**DynamoDB**

**what**?

* NOSQL database .works on key value pair.
* provided by AWS.

**WHY**?

* Scalability: scales seamlessly no matter of the traffic
* Customizable: according to the priorities of app database is customizable
* datatypes: can stored multiple data types
* Stable Performance: doesn't matter the traffic size, user get stable performance.

COMPONENTS

* a secondary index can be used as attribute, if we don't want to use primary key.
* Dynamo Streams is an additional/optional feature provided by Dynamo DB to keep track of data modification events in a table. Here each event is represented by a stream record.

* **Limitations**

DynamoDB does suffer from certain limitations, however, these limitations do not necessarily create huge problems or hinder solid development.

You can review them from the following points −

* Capacity Unit Sizes − A read capacity unit is a single consistent read per second for items no larger than 4KB. A write capacity unit is a single write per second for items no bigger than 1KB.
* Provisioned Throughput Min/Max − All tables and global secondary indices have a minimum of one read and one write capacity unit. Maximums depend on region. In the US, 40K read and write remains the cap per table (80K per account), and other regions have a cap of 10K per table with a 20K account cap.
* Provisioned Throughput Increase and Decrease − You can increase this as often as needed, but decreases remain limited to no more than four times daily per table.
* Table Size and Quantity Per Account − Table sizes have no limits, but accounts have a 256 table limit unless you request a higher cap.
* Secondary Indexes Per Table − Five local and five global are permitted.
* Projected Secondary Index Attributes Per Table − DynamoDB allows 20 attributes.
* Partition Key Length and Values − Their minimum length sits at 1 byte, and maximum at 2048 bytes, however, DynamoDB places no limit on values.
* Sort Key Length and Values − Its minimum length stands at 1 byte, and maximum at 1024 bytes, with no limit for values unless its table uses a local secondary index.
* Table and Secondary Index Names − Names must conform to a minimum of 3 characters in length, and a maximum of 255. They use the following characters: AZ, a-z, 0-9, “\_”, “-”, and “.”.
* Attribute Names − One character remains the minimum, and 64KB the maximum, with exceptions for keys and certain attributes.
* Reserved Words − DynamoDB does not prevent the use of reserved words as names.
* Expression Length − Expression strings have a 4KB limit. Attribute expressions have a 255-byte limit. Substitution variables of an expression have a 2MB limit.

⦁ **Primary Key**

The Primary Keys serve as the means of unique identification for table items, and secondary indexes provide query flexibility. DynamoDB streams record events by modifying the table data.

The Table Creation requires not only setting a name, but also the primary key; which identifies table items. No two items share a key. DynamoDB uses two types of primary keys −

⦁ **Partition Key** − This simple primary key consists of a single attribute referred to as the “partition key.” Internally, DynamoDB uses the key value as input for a hash function to determine storage.

⦁ Partition Key and Sort Key − This key, known as the “Composite Primary Key”, consists of two attributes.

⦁ The partition key and

⦁ The sort key.

⦁ **Secondary Indexes**

These indexes allow you to query table data with an alternate key. Though DynamoDB does not force their use, they optimize querying.

DynamoDB uses two types of secondary indexes −

⦁ **Global Secondary Index** − This index possesses partition and sort keys, which can differ from table keys.

⦁ **Local Secondary Index** − This index possesses a partition key identical to the table, however, its sort key differs.

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. This guide assumes a local install.

When ready for deployment, you can make a few small adjustments to your application to convert it to AWS use.

The install file is a .jar executable. It runs in Linux, Unix, Windows, and any other OS with Java support. Download the file by using one of the following links −

⦁ Tarball –

<http://dynamodb-local.s3-website-us-west2.amazonaws.com/dynamodb_local_latest.tar.gz>

⦁ Zip archive –

<http://dynamodb-local.s3-website-us-west2.amazonaws.com/dynamodb_local_latest.zip>

Note − Other repositories offer the file, but not necessarily the latest version. Use the links above for up-to-date install files. Also, ensure you have Java Runtime Engine (JRE) version 6.x or a newer version. DynamoDB cannot run with older versions.

After downloading the appropriate archive, extract its directory (DynamoDBLocal.jar) and place it in the desired location.

You can then start DynamoDB by opening a command prompt, navigating to the directory containing DynamoDBLocal.jar, and entering the following command −

java -Djava.library.path=./DynamoDBLocal\_lib -jar DynamoDBLocal.jar –sharedDb

**Installation**

* pip install aws in laptop.
* Go to aws console in browser.
* Go to Services.
* Click On IAM.
* Click on User(darshaws). [or any specified given by you]
* Note: If there isn’t any user you should create a user and groups
* Click on Security credentials(and scroll down)
* Create a new access id.
* you get your new access id & security id
* go to CMD prompt and enter " aws configure "

**NOTE**: if any error while running aws configure then .. pip uninstall aws configure .. and install it again. for future purpose pip install awscli ..

11. enter access key id,Secret Access Key and the Default region name,,(( All the details are available at step 8 and Step 9)

**Codes**:

**1: create a new Table**

import boto3

#enter the access key n other details

dynamodb = boto3.resource('dynamodb',aws\_access\_key\_id='xxxx', aws\_secret\_access\_key='yyyy', region\_name='zzzz')

table = dynamodb.create\_table (

TableName = 'Employees',

KeySchema = [

{

'AttributeName': 'Name',

'KeyType': 'HASH'

},

{

'AttributeName': 'Email',

'KeyType': 'RANGE'

}

],

AttributeDefinitions = [

{

'AttributeName': 'Name',

# 'S' indicates for String type.

'AttributeType': 'S'

},

{

'AttributeName':'Email',

'AttributeType': 'S'

}

],

ProvisionedThroughput={

'ReadCapacityUnits':1,

'WriteCapacityUnits':1

}

)

print(table)

2. **MySQL to DynamoDB**

* create a simple database in MYSQL with couples of data.
* Make sure you turn on xamp or another localhost.

Note: Make sure primary key of both table are of same name and type.

import mysql.connector as conn

import pandas as pd

from pandas.io import sql

import boto3

import awswrangler as wr

#enter the access key n other details

dynamodb = boto3.resource('dynamodb',aws\_access\_key\_id='xxxx', aws\_secret\_access\_key='yyyy', region\_name='zzzz')

#enter your database name in the database section

con = conn.connect(host='localhost', database='nb3', user='root' , passwd='')

#retrive some values from the table you created

query = "SELECT \* FROM [table-name] "

df = sql.read\_sql(query, con)

print(df)

def float\_to\_decimal(num):

return Decimal(str(num))

def pandas\_to\_dynamodb(df):

df = df.fillna(0)

# convert any floats to decimals

for i in df.columns:

datatype = df[i].dtype

if datatype == 'float64':

df[i] = df[i].apply(float\_to\_decimal)

# write to dynamodb

#enter the name you want to give for your table.

wr.dynamodb.put\_df(df=df, table\_name='Darsh')

pandas\_to\_dynamodb(df)

**3:: Providing own column and data in rows in already created table..**

import boto3

#enter the access key n other details

dynamodb = boto3.resource('dynamodb',aws\_access\_key\_id='xxxx', aws\_secret\_access\_key='yyyy', region\_name='zzzz')

#name of table already which already present in dynamoDB

table = dynamodb.Table('Employees')

response = table.put\_item(

#one primary key should be given

Item = {

'Name': 'Darsh',

'Email': 'xyz@.com'

}

)

print(response)

4. Updating Values Or Adding New columns with values

in a Table

import boto3

from boto3.dynamodb.conditions import Key

#enter the access key n other details

dynamodb = boto3.resource('dynamodb',aws\_access\_key\_id='xxxx', aws\_secret\_access\_key='yyyy', region\_name='zzzz')

table = dynamodb.Table('Darsh')

for n in range(3):

table.put\_item(Item={

#end a primary key with its values

'emp': 'fg5',

#enter new or existing column name and its value

'cc': '70'

})

5. Retrieve all the values of the table

import boto3

from boto3.dynamodb.conditions import Key

dynamodb = boto3.resource('dynamodb')

table = dynamodb.Table('Darsh')

#enter column names.

resp = table.scan(ProjectionExpression="emp, id")

print(resp['Items'])

**5: Batch write items in a Table**

import boto3

#enter the access key n other details

dynamodb = boto3.resource('dynamodb',aws\_access\_key\_id='xxxx', aws\_secret\_access\_key='yyyy', region\_name='zzzz')

table = dynamodb.Table('Employees')

with table.batch\_writer() as batch:

batch.put\_item(Item={"Name": "Luzze John", "Email": "john@handson.cloud",

"Department": "IT", "Section": { "QA": "QA-1", "Reporting Line": "L1" } })

batch.put\_item(Item={"Name": "Lugugo Joshua", "Email": "joshua@handson.cloud",

"Department": "IT", "Section": { "Development": "SD-1", "Reporting Line": "L1" } })

batch.put\_item(Item={"Name": "Robert Nsamba", "Email": "robert@handson.cloud",

"Department": "IT", "Section": { "PM": "PM-1", "Reporting Line": "L1" } })

print(batch)

Note : Amazon DynamoDB only supports certain data

1: Scalar – Number, String, Binary, Boolean, and Null.

2: Multi-valued – String Set, Number Set, and Binary Set.

3: Document – List and Map.

**Scalar** − These types represent a single value, and include number, string, binary, Boolean, and null.

**Document** − These types represent a complex structure possessing nested attributes, and include lists and maps.

**Set** − These types represent multiple scalars, and include string sets, number sets, and binary sets.

Issues:

1.Datetime format is not supported in dynmodb..conversion is needed to done before loading..

2. Cannot directly load any elements in the table, one primary key must be defined.

3. DynamoDb is not free basic usages are also charges.

follow this link for future purpose.

<https://highlandsolutions.com/blog/hands-on-examples-for-working-with-dynamodb-boto3-and-python>