

## AMAZON DYNAMODB: CREACIÓN DE TABLAS Y OPERACIONES DE MANIPULACIÓN DE DATOS

Amazon DynamoDB es un servicio de base de datos NoSQL completamente administrado y escalable que ofrece un alto rendimiento, durabilidad y flexibilidad. Es una herramienta esencial en el arsenal de un desarrollador para gestionar datos en aplicaciones que requieren una rápida respuesta y una escalabilidad sencilla.

El objetivo principal de esta práctica es familiarizarse con Amazon DynamoDB y aprender a realizar las siguientes tareas:

- Crear una tabla DynamoDB: Aprenderemos a diseñar una tabla que se adapte a nuestras necesidades y a configurarla en el entorno de DynamoDB.
- Inserción y consulta de datos: Aprenderemos cómo agregar y recuperar datos en nuestra tabla DynamoDB, utilizando las operaciones de inserción y consulta.
- Actualización y eliminación de datos: Exploraremos cómo actualizar y eliminar registros en una tabla DynamoDB de manera eficiente.
- Exploración de índices: Investigaremos cómo usar índices para optimizar las consultas y acceder a datos de manera más eficiente.

### Requerimientos:

- Disponer de acceso a los recursos de AWS a través de un *sandbox* de AWS Academy
- Disponer de una máquina o instancia EC2 con acceso a Internet desde la que se ejecutará el cliente de la base de datos

### Arquitectura propuesta:



### Realización:

#### CONFIGURACIÓN DEL ENTORNO CLIENTE

- 1) Para comenzar esta práctica, previamente es necesario tener instalada y configurada la interfaz de línea de comandos de AWS (AWS CLI v2). Para ello, dependiendo del sistema operativo de la máquina desde donde se vaya a lanzar el cliente, se realizará la tarea indicada en el siguiente enlace:

[https://docs.aws.amazon.com/es\\_es/cli/latest/userguide/getting-started-install.html](https://docs.aws.amazon.com/es_es/cli/latest/userguide/getting-started-install.html)

- 2) A continuación, se debe configurar la AWS CLI con las credenciales del AWS Academy Learner Lab. Para ello, desde la consola del laboratorio, seleccionamos la opción **AWS Details** y a continuación presionamos el botón **Show** en el apartado correspondiente a la AWS CLI:

The screenshot shows the AWS Academy interface. On the left is a terminal window with the prompt `eee_N_2513646@runeb978741:~$`. On the right is a 'Cloud Access' sidebar. Under 'AWS CLI:', it says 'Copy and paste the following into ~/.aws/credentials' and provides a block of credentials:

```
[default]
aws_access_key_id=ASIARPVFCJIDUH5ZMJ7K
aws_secret_access_key=MS6c4EuZsfkkRGquw0h4XxALI7G81oJBMRScq7nt
aws_session_token=FwoGZXIvYXZlEPn////////wEaDPr2yG7bfun0tImGBCK9AeQgnzqtZ+v
Giw+kc3aPMq2Xm8EYoyXPH7MnIQM0S8UvcIA6pe/iPtQv0H72dzsMXT+lTJav1Xcf70R0DWecwDnHkMLq0kIaLFUTE+Wu
49b+wNDkgQcB56U6i73Aqa4SgFXT6k5WTY0f22NNY0k419rHnhMDzKrl0JtwDaU/CYsGJC8RUExzG34iraQPDUIUFfe04+xp
TvQ5JbRRu5wjGLrDF924bouXgu6t2FB8BzU3h2kuFDVl+2sja1mC iTm90pBjItjhZYVH6f/2+apHoVBmA0zrds iPzRacpU
w8+0kYhS05Q5CIg55eoREJCqTzDG"
```

Copiamos las credenciales indicadas en el recuadro.

- 3) Desde la máquina donde se ha instalado la AWS CLI, se abre una línea de comandos y se introduce la orden:

```
$ aws configure
```

Se introducen los siguientes valores que se solicitan:

- **AWS Access Key ID:** Copiar y pegar el indicado en el recuadro
- **AWS Secret Access Key:** Copiar y pegar el indicado en el recuadro
- **Default region name:** `us-east-1`
- **Default output format:** `json`

```
root@aws:~# aws configure
AWS Access Key ID [None]: ASIARPVFCJIDUH5ZMJ7K
AWS Secret Access Key [None]: MS6c4EuZsfkkRGquw0h4XxALI7G81oJBMRScq7nt
Default region name [None]: us-east-1
Default output format [None]: json
root@aws:~#
```

- 4) Por último, falta añadir al archivo de configuración una línea con el token de sesión (hay que copiarlo de las credenciales del AWS Academy Learner Lab). Para ello ejecutamos la orden siguiente, sustituyendo el *placeholder* por el valor correspondiente:

```
echo "aws_session_token=<token_sesion>" >> .aws/credentials
```

```
root@aws:~# echo "aws_session_token=FwoGZXIvYXZlEPn////////wEaDPr2yG7bfun0tImGBCK9AeQgnzqtZ+v
Giw+kc3aPMq2Xm8EYoyXPH7MnIQM0S8UvcIA6pe/iPtQv0H72dzsMXT+lTJav1Xcf70R0DWecwDnHkMLq0kIaLFUTE+Wu
49b+wNDkgQcB56U6i73Aqa4SgFXT6k5WTY0f22NNY0k419rHnhMDzKrl0JtwDaU/CYsGJC8RUExzG34iraQPDUIUFfe04+xp
TvQ5JbRRu5wjGLrDF924bouXgu6t2FB8BzU3h2kuFDVl+2sja1mC iTm90pBjItjhZYVH6f/2+apHoVBmA0zrds iPzRacpU
w8+0kYhS05Q5CIg55eoREJCqTzDG" >> .aws/credentials
```

- 5) Una vez configuradas las credenciales, procederemos a instalar la herramienta **NoSQL Workbench**, que es una herramienta para escritorio que permite operar gráficamente con bases de datos en Amazon DynamoDB y Amazon Keyspaces. Para ello, seguiremos las indicaciones del siguiente enlace, desde donde se puede descargar e instalar el software de manera gratuita:

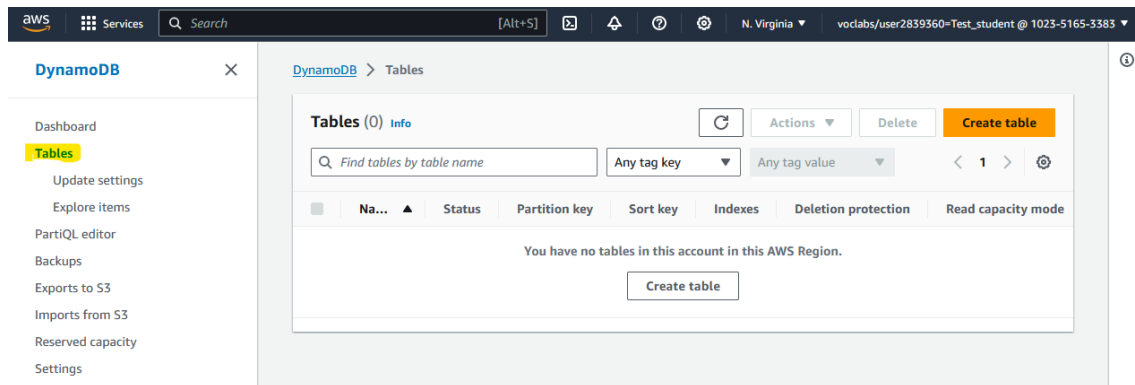
<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/workbench.settingup.html>

## CREACIÓN DE LA TABLA DE AMAZON DYNAMODB

Previamente a la creación de la tabla, clonaremos el siguiente repositorio donde se encuentran los recursos necesarios para esta práctica:

```
git clone https://github.com/jose-emilio/aws-academy-fp-dam.git
```

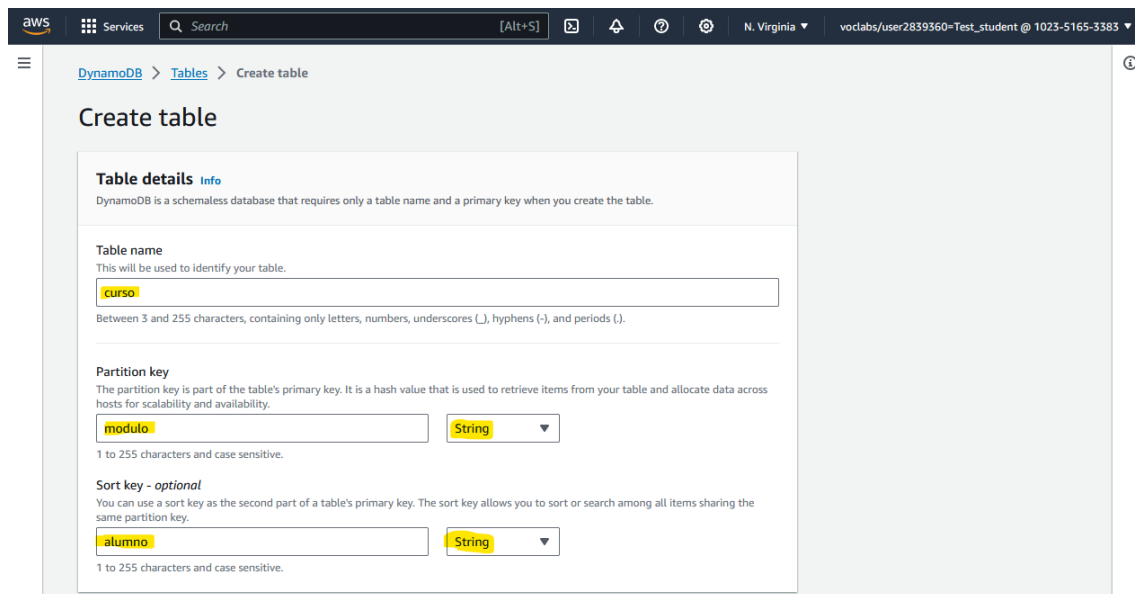
- 6) Para crear la tabla de Amazon DynamoDB de esta práctica, accederemos a la consola de Amazon DynamoDB y, desde la opción **Tables** del menú lateral, presionamos el botón **Create table**:



- 7) (Mediante la consola de Administración de AWS) En el asistente que aparece a continuación, introduciremos el nombre de la tabla y los atributos que componen la clave primaria y sus correspondientes descriptores de tipos de datos, tal y como se indica a continuación:

- **Table name:** *curso*
- **Partition key:** *modulo (String)*
- **Sort key:** *alumno (String)*

El resto de las opciones las dejamos en sus valores por defecto y presionamos el botón **Create table**



(Mediante la **AWS CLI**) La operación anterior, también puede realizarse programáticamente mediante la AWS CLI o un AWS SDK correspondiente. Para ello, accedemos al repositorio, en el directorio donde se encuentran los recursos de esta práctica:

```
cd aws-academy-fp-dam/resources/dynamodb
```

Y por último creamos la tabla con la AWS CLI:

```
aws dynamodb create-table --table-name curso --key-schema file://tabla/esquema-claves.json
--attribute-definitions file://tabla/atributos.json --provisioned-throughput
file://tabla/capacidad.json
```

Tras el proceso, se devolverá un documento JSON que nos indicará que la tabla se está creando con los parámetros indicados:

```
root@aws:~/aws-academy-fp-daw/resources/dynamodb# aws dynamodb create-table --table-name curso --key-schema file://tabla/esquema-claves.json --attribute-definitions file://tabla/atributos.json --provisioned-throughput file://tabla/capacidad.json
{"TableDescription": {"AttributeDefinitions": [{"AttributeName": "alumno", "AttributeType": "S"}, {"AttributeName": "modulo", "AttributeType": "S"}], "TableName": "curso", "KeySchema": [{"AttributeName": "modulo", "KeyType": "HASH"}, {"AttributeName": "alumno", "KeyType": "RANGE"}], "TableStatus": "CREATING", "CreationDateTime": "2023-10-22T08:46:27.246000+00:00", "ProvisionedThroughput": {"NumberOfDecreasesToday": 0, "ReadCapacityUnits": 5, "WriteCapacityUnits": 5}, "TableSizeBytes": 0, "ItemCount": 0, "TableArn": "arn:aws:dynamodb:us-east-1:102351653383:table/curso", "TableId": "04975d72-c549-4080-896c-5dec8521a376", "DeletionProtectionEnabled": false}}
```

- 8) Tras el proceso de creación, podremos comprobar en la Consola de Administración del servicio Amazon DynamoDB que nuestra tabla se ha creado correctamente:

| Name  | Status | Partition key | Sort key   | Indexes | Deletion protection | Read capacity mode | Write capacity mode | Total size | Table class |
|-------|--------|---------------|------------|---------|---------------------|--------------------|---------------------|------------|-------------|
| curso | Active | modulo (S)    | alumno (S) | 0       | Off                 | Provisioned (5)    | Provisioned (5)     | 0 bytes    | Standard    |

- 9) Por último, vamos a poblar nuestra tabla de Amazon DynamoDB. Para ello ejecutamos el *script* que realizará tal función:

```
./cargar-datos.sh
```

Tras esta operación, podremos volver a la Consola de Administración del servicio Amazon DynamoDB y, entrando en la tabla *curso* podremos acceder al enlace del menú lateral **Explore items**, seleccionar la tabla *curso*. Aparecerán todos los elementos añadidos por el *script* lanzado:

|                          | modulo (String)    | alumno (String)  | convocatoria | curso | nota             | observaciones |
|--------------------------|--------------------|------------------|--------------|-------|------------------|---------------|
| <input type="checkbox"/> | <a href="#">ED</a> | Ana Suarez       | 3            | 2022  | [[{"N": "1" ...  |               |
| <input type="checkbox"/> | <a href="#">ED</a> | Andres Lledo     | 1            | 2023  | [[{"N": "5" ...  |               |
| <input type="checkbox"/> | <a href="#">ED</a> | Antonio Aparicio | 1            | 2023  | [[{"N": "8" ...  |               |
| <input type="checkbox"/> | <a href="#">ED</a> | Belen Perello    | 1            | 2023  | [[{"N": "6" ...  |               |
| <input type="checkbox"/> | <a href="#">ED</a> | Juan Martinez    | 2            | 2023  | [[{"N": "3.5..." |               |
| <input type="checkbox"/> | <a href="#">ED</a> | Lujan Torres     | 1            | 2022  | [[{"N": "7.5..." |               |
| <input type="checkbox"/> | <a href="#">ED</a> | Simon Miranda    | 2            | 2022  | [[{"N": "4" ...  |               |

La tabla creada contiene información simulada de calificaciones de alumnos en diferentes módulos del ciclo de “DAM” o “DAW”, indicando para cada alumno matriculado en un módulo, la convocatoria actual, el curso y una lista de calificaciones que ha obtenido en dichas convocatorias. La lista de calificaciones tiene una longitud igual al valor indicado en el atributo *convocatoria*, de forma que si un alumno ha utilizado más de una convocatoria es porque sus calificaciones en las convocatorias anteriores a la actual han sido inferiores a 5.

## MANIPULACIÓN DE AMAZON DYNAMODB

Amazon DynamoDB dispone de las siguientes operaciones para manipular los datos de una tabla:

- **GetItem.** Permite recuperar un único elemento en una tabla de Amazon DynamoDB, identificado por los atributos clave.
- **PutItem.** Permite introducir un único elemento en una tabla de Amazon DynamoDB, identificado por los atributos clave, o su sobrescritura si ya existía.
- **DeleteItem.** Permite eliminar un único elemento de una tabla de Amazon DynamoDB, identificado por los atributos clave.
- **UpdateItem.** Permite actualizar un único elemento de una tabla de Amazon DynamoDB, identificado por los atributos clave.
- **Query.** Permite recuperar un conjunto de elementos que tengan el mismo valor en su campo clave de partición (o clave *hash*) de una tabla de Amazon DynamoDB
- **Scan.** Permite recuperar un conjunto de elementos de todas las particiones de una tabla de Amazon DynamoDB

En este apartado de la práctica se verán las diferentes operaciones de manipulación de datos utilizando tanto la AWS CLI como la herramienta NoSQL Workbench.

Para ello, supondremos que deseamos realizar las siguientes operaciones:

- Operación 1: Obtener las calificaciones y la convocatoria del alumno *Antonio Aparicio* en el módulo de programación (*PRG*)
- Operación 2: Introducir un nuevo elemento en la tabla para un alumno llamado Miguel Domínguez, que cursa el módulo de Bases de Datos (*BD*) en el curso 2023 y que en la convocatoria 1 ha obtenido una nota de 8.5

- Operación 3: Eliminar el elemento correspondiente al alumno de nombre *Andres Lledo* que cursa el módulo de Bases de Datos (*BD*)
- Operación 4: Actualizar el elemento correspondiente de la tabla, asumiendo que la alumna *Ana Suarez* acaba de suspender una nueva convocatoria del módulo de Lenguajes de Marcas y Sistemas de Gestión de la Información (*LMSI*) con una nota de 3
- Operación 5: Obtener las notas y nombres de los alumnos que cursen (o hayan cursado) el módulo de Entornos de Desarrollo (*ED*) en el curso 2023 y lo hayan aprobado en la primera convocatoria.
- Operación 6: Obtener los nombres de los módulos y calificaciones de todos los que ha cursado la alumna *Sofia Diaz*

### Manipulación de datos mediante la AWS CLI

- 10) Operación 1: Para introducir el elemento correspondiente al alumno, es necesario indicar en un documento JSON la información con los datos de la clave del alumno que se desea obtener. Esta información se encuentra ya creada en la ruta *operaciones/get-item/clave.json*:

```
{
  "modulo": { "S": "PRG" },
  "alumno": { "S": "Antonio Aparicio" }
}
```

Para ejecutar la consulta desde la AWS CLI introducimos el siguiente comando:

```
aws dynamodb get-item --table-name curso --key file://operaciones/get-item/clave.json --projection-expression "convocatoria, nota"
```

En el comando anterior, se ejecuta una orden **GetItem**, indicándose los siguientes parámetros:

- **--table-name**: Nombre de nuestra tabla, en este caso *curso*
- **--key**: Fichero donde se encuentra el documento JSON con la información de los atributos clave del elemento que se obtendrá
- **--projection-expression**: Indica los atributos del elemento obtenido que se proyectarán en el resultado de la consulta

Como resultado, se obtendrá un documento JSON con los datos solicitados:

```
root@aws:~/aws-academy-fp-daw/resources/dynamodb# aws dynamodb get-item --table-name curso --key file://operaciones/get-item/clave.json --projection-expression "convocatoria, nota"
{
  "Item": {
    "convocatoria": {
      "N": "2"
    },
    "nota": {
      "L": [
        {
          "N": "4.5"
        },
        {
          "N": "7"
        }
      ]
    }
  }
}
root@aws:~/aws-academy-fp-daw/resources/dynamodb#
```

- 11) Operación 2: Para introducir un nuevo elemento en la tabla *curso*, es necesario indicar en un documento JSON la información con todos los atributos del elemento que se desea introducir. Esta información se encuentra ya creada en la ruta *operaciones/put-item/item.json*:

```
{
  "modulo": { "S": "BD" },
  "alumno": { "S": "Miguel Dominguez" },
  "curso": { "N": "2023" },
  "nota": { "L": [ { "N": "8.5" } ] },
  "convocatoria": { "N": "1" }
}
```

}

Para ejecutar la operación de inserción desde la AWS CLI introducimos el siguiente comando:

```
aws dynamodb put-item --table-name curso --item file://operaciones/put-item/item.json
```

En el comando anterior, se ejecuta una orden **PutItem**, indicándose los siguientes parámetros:

- **--table-name:** Nombre de nuestra tabla, en este caso *curso*
- **--item:** Fichero donde se encuentra el documento JSON con la información de todos los atributos del elemento que se introducirá

Como resultado, la instrucción anterior no devolverá nada por la salida. Para verificar que el elemento se ha introducido correctamente podemos visualizarlo en la Consola de Administración del servicio de Amazon DynamoDB:

| modulo (String) | alumno (String)  | convocatoria | curso | nota              | observaciones |
|-----------------|------------------|--------------|-------|-------------------|---------------|
| BD              | Lujan Torres     | 1            | 2022  | [{"N": "7.5..."}] |               |
| LMSI            | Lujan Torres     | 2            | 2022  | [{"N": "4" ...}]  |               |
| BD              | Miguel Dominguez | 1            | 2023  | [{"N": "8.5..."}] |               |
| ED              | Simon Miranda    | 2            | 2022  | [{"N": "4" ...}]  |               |
| PRG             | Simon Miranda    | 3            | 2022  | [{"N": "4" ...}]  |               |

- 12) **Operación 3:** Para eliminar el elemento correspondiente al alumno, es necesario indicar en un documento JSON la información con los datos de la clave del alumno. Esta información se encuentra ya creada en la ruta *operaciones/delete-item/clave.json*:

```
{
  "modulo": { "S": "BD" },
  "alumno": { "S": "Andres Lledo" }
}
```

Para ejecutar la operación de eliminación desde la AWS CLI introducimos el siguiente comando:

```
aws dynamodb delete-item --table-name curso --key file://operaciones/delete-item/clave.json --return-values ALL_OLD
```

En el comando anterior, se ejecuta una orden **DeleteItem**, indicándose los siguientes parámetros:

- **--table-name:** Nombre de nuestra tabla, en este caso *curso*
- **--key:** Fichero donde se encuentra el documento JSON con la información de los atributos clave del elemento que se obtendrá
- **--return-values:** Es un parámetro opcional que, en este caso, se utiliza para indicar al servicio Amazon DynamoDB que retorne los valores de los atributos que tenía el elemento antes de ser eliminado

Como resultado, se obtendrá un documento JSON con los datos del elemento que se acaba de eliminar de la tabla:

```
root@aws:~/aws-academy-fp-daw/resources/dynamodb# aws dynamodb delete-item --table-name curso --key file://operaciones/delete-item/clave.json --return-values ALL_OLD
{
  "Attributes": {
    "alumno": {
      "S": "Andres Lledo"
    },
    "curso": {
      "N": "2023"
    },
    "convocatoria": {
      "N": "3"
    },
    "modulo": {
      "S": "BD"
    },
    "nota": {
      "L": [
        {
          "N": "4"
        },
        {
          "N": "3"
        },
        {
          "N": "5"
        }
      ]
    }
  }
}
root@aws:~/aws-academy-fp-daw/resources/dynamodb#
```

- 13) **Operación 4:** Para actualizar el elemento solicitado al alumno, es necesario indicar en un documento JSON la información con los datos de la clave del alumno que se modifica. Esta información se encuentra ya creada en la ruta *operaciones/delete-item/clave.json*:

```
{
  "modulo": { "S": "LMSI" },
  "alumno": { "S": "Ana Suarez" }
}
```

Además, es necesario indicar la información que se va a modificar en otro documento JSON. En este caso, se incrementará la convocatoria y se añadirá una nueva nota a la lista de calificaciones. Esta información se encuentra ya creada en la ruta *operaciones/update-item/valores.json*:

```
{
  ":inc" : { "N" : "1" },
  ":n" : { "L" : [ { "N": "3" } ] }
}
```

Para realizar la operación de actualización desde la AWS CLI, se ejecuta el siguiente comando:

```
aws dynamodb update-item --table-name curso --key file://operaciones/update-item/clave.json --update-expression "SET convocatoria = convocatoria + :inc, nota = list_append(nota, :n)" --expression-attribute-values file://operaciones/update-item/valores.json --return-values ALL_NEW
```

En el comando anterior, se ejecuta una orden **UpdateItem**, indicándose los siguientes parámetros:

- **--table-name:** Nombre de nuestra tabla, en este caso *curso*
  - **--key:** Fichero donde se encuentra el documento JSON con la información de los atributos clave del elemento que se obtendrá
  - **--update-expression:** Expresión que indica los atributos del elemento que se actualizarán; en este caso, se incrementará el atributo *convocatoria* en 1, y se añadirá una nueva calificación a la lista de notas mediante la función *list\_append*
  - **--expression-attribute-values:** Fichero donde se encuentran definidos los valores de las variables *:inc* y *:n*, definidos en la expresión de actualización
- 14) **Operación 5:** Para realizar la operación de búsqueda, en este caso se realiza tomando como índice el valor del atributo *modulo*, que es precisamente la clave de partición de la tabla *curso*. Por lo tanto, se deberá ejecutar una operación **Query** indicando la partición donde se realizará la búsqueda. La definición de los valores que toma la clave de partición (*modulo*) se encuentra en el fichero *operaciones/query/valores.json*. En este mismo fichero, también se encuentran los valores necesarios que se necesitan para el atributo *curso* (2023) y la calificación necesaria para aprobar (5):



```
{
  ":mod": { "S": "ED" },
  ":cur": { "N": "2023" },
  ":n": { "N": "5" }
}
```

Para realizar la operación de consulta desde la AWS CLI, se ejecuta el siguiente comando:

```
aws dynamodb query --table-name curso --key-condition-expression "modulo= :mod" --
filter-expression "curso = :cur AND nota[0]>=:n" --expression-attribute-values
file://operaciones/query/valores.json --projection-expression "alumno, nota"
```

En el comando anterior, se ejecuta una orden **Query**, indicándose los siguientes parámetros:

- **--table-name:** Nombre de nuestra tabla, en este caso *curso*
- **--key-condition-expression:** Expresión relacional que determina la partición donde se encuentran los datos. Esta expresión debe ser siempre una comparación de igualdad entre el campo que es clave de partición (*modulo*) con un valor, en este caso especificado en la variable *:mod*
- **--filter-expression.** Expresión lógica que determina, dentro de la partición, las condiciones que deben cumplir los elementos filtrados; en este caso el atributo *curso* debe tener el valor de la variable *:cur*, y la primera calificación, *nota[0]* debe ser superior al valor de la variable *:n*
- **--expression-attribute-values.** Fichero donde se encuentran definidos los valores de las variables *:cur* y *:n*, utilizados en la operación
- **--projection-expression:** Indica los atributos de los elementos obtenidos que se proyectarán en el resultado de la consulta

Como resultado, se obtendrá un documento JSON con los elementos consultados:

```
root@aws:~/aws-academy-fp-daw/resources/dynamodb# aws dynamodb query --table-name curso --key-condition-expression "modulo= :mod" --filter-expression "curso = :cur AND nota[0]>=:n" --expression-attribute-values file://operaciones/query/valores.json --projection-expression "alumno, nota"
{
  "Items": [
    {
      "alumno": {
        "S": "Andres Lledo"
      },
      "nota": {
        "L": [
          {
            "N": "5"
          }
        ]
      }
    },
    {
      "alumno": {
        "S": "Antonio Aparicio"
      },
      "nota": {
        "L": [
          {
            "N": "8"
          }
        ]
      }
    },
    {
      "alumno": {
        "S": "Belen Perello"
      },
      "nota": {
        "L": [
          {
            "N": "6"
          }
        ]
      }
    }
  ],
  "Count": 3,
  "ScannedCount": 8,
  "ConsumedCapacity": null
}
root@aws:~/aws-academy-fp-daw/resources/dynamodb#
```

- 15) **Operación 6:** En este caso, la operación de consulta se hace en base a un atributo que no es la clave de partición, por lo que deberá utilizarse una función **Scan** que permita realizar un barrido por todas las particiones en busca de los elementos que cumplan con los requerimientos. La operación **Scan** sólo es recomendable en los casos que sea estrictamente necesario recorrer todos los elementos de la tabla; en otras condiciones es preferible utilizar una operación **Query** o reorganizar la tabla en base a un índice (ver más adelante). En este caso, el valor del atributo *alumno* que se desea filtrar se encuentra en el archivo *operaciones/scan/valores.json*:

```
{
  ":al": { "S": "Sofia Diaz"}
}
```

Para realizar la operación de consulta desde la AWS CLI, se ejecuta el siguiente comando:

```
aws dynamodb scan --table-name curso --filter-expression "alumno = :al" --expression-attribute-values file://operaciones/scan/valores.json --projection-expression "modulo,nota"
```

En el comando anterior, se ejecuta una orden **Scan**, indicándose los siguientes parámetros:

- **--table-name:** Nombre de nuestra tabla, en este caso *curso*
- **--filter-expression.** Expresión lógica que determina las condiciones que deben cumplir los elementos filtrados; en este caso el atributo *alumno* debe tener el valor de la variable *:al*
- **--expression-attribute-values.** Fichero donde se encuentran definidos los valores de la variable *:al*, con el nombre del alumno

Como resultado, se obtendrá un documento JSON con los elementos consultados:

```
root@aws:~/aws-academy-fp-daw/resources/dynamodb# aws dynamodb scan --table-name curso --filter-expression "alumno = :al" --expression-attribute-values file://operaciones/scan/valores.json --projection-expression "modulo,nota"
{
  "Items": [
    {
      "modulo": {
        "S": "ED"
      },
      "nota": {
        "L": [
          {
            "N": "9,75"
          }
        ]
      }
    },
    {
      "modulo": {
        "S": "PRG"
      },
      "nota": {
        "L": [
          {
            "N": "9"
          }
        ]
      }
    },
    {
      "modulo": {
        "S": "SI"
      },
      "nota": {
        "L": [
          {
            "N": "10"
          }
        ]
      }
    }
  ]
}
```

## Manipulación de datos mediante NoSQL Workbench

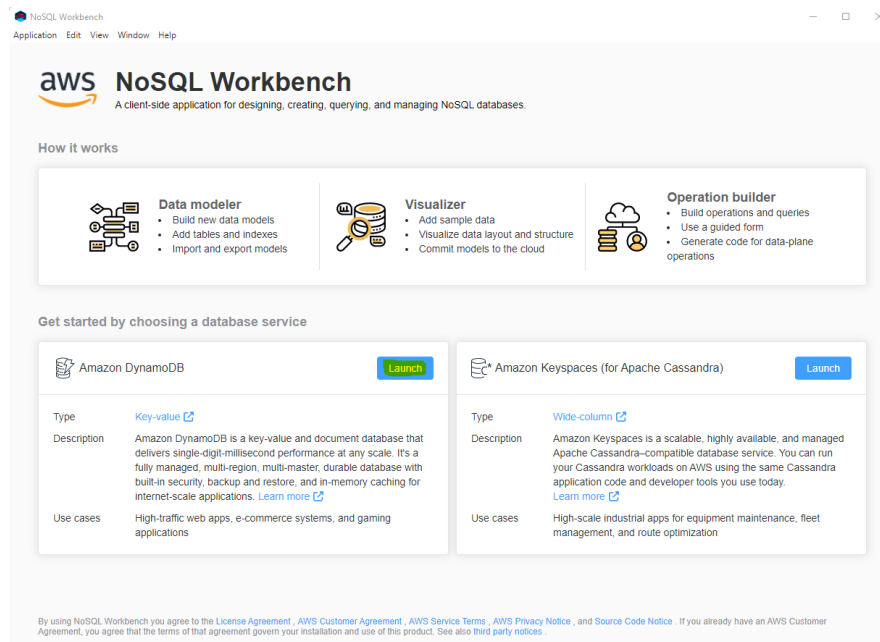
NoSQL Workbench es una herramienta desarrollada por Amazon Web Services (AWS) que se utiliza para diseñar y modelar bases de datos NoSQL en AWS. Está diseñada para trabajar con bases de datos NoSQL, como Amazon DynamoDB y Amazon DocumentDB. NoSQL Workbench facilita la creación y la optimización de esquemas de base de datos NoSQL, lo que ayuda a los desarrolladores a diseñar bases de datos escalables y eficientes.

Algunas de las características clave de NoSQL Workbench incluyen:

- **Modelado visual:** Permite a los usuarios diseñar esquemas de bases de datos NoSQL mediante una interfaz gráfica intuitiva, lo que simplifica la creación de tablas, definición de claves primarias, y relaciones entre datos.
- **Simulación y consulta:** Los desarrolladores pueden realizar simulaciones de consultas para evaluar el rendimiento de las operaciones en sus bases de datos NoSQL, lo que les ayuda a optimizar sus diseños.
- **Integración con servicios de AWS:** NoSQL Workbench se integra de manera nativa con otros servicios de AWS, lo que facilita la configuración y el despliegue de bases de datos NoSQL en la nube de AWS.

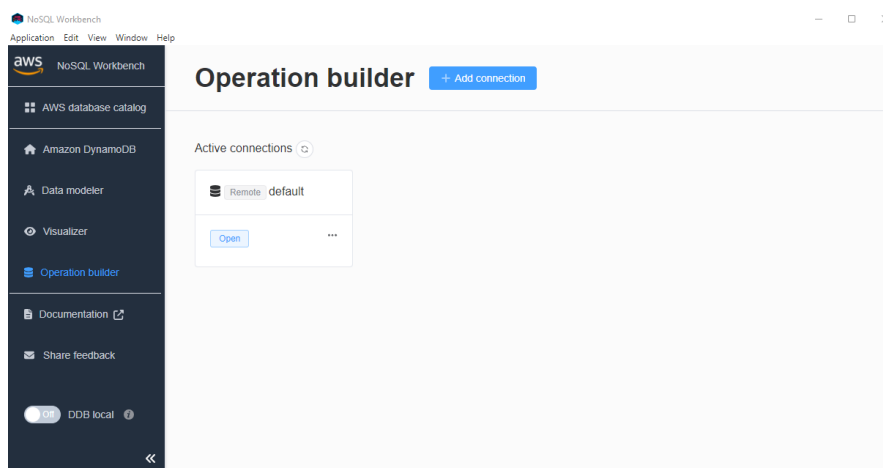
- **Colaboración:** Permite a equipos de desarrollo colaborar en el diseño de bases de datos NoSQL al proporcionar herramientas para compartir esquemas y modelos de datos.
- **Posibilidad de utilizar sintaxis similar a SQL.** NoSQL Workbench incluye una característica llamada **PartiQL** que permite consultar una tabla utilizando una sintaxis restringida, parecida a SQL.

En esta práctica, utilizaremos esta herramienta para ejecutar las operaciones propuestas sobre los datos, de manera similar a cómo se hizo desde la AWS CLI. Para ello ejecutamos la herramienta, tras lo cual aparecerá una pantalla similar a la siguiente:



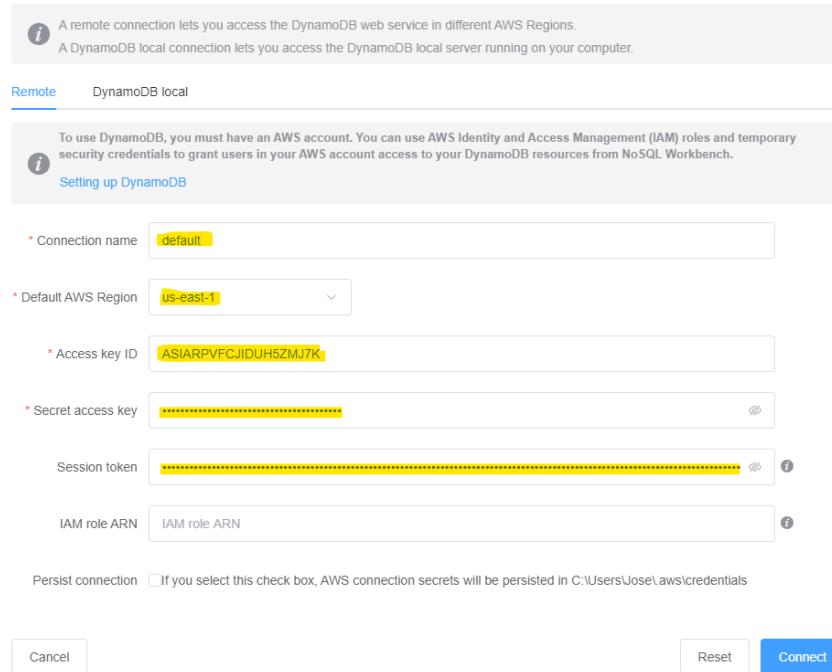
En la ventana anterior, presionamos el botón **Launch** correspondiente a la opción **Amazon DynamoDB**

- 16) Lo primero que se debe hacer es configurar la conexión a la cuenta de AWS Academy Learner Lab. Para ello, desde el menú lateral, presionamos la opción **Operation builder** y comprobamos que nuestra conexión *default* ya se encuentra configurada:



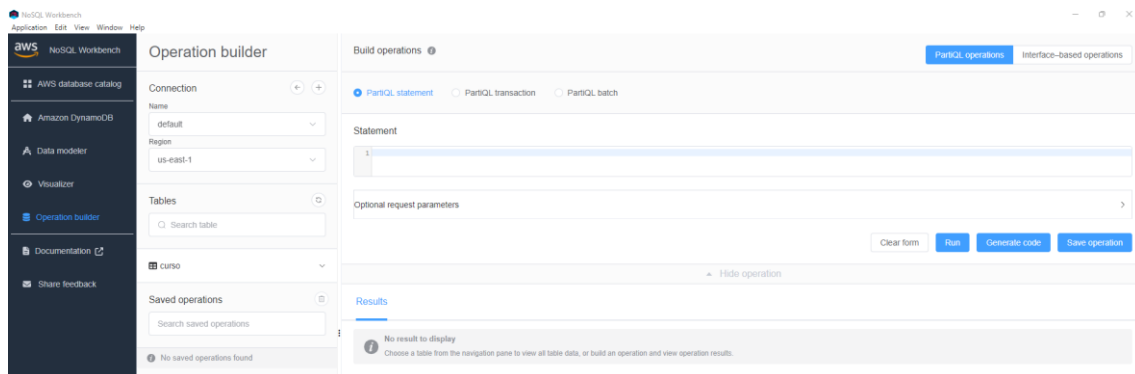
En caso de no estar configurada la conexión *default*, presionamos el botón **Add connection** y, dentro de la pestaña **Remote** añadimos la información con las credenciales de acceso obtenidas del AWS Academy Learner Lab, tal y como se muestra en la siguiente imagen:

Add a new database connection



Por último, presionamos el botón **Connect** para crear la conexión

- 17) Una vez creada la conexión, presionamos el botón **Open**, tras lo cual aparecerá una pantalla donde podremos ejecutar nuestras operaciones, tal y como se muestra a continuación:



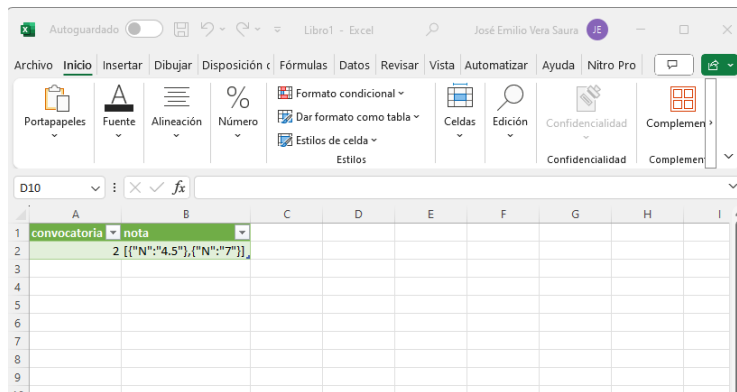
- 18) Para poder ejecutar nuestras operaciones sobre los datos, seleccionaremos nuestra tabla *curso* y, de forma automática, en la pantalla aparecerán un listado de los elementos de nuestra tabla:

| # | modulo | alumno           | curso | convocatoria | nota                              | observaciones |
|---|--------|------------------|-------|--------------|-----------------------------------|---------------|
| 1 | ED     | Ana Suarez       | 2022  | 3            | [[{"N":1}, {"N":3}, {"N":5}]]     |               |
| 2 | ED     | Andres Liedo     | 2023  | 1            | [[{"N":5}]]                       |               |
| 3 | ED     | Antonio Aparicio | 2023  | 1            | [[{"N":6}]]                       |               |
| 4 | ED     | Belen Perello    | 2023  | 1            | [[{"N":6}]]                       |               |
| 5 | ED     | Juan Martinez    | 2023  | 2            | [[{"N":3.5}, {"N":7.5}]]          |               |
| 6 | ED     | Lujan Torres     | 2022  | 1            | [[{"N":7.5}]]                     |               |
| 7 | ED     | Simon Miranda    | 2022  | 2            | [[{"N":4}, {"N":7}]]              |               |
| 8 | ED     | Sofia Diaz       | 2022  | 1            | [[{"N":9.75}]]                    |               |
| 9 | PRG    | Ana Suarez       | 2022  | 4            | [[{"N":2}, {"N":3.5}, {"N":...}]] | No estudia    |

- 19) Para realizar las operaciones presionaremos el botón **Interface-based operations** en la parte superior derecha de la pantalla y, a continuación, mostramos la interfaz de operaciones presionando la opción **Expand operation**:

- 20) Operación 1: En esta operación debemos obtener los datos de la convocatoria y calificaciones del alumno *Antonio Aparicio* en el módulo de programación (*PRG*). Para ello, desde la ventana de NoSQL Workbench seleccionamos la opción **GetItem** del menú desplegable *Operations*, la opción **curso** del menú desplegable *Table name* y, a continuación, completamos los campos de la clave de partición (*modulo*) y la clave de ordenación (*alumno*) con los valores indicados. Por último, activamos la casilla de verificación de **Projection expresión** y añadimos los atributos *nota* y *convocatoria*, tal y como se muestra en la siguiente imagen. Por último, para obtener los datos de la consulta, presionamos el botón **Run**

Una vez ejecutados los resultados, deberíamos poderlos visualizar en NoSQL Workbench, pero por alguna razón, cuando se utilizan expresiones de proyección, la herramienta no funciona bien. No obstante, podemos descargar el documento resultado presionando el botón **Export to CSV** y comprobar que, efectivamente contiene la información solicitada:



Si deseásemos emplear **PartiQL** para ejecutar la operación anterior utilizando SQL, deberíamos activar la opción **PartiQL operations** en la parte superior derecha del interfaz de NoSQL Workbench, e introducir la sentencia en SQL correspondiente, tal y como se muestra en la siguiente figura:

Build operations ⓘ

PartiQL operations Interface-based operations

PartiQL statement PartiQL transaction PartiQL batch

Statement

```
1 select nota, convocatoria from curso where modulo='PH6' and alumno='Antonio Aparicio'
```

Optional request parameters

Clear form Run Generate code Save operation

Hide operation

Results Generated code

Items returned (1) Export to CSV

|   | modulo | alumno |
|---|--------|--------|
| 1 |        |        |

- 21) Operación 2:** Para introducir el nuevo elemento solicitado en la tabla *curso*, procedemos a utilizar la operación **PutItem**, introduciendo los valores correspondientes en la siguiente imagen y presionando el botón **Run**:

Build operations ⓘ

PartiQL operations Interface-based operations

Operations: PutItem Table name: curso

\* Partition key: BD

\* Sort key: Miguel.Dominguez

Other attributes (+) ⓘ

| Attribute name | Attribute type | Attribute value |
|----------------|----------------|-----------------|
| curso          | Number         | 2023            |
| nota           | List           | [{"N": "8.5"}]  |
| convocatoria   | Number         | 1               |

Condition expression: + Condition + Child expression ⓘ

Clear form Run Generate code Save operation

De igual forma, podría haberse ejecutado mediante **PartiQL**, con la expresión indicada en la imagen siguiente:

Build operations ⓘ

PartiQL operations Interface-based operations

PartiQL statement PartiQL transaction PartiQL batch

Statement

```
1 insert into curso value ('modulo': 'BD', 'alumno': 'Miguel Dominguez', 'curso': '2023', 'nota': [8.5], 'convocatoria': '1')
```

Optional request parameters >

Clear form Run Generate code Save operation

Hide operation

Results Generated code

✓ No items returned.

- 22) Operación 3: Para eliminar el elemento solicitado en la tabla *curso*, procedemos a utilizar la operación **DeleteItem**, indicando los valores de los atributos clave del alumno *Andres Lledo* en el módulo *BD*:

Build operations ⓘ

PartiQL operations Interface-based operations

Operations **DeleteItem** Table name **curso**

\* Partition key **ID** ⓘ

\* Sort key **Andres Lledo** ⓘ

Condition expression + Condition + Child expression ⓘ

Clear form Run Generate code Save operation

Hide operation

Results Generated code

✓ No items returned.

De igual manera, podría haberse realizado en *PartiQL*:

Build operations ⓘ

PartiQL operations Interface-based operations

PartiQL statement PartiQL transaction PartiQL batch

Statement

1 `delete from curso where modulo="ID" and alumno="Andres Lledo"`

Optional request parameters >

Clear form Run Generate code Save operation

Hide operation

Results Generated code

✓ No items returned.

- 23) Operación 4: Para actualizar el elemento solicitado en la tabla *curso*, procedemos a utilizar la operación **UpdateItem**, indicando los valores de los atributos clave de la alumna *Ana Suarez* en el módulo *LMSI*, incrementando en 1 el valor del atributo *convocatoria* y añadiendo la calificación 3 al atributo *nota*, tal y como se indica en la siguiente imagen:

Build operations ⓘ

PartiQL operations Interface-based operations

Operations **UpdateItem** Table name **curso**

\* Partition key **LMSI** ⓘ

\* Sort key **Ana Suarez** ⓘ

Update expression + ⓘ

\* Set **convocatoria** **+** **increment\_by** **Number** **1** ⓘ

\* Set **nota** **list\_append** **nota** **end\_of\_list** **List** ⓘ

Condition expression + Condition + Child expression ⓘ

Clear form Run Generate code Save operation



De forma similar, en PartiQL:

Build operations ⓘ

PartiQL operations Interface-based operations

☒ PartiQL statement
 ☐ PartiQL transaction
 ☐ PartiQL batch

Statement

```

1 update curso
2 set convocatoria = convocatoria + 1
3 set nota = list_append(nota,[3])
4 where modulo = "LMSI" and alumno = "Ana Suarez"
  
```

Optional request parameters >

Clear form Run Generate code Save operation

- 24) **Operación 5:** Para consultar los elementos solicitados en la tabla *curso*, procedemos a utilizar la operación **Query**, indicando el valor de la clave de partición y especificando la condición indicada en el enunciado:

Build operations ⓘ

PartiQL operations Interface-based operations

Operations: Query Table name: curso

\* Partition key: ED

☐ Sort key ⓘ

☒ Projection expression ⓘ +

\* Projected attribute: alumno

\* Projected attribute: nota

Filter expression + Condition + Child expression ⓘ

And Or Negate ☐

\* Condition: Negate ☐ curso == Number 2023

Other parameters >

Clear form Run Generate code Save operation

De forma similar, en PartiQL:

Build operations ⓘ

PartiQL operations Interface-based operations

☒ PartiQL statement
 ☐ PartiQL transaction
 ☐ PartiQL batch

Statement

```

1 select alumno, nota
2 from curso
3 where modulo="ED" and curso=2023
  
```

Optional request parameters >

Clear form Run Generate code Save operation

25) **Operación 6:** Para consultar los elementos solicitados en la tabla *curso*, procedemos a utilizar la operación **Scan**, indicando las condiciones de filtrado:

The screenshot shows the 'Build operations' interface in the AWS IAM console. The 'Interface-based operations' tab is selected. Under 'Operations', 'Scan' is chosen, and the 'Table name' is 'curso'. The 'Filter expression' section shows a condition: 'alumno' is equal to 'Sofia Diaz'. The 'Projection expression' section shows two attributes: 'module' and 'nota'. At the bottom, there are buttons for 'Clear form', 'Run', 'Generate code', and 'Save operation'.

De forma similar, en PartiQL:

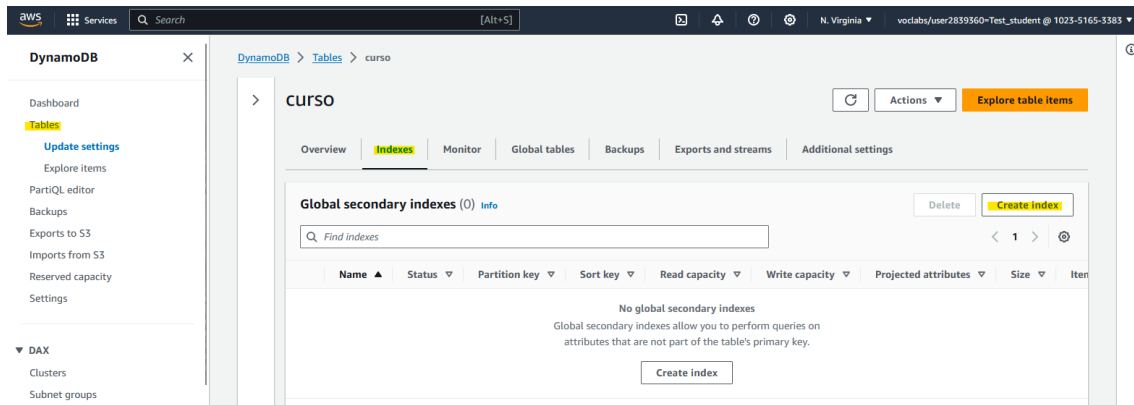
The screenshot shows the 'Build operations' interface in the AWS IAM console, but with the 'PartiQL operations' tab selected. Under 'PartiQL statement', the following SQL query is entered: `1 select module, nota  
2 from curso  
3 where alumno='Sofia Diaz'`. At the bottom, there are buttons for 'Clear form', 'Run', 'Generate code', and 'Save operation'.

## DEFINICIÓN DE ÍNDICES SECUNDARIOS GLOBALES (GSI, *GLOBAL SECONDARY INDEX*)

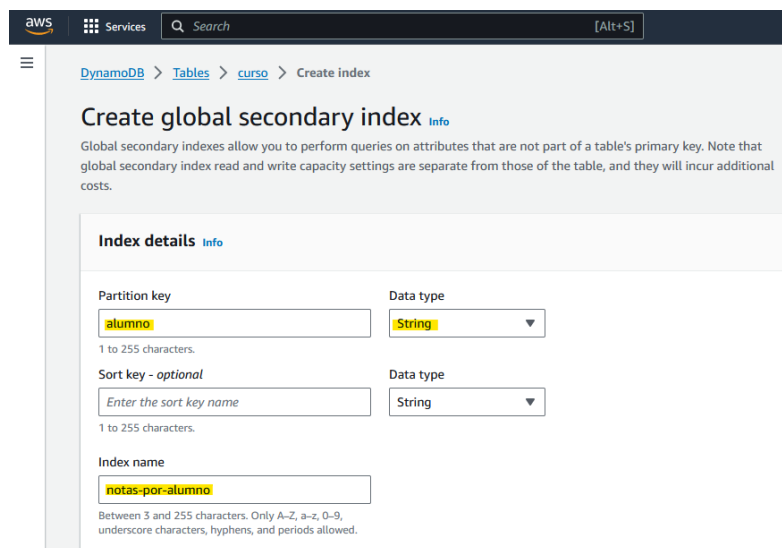
Los índices secundarios globales de Amazon DynamoDB permiten tener particionada una tabla en base a un atributo diferente de la clave de partición y ordenada en base a otra clave de ordenación diferente. Esta herramienta permite poder ejecutar eficientemente consultas que acceden a datos que no están particionados en función de la clave de partición de la tabla.

En esta práctica, en concreto, la última operación realizada busca en todas las particiones de la tabla *curso* las calificaciones de un alumno concreto. Si la tabla hubiera estado particionada por el campo *alumno*, se podría haber ejecutado una operación **Query** en lugar de una operación **Scan**, obteniendo un resultado más eficiente y con un menor consumo de RCUs (*Read Capacity Units*)

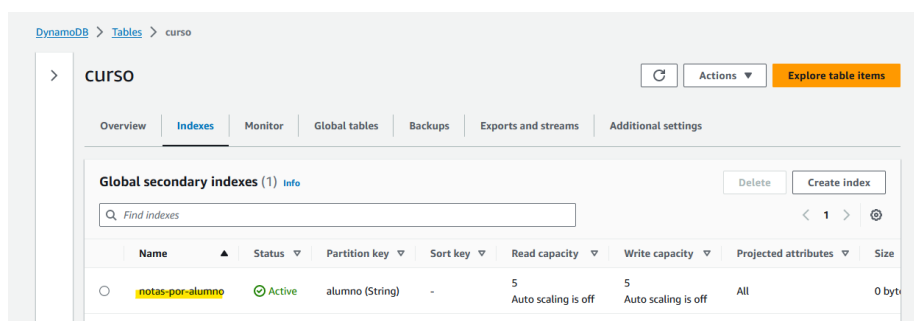
- 26) Para crear un índice secundario global, desde la Consola de Administración del servicio Amazon DynamoDB, accedemos a la tabla *curso*, activamos la pestaña **Indexes** y presionamos el botón **Create index**:



- 27) Desde la siguiente pantalla, introducimos como clave de partición el atributo *alumno* e indicamos como nombre del índice *notas-por-alumno*, dejamos el resto de los atributos por defecto y presionamos el botón **Create index**:



Tras unos minutos, nuestro índice estará creado:



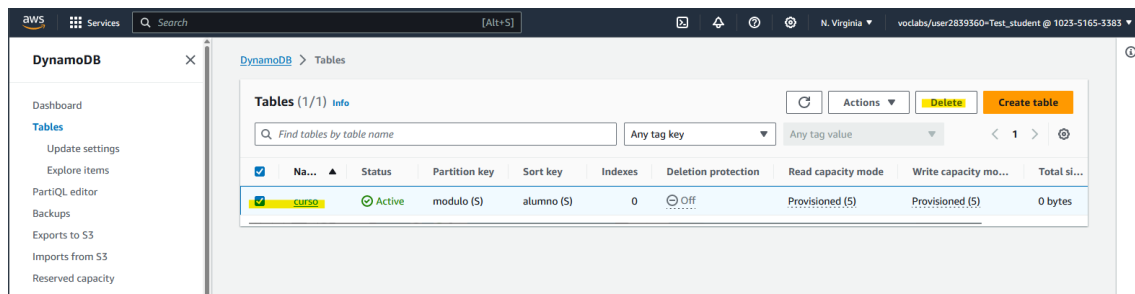
- 28) A continuación, podríamos utilizar este índice para ejecutar eficientemente consultas sobre elementos que se encuentren en la misma partición correspondiente al campo clave de partición definido en el índice. En el caso de la operación 6, podríamos ejecutar directamente una operación **Query** en lugar de una operación **Scan**:

```
aws dynamodb query --table-name curso --index-name notas-por-alumno --key-condition-expression "alumno= :al" --expression-attribute-values '{ ":al": { "S": "Sofia Diaz" } }' --projection-expression "modulo,nota"
```

### Limpieza de la Práctica:

Para terminar esta práctica y liberar los recursos creados, evitando así el consumo de créditos de AWS Academy Learner Labs, simplemente debemos dar los siguientes pasos:

- Eliminar la tabla de Amazon DynamoDB. Para ello, desde la consola de Amazon RDS seleccionamos nuestra instancia y, desde el menú **Actions** elegimos la opción **Delete**.



En la siguiente ventana, confirmamos la eliminación tecleando en el cuadro de texto la palabra solicitada y presionamos el botón **Delete**:

