**AWS CLI and DynamoDB**

Benjamin M. Brandhorst

University of Maryland Global Campus

SDEV 400 7980 Secure Programming in the Cloud

Doctor James Robertson

April 12th, 2020

**Create Table named Sensors & Add Records**

The Sensors table was created from the AWS Cloud9 CLI with the following command:

**aws dynamodb create-table --table-name Sensors --attribute-definitions AttributeName=Sensor,AttributeType=S --key-schema AttributeName=Sensor,KeyType=HASH --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5**

The successful table creation can be seen in Figure 1.A screenshot of a cell phone

Description automatically generated

Figure 1 – Illustrates the successful creation of the Sensors table

Following the creations of the Sensors table, the following code was used to fill the table with 20 records whose information was contained within a json:

**aws dynamodb batch-write-item --request-items file://sensorcontents.json --return-consumed-capacity TOTAL**

Figure 2 shows the results of executing this command.

A screenshot of a cell phone

Description automatically generated

Figure 2 – Batch writing items from a json into the Sensors table

The following command was used to display the contents of the Sensors table:

**aws dynamodb scan --table-name Sensors**

Figure 3 through Figure 7 show the contents of the Sensor table once the scan command is given.

A screenshot of a cell phone

Description automatically generated

Figure 3 – Scan table results

A screenshot of a cell phone

Description automatically generated

Figure 4 – Scan table results continued

A close up of text on a black background

Description automatically generated

Figure 5 – Scan table results continued

A close up of text on a black background

Description automatically generated

Figure 6 – Scan Table results continued

A close up of text on a black background

Description automatically generated

Figure 7 – Scan completed

**Create table named Courses & Create CLI to support searches**

Below is the source code used to create the Courses table, insert the records, and generate the interface from which users can search for items:

# File: Homework2.py

# Author: Ben Brandhorst

# Date: April 12th, 2020

# Purpose: SDEV400 Homework

from \_\_future\_\_ import print\_function # Python 2/3 compatibility

import boto3

import boto3

import json

import decimal

import time

from boto3.dynamodb.conditions import Key, Attr

def createTable():

# Creates the Courses table with CourseID as HASH key and a Global Secondary Index

dynamodb = boto3.resource('dynamodb')

table = dynamodb.create\_table(

TableName='Courses',

KeySchema=[

{

'AttributeName': 'CourseID',

'KeyType': 'HASH' #Partition key

}

],

AttributeDefinitions=[

{

'AttributeName': 'CourseID',

'AttributeType': 'N'

},

{

'AttributeName': 'Subject',

'AttributeType': 'S'

},

{ 'AttributeName': 'CatalogNbr',

'AttributeType': 'N'

},

],

ProvisionedThroughput={

'ReadCapacityUnits': 10,

'WriteCapacityUnits': 10

},

# Needed a GlobalSecondaryIndex to search for items without using the table key

GlobalSecondaryIndexes=[

{

'IndexName': 'topic\_index',

'KeySchema':[

{ #Required HASH type attribute

'AttributeName': 'CatalogNbr',

'KeyType': 'HASH',

},

{ 'AttributeName': 'Subject',

'KeyType': 'RANGE'

}

],

'Projection':{

'ProjectionType': 'ALL'

},

'ProvisionedThroughput': {

'ReadCapacityUnits': 10,

'WriteCapacityUnits': 10,

},

},

],

)

print("Table status:", table.table\_status)

def insertRecords():

# Inserts 10 items into the Courses table with Subject, CatalogNbr, Title,

# NumCredits, and CourseID as attributes

dynamodb = boto3.resource('dynamodb')

table = dynamodb.Table('Courses')

subject = ["GVPT", "SDEV", "SDEV", "BIOL", "WRTG", "SDEV", "SDEV", "SDEV", "CMSC", "SDEV"]

catalognbr = [170, 325, 350, 103, 393, 360, 425, 400, 495, 460]

title = ["American Government","Detecting Software Vulnerabilities", "Database Security",

"Introduction to Biology", "Advanced Technical Writing", "Secure Software Engineering",

"Mitigating Software Vulnerabilities", "Secure Programming in the Cloud",

"Current Trends and Projects in Computer Science", "Software Security Testing"]

numcredits = [3, 3, 3, 4, 3, 3, 3, 3, 3, 3,]

courseid = [80251, 80965, 83654, 83784, 21738, 22043, 24047, 24181, 50510, 53091]

length = len(subject)

i = 0

# Simple while loop to fill the table with the list contents from the variables

while i < length:

response = table.put\_item(

Item={

'Subject': subject[i],

'CatalogNbr': catalognbr[i],

'Title': title[i],

'NumCredits': numcredits[i],

'CourseID': courseid[i]

}

)

i += 1

print("Course items inserted")

def main():

# Loop to search for items in the Courses table

loop=True

while loop:

subject = input('Please enter a Subject ')

# Confirms user input is not empty

if not subject:

print('A subject is required')

continue;

try:

# We have to take the catalog input as an int in order to use it

# in a query. The except is necessary to confirm input is an int.

catalognbr = int(input('Please enter the catalog number '))

except ValueError:

print('A catalog number is required. Please try again.')

continue;

# Conduct the actual query using the command line input and printing the

# Title of the course searched for

dynamodb = boto3.resource('dynamodb')

table = dynamodb.Table('Courses')

response = table.query(

IndexName='topic\_index',

KeyConditionExpression=Key('Subject').eq(subject) & Key('CatalogNbr').eq(catalognbr)

)

for i in response['Items']:

print('The title of',subject,catalognbr,'is',i['Title'])

answer = input('Would you like to conduct another search? Input Yes or No ')

# Loops back to the beginning of the function if any input other than

# No is detected.

if answer==('No'):

print('Thank you for using this program. Goodbye')

loop=False

else:

continue;

createTable()

# A delay is necessary to allow time for table creation before content is inserted

time.sleep(20)

insertRecords()

main()

Figure 8 shows the results of running the program and the menu selection options being utilized. Of note, this program does not confirm that the table items exist before attempting to print the course title. Therefore, if user input does not match a table item, the program will not return any results and instead simply asks users if they would like to conduct another search. This is illustrated in Figure 9.

A screenshot of a computer

Description automatically generated

Figure 8 – Command Line Interface interactions

A screenshot of a cell phone

Description automatically generated

Figure 9 – Demonstrating user searching for a record that does not exist

**Use AWS CLI to delete DynamoDB tables**

The following commands were used to delete the Sensors and Courses tables:

**aws dynamodb delete-table \**

**--table-name Sensors**

**aws dynamodb delete-table \**

**--table-name Courses**

The results of running these commands can be seen in Figure 10 and Figure 11.

A screenshot of a cell phone

Description automatically generated

Figure 10 – Deleting the Sensors table

A screenshot of a cell phone

Description automatically generated

Figure 11 – Deleting the Courses table

References

No outside code other than the provided Python snippets, examples in the UsingDynamoDB.pdf, and examples found on the AWS DynamoDB support site were used in the completion of this assignment.