	Bicycle	18-Bike-201	Road	Mountain A	{ "Black", "R"
204	400	Bicycle	18-Bike-204	Mountain	Brand-Comp...	{ "Red", "R"
102	20	Book	Book 102 Title			
103	2000	Book	Book 103 Title			

Step 6: Terminate the instances

Prepared by Sriganesh

After you've played around on DynamoDb delete the instance we have created.

~~~~~ End of Lab sessions ~~~~~

Prepared by Sriganesh