BASES DE DATOS

Clase 20 - Bases de datos key-value

Profesora: Erika Gutiérrez Beltrán

Tema 3: Bases de datos NoSQL

Objetivo

Construir bases de datos para datos no estructurados, comprender el funcionamiento y aplicación de estos tipos de bases de datos

Temario

- Arquitecturas de bases de datos manejadoras de grandes volúmenes de datos ✅
- Bases de datos bajo modelos Key - Value ✅
- Bases de datos orientadas a documentos
- Bases de datos columnares
- Bases de datos orientadas a grafos

Logros

Interactuar con grandes volúmenes de datos, realizar consultas e interactuar con los datos

¿Qué es Dynamo?

Para recordar:

- Dynamo es un servicio para el almacenamiento de datos NoSQL
- Facilita a los usuarios resultados a altas velocidades
- Se puede escalar de forma rápida y sencilla
- Proporciona operaciones automáticas
- Posibilidad de hacerla una base de datos distribuida
- Alto rendimiento
- Cada tabla se puede escalar
- Creación de copias de seguridad bajo demanda
- Acceso por api o por consola de comandos
- Todas las tablas son colecciones de ítems
- Los item son colecciones de atributos o llaves

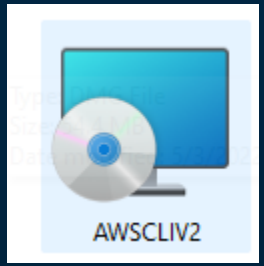
Práctica con Dynamo

Para la práctica con Dynamo, todos utilizaremos el mismo usuario, es importante seguir detalladamente las instrucciones que aquí se describen:

1.Ir al enlace que se presenta a continuación y descargar una de las versiones de Dynamo para su computador (Región del oeste de EE.UU)
https://docs.aws.amazon.com/es_es/amazondynamodb/latest/developerguide/DynamoDBLocal.DownloadingAndRunning.html#DynamoDBLocal.DownloadingAndRunning.title
(Enlace de descarga para los que tiene Windows: https://s3.us-west-2.amazonaws.com/dynamodb-local/dynamodb_local_latest.zip)

2. Cuando se descargue el archivo, descomprimir para obtener la carpeta que será utilizada más adelante

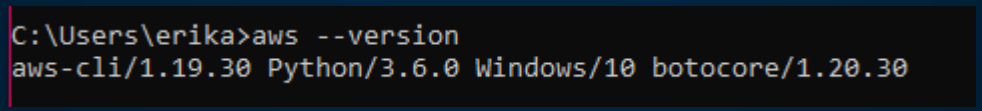
3. Ahora descargue la consola de AWS, esta la utilizaremos para ejecutar comandos, ir al enlace: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html



Práctica con Dynamo

4. Ejecute el archivo descargado y espere a que se complete la instalación

5. Ahora, ejecuta el comando en tu cmd o terminal *aws –versión* y deberá visualizarse con la siguiente imagen:



Práctica con Dynamo

6. Vamos a configurar nuestro ambiente de manera local, la idea es que al final de la clase migremos lo que hemos construido a la base de datos remota. Ejecuta en la consola AWS configure

AWS Access Key ID: foo
AWS Secret Access Key: bar
Default región name: us-east-2
Default output format: json

```
C:\Users\erika>aws configure
AWS Access Key ID [****************FCBV]: foo
AWS Secret Access Key [****************d+fk]: bar
Default region name [us-east-2]:
Default output format [json]:
```

Práctica con Dynamo

7. Crea el servicio que te permitirá interactuar con la base de datos local de Dynamo por medio del siguiente comando:

java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb -port 3001
aws dynamodb list-tables --endpoint-url http://localhost:8001

```
C:\Users\erika\Downloads\dynamodb_local_latest>java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -shar
edDb -port 8001
Initializing DynamoDB Local with the following configuration:
Port:   8001
InMemory:       false
DbPath: null
SharedDb:       true
shouldDelayTransientStatuses:   false
CorsParams:     *
```

Práctica con Dynamo

8. Crea el servicio que te permitirá interactuar con la base de datos local de Dynamo por medio del siguiente comando:

java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb -port 3001
aws dynamodb list-tables --endpoint-url http://localhost:8001

```
C:\Users\erika\Downloads\dynamodb_local_latest>java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -shar
edDb -port 8001
Initializing DynamoDB Local with the following configuration:
Port:   8001
InMemory:       false
DbPath: null
SharedDb:       true
shouldDelayTransientStatuses:   false
CorsParams:     *
```

Práctica con Dynamo

Antes de iniciar con la creación de la tabla que permitirá almacenar información, imagina que previamente diseñaste un sistema de manejo de clientes que compran en una tienda, pero la empresa quiere migrar a un modelo NoSQL porque migraran toda su infraestructura a la nube buscando bajar costos, mitigar riesgos de indisponibilidad, permitir despliegue continuo y además de aumentar de manera elástica cuando hay fechas especiales donde incrementan las ventas.

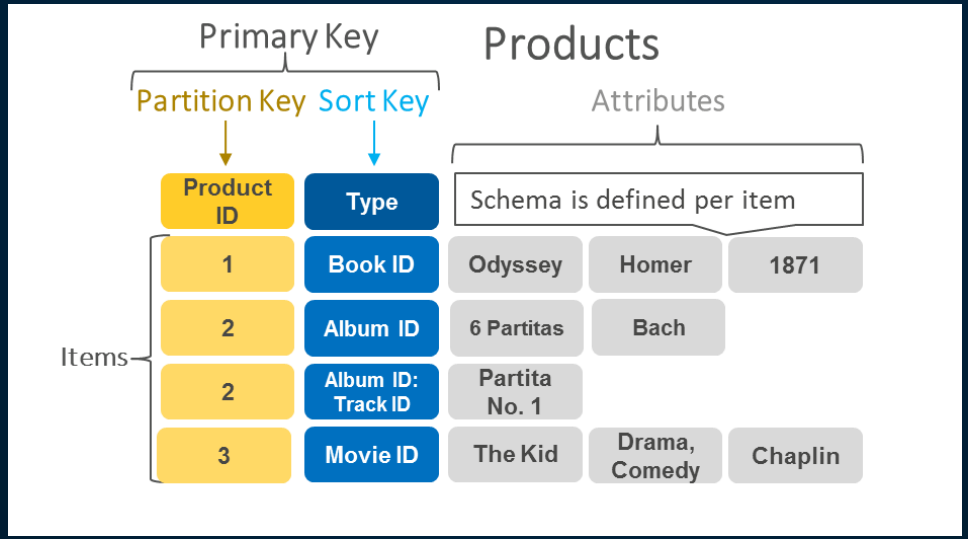
Práctica con Dynamo

Imagina en el modelo relacional te han pedido consultar la entidad Customers, el cual tiene customerid, firstName, LastName. Esta entidad tiene una relación con Addresses, la cual tiene como atributos AddressId, CustomerId, Nickname, StreetAddress, PostalCode, Country.

Crea el diagrama relacional entre estas dos entidades con 5 registros en Customers y 10 en Addresses. Puedes usar Excel para esto.

Práctica con Dynamo

Ahora intenta pensar en como representarías las tablas que acabas de crear usando esta estructura



Práctica con Dynamo

DynamoDB funciona distinto ya que en este tipo de base de datos no hay uniones o relaciones, es necesario construir una manera de unir los datos, aunque consideremos que son de entidades o tablas diferentes. En DynamoDB es correcto pensar en la desnormalización.

Primary key		Attributes		
Partition key: PK	Sort key: SK			
CUSTOMER #alexdebrie	CUSTOMER #alexdebrie	FirstName	LastName	MailingAddresses
		Alex	DeBrie	{ "Home": { "StreetAddress: "1122 1st Avenue", "PostalCode": "90210", "Country": "UnitedStates" }, "Business": { "StreetAddress: "555 Broadway", "PostalCode": "90211", "Country": "UnitedStates" } }
CUSTOMER #jeffbezos	CUSTOMER #jeffbezos	FirstName	LastName	MailingAddresses
		Jeff	Bezos	{ "Home": { "StreetAddress: "123 Spruce Drive", "PostalCode": "69361", "Country": "UnitedStates" } }

Práctica con Dynamo

Ahora, intenta pasar las relaciones construidas previamente en el modelo de base de datos del sitio web de películas, a un esquema NoSQL con DynamoDB. Enfoquémonos en la relación que tiene película favorita, película y usuario.

Práctica con Dynamo

9. Crea el archivo de configuración para la creación de tu tabla en la base de datos de Dynamo

```
{
    "TableName": “Nombre de la tabla a crear",
    "KeySchema": [
        {
            "AttributeName": “nombre del atributo que te servirá como identificador",
            "KeyType": "HASH"
        }
    ],
    "AttributeDefinitions": [
        {
            "AttributeName": “nombre del atributo identificador",
            "AttributeType": “tipo del identificador S-String, N-Number"
        }
    ],
    "ProvisionedThroughput": {
        "ReadCapacityUnits": unidad de solicitud de lectura,
        "WriteCapacityUnits": unidad de escritura de un elemento
    }
}
```

Práctica con Dynamo

10. Ejecuta el script creation.json que acabas de crear, ejecutando el siguiente comando en la consola: *aws dynamodb create-table --cli-input-json file:///Users/erika/OneDrive/Desktop/Clases-Eafit/Bases-Datos-Practica/dynamo/config/ creation.json --endpoint-url http://localhost:8001*

11. Valida las tablas existentes usando el siguiente comando: *aws dynamodb list-tables --endpoint-url http://localhost:8001*

```
C:\Users\erika>aws dynamodb list-tables --endpoint-url http://localhost:8001
{
    "TableNames": [
        "DB_Dynamo",
        "Movies",
        "table_course_DB"
    ]
}
```

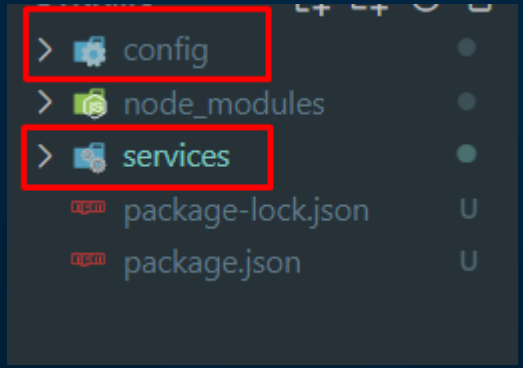
Práctica con Dynamo

12. Crea una carpeta nueva (donde tienes el repositorio descargado, se debe subir la práctica al repositorio) llamada dynamo

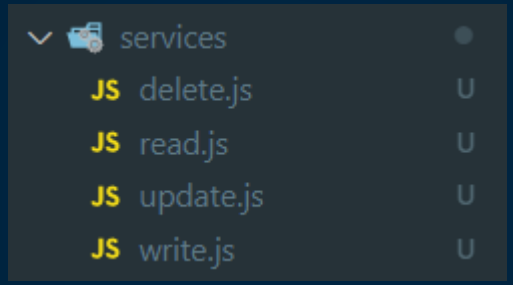
13. Abre la carpeta con tu editor de preferencia, puede ser visual studio code. Luego abre una consola en la ubicación de la carpeta y ejecuta el comando *npm init*

14. Al finalizar la ejecución del comando escribe en consola *npm install aws-sdk –save,* espera a que finalice la instalación

15. Crea dos carpetas en la raíz del proyecto config y services



16. En la carpeta services crea los archivos delete.js, read.js, update.js, create.js (Crearemos un crud)



17. En la carpeta config crea el archivo config.js

Práctica con Dynamo

18. Adjunta el siguiente código en config.js

```
module.exports = {
    table_course: 'DB-Course',
    aws_local_configu: { //Se usa esta configuración ya que tenemos nuestra base de datos de manera
local
        region: 'local',
        endpoint: 'http://localhost:8001'
    },
    aws_remote_config:{ // Se usa esta configuración en el momento que nos conectamos a la base de
datos en AWS de manera remota
        accessKeyId: ‘tomar de interactiva',
        secretAccessKey: ‘tomar de interactiva',
        region: 'us-east-2',
    }
}
```

Práctica con Dynamo

19. En el archivo create.js copiar el siguiente código

```
var AWS = require("aws-sdk") //Se importa librería
const config = require("../config/config.js") //Se llama archivo creado previamente de configuración
AWS.config.update(config.aws_local_config) //Se define en AWS la configuración local

const doClient = new AWS.DynamoDB.DocumentClient() //Se crea el cliente de AWS
const saveStudents = function(){ //Se define la función encargada de crear los datos en la table de Dynamo
    var object = { //Crear objeto para almacenar en la base de datos
        "id": "111",
        "name": "Pepe Gutierrez",
        "email": "ejgutierrb@eafit.edu.co",
        "cellphone": "0000000000",
        "created_at": new Date().toString()
    }
    const params = {
        TableName: config.table_course, //Nombre de la table donde se almacenará la información
        Item: object //Datos a insertar
    }
    doClient.put(params, function(err, data) { //Se llama función del objeto encargado de crear información
        if(err)
            console.log("Error: ", err)
        else
            console.log("Datos almacenados correctamente ", data)
    })
}
saveStudents()
```

Práctica con Dynamo

20. Código para cargar datos de la base de datos (read.js)

```
var AWS = require("aws-sdk") //Se importa librería
const config = require("../config/config.js") //Se llama archivo creado previamente de configuración
AWS.config.update(config.aws_local_configu) //Se define en AWS la configuración local

const doClient = new AWS.DynamoDB.DocumentClient() //Se crea el cliente de AWS
const readStudents = function(){
    const params = {
        TableName: config.table_course,
        Key: {"id": "15" }
    }
    doClient.get(params, function(err, data) { //Se cambia función para obtener
        if(err)
            console.log("Error: ", err)
        else
            console.log("Datos: ", data)
    })
}
readStudents()
```

Práctica con Dynamo

21. Código para actualizar datos (update.js)

```
var AWS = require("aws-sdk") //Se importa librería
const config = require("../config/config.js") //Se llama archivo creado previamente de configuración
AWS.config.update(config.aws_local_configu) //Se define en AWS la configuración local

const doClient = new AWS.DynamoDB.DocumentClient() //Se crea el cliente de AWS
const readStudents = function(){

    const params = {
        TableName: config.table_course,
        Key: {"id": "15" },
        UpdateExpression: "set #name= :newNameAttri",
        ExpressionAttributeValues: {
            ":newNameAttri": "Jissel"
        },
        ExpressionAttributeNames: {
            "#name": "name"
        },
        ReturnValues: "UPDATED_NEW"
    }
    doClient.update(params, function(err, data) {
        if(err)
            console.log("Error: ", err)
        else
            console.log("Datos actualizados correctamente ", data)
    })
}
readStudents()
```

Práctica con Dynamo

22. Código para eliminar datos (delete.js)

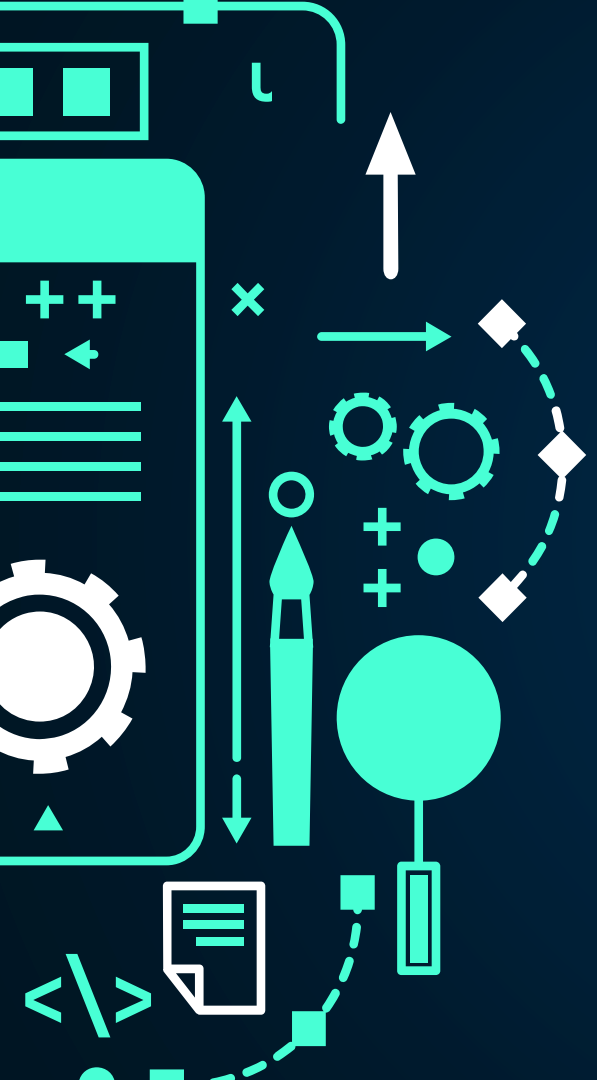
```
var AWS = require("aws-sdk") //Se importa librería
const config = require("../config/config.js") //Se llama archivo creado previamente de configuración
AWS.config.update(config.aws_local_configu) //Se define en AWS la configuración local

const doClient = new AWS.DynamoDB.DocumentClient() //Se crea el cliente de AWS
const saveStudents = function(){
    const params = {
        TableName: config.table_course,
        Key: {"id": "111" }
    }

    doClient.delete(params, function(err, data) {
        if(err)
            console.log("Error: ", err)
        else
            console.log("Datos eliminados correctamente ", data)
    })
}
saveStudents()
```

REFERENCIAS

- Dynamo: https://aws.amazon.com/dynamodb/
- acenswhitepapers: https://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf
- ¿Qué es DynamoDB AWS?: https://docs.aws.amazon.com/es_es/amazondynamodb/latest/developerguide/Introduction.html
- Componentes básicos Dynamo: https://docs.aws.amazon.com/es_es/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html
- Dynamodb-one-to-many: https://www.alexdebrie.com/posts/dynamodb-one-to-many/
- CLI AWS: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
- DynamoDB Local: https://s3.us-west-2.amazonaws.com/dynamodb-local/dynamodb_local_latest.zip



Gracias!