

1. DynamoDB

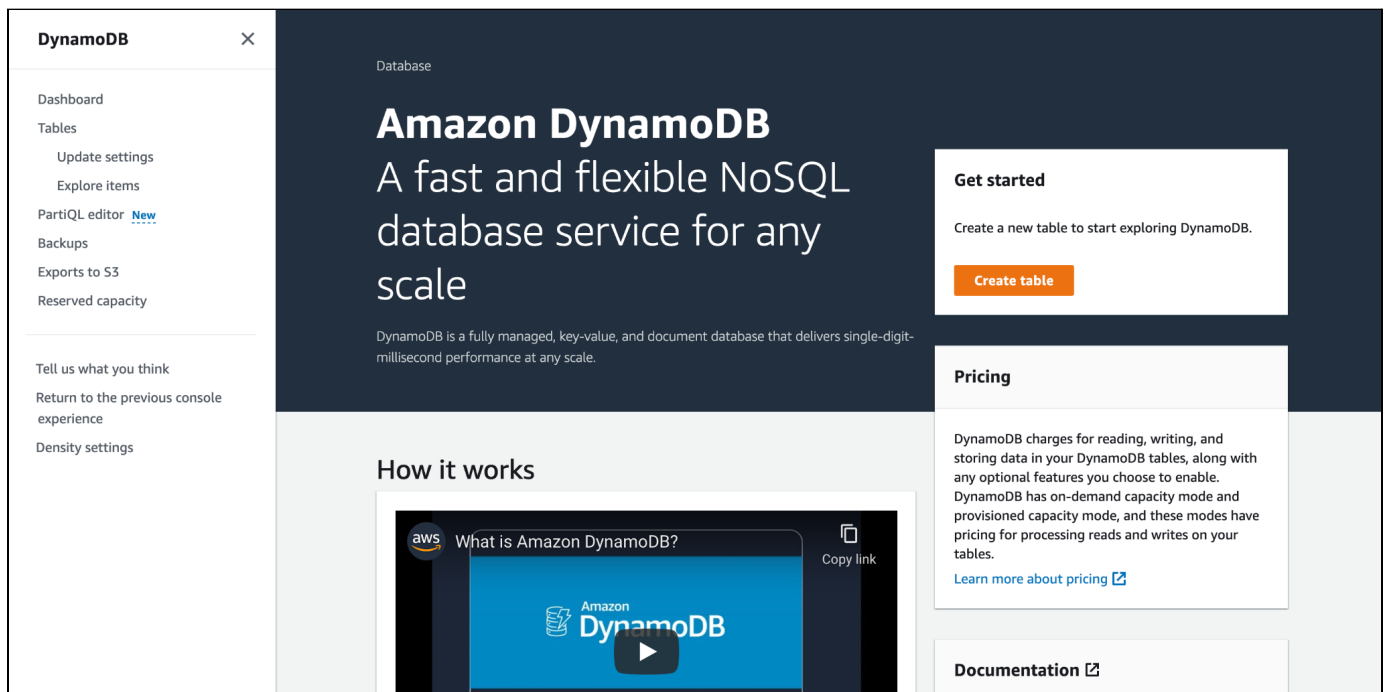


Figure 1: DynamoDB Home page

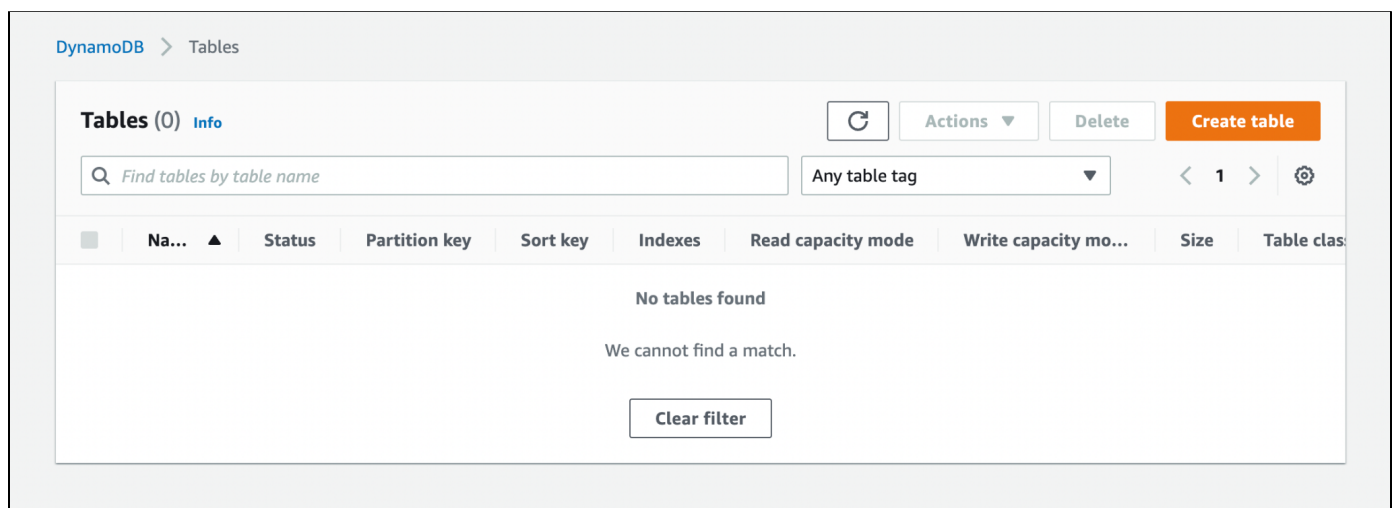


Figure 2: DynamoDB dashboard before creating a collection

2. Code snippets

Generic DynamoDB class to add and update items in the collection "Parks_NovaScotia".

```
public class DynamoDB {
    private AmazonDynamoDB client;

    public DynamoDB() {
        AWSCredentials credentials = new
BasicAWSCredentials(System.getenv("AWS_ACCESS_KEY"),
System.getenv("AWS_SECRET_KEY"));
        this.client = AmazonDynamoDBClientBuilder
            .standard()
            .withRegion("ca-central-1")
            .withCredentials(new AWSStaticCredentialsProvider(credentials))
            .build();
    }

    public void add(String name, String region, Map<String, String> attributes)
    {
        PutItemRequest request = new PutItemRequest();
        request.setTableName("Parks_NovaScotia");

        Map<String, AttributeValue> keysMap = new HashMap<>();
        for (Map.Entry<String, String> entry: attributes.entrySet()) {
            keysMap.put(entry.getKey(), new AttributeValue(entry.getValue()));
        }
        keysMap.put("name", new AttributeValue(name));
        keysMap.put("region", new AttributeValue(region));
        request.setItem(keysMap);

        try {
            PutItemResult result = this.client.putItem(request);
            System.out.println("Status: " +
result.getSdkHttpMetadata().getHttpStatusCode());
        } catch (AmazonServiceException e) {
            System.out.println(e.getErrorMessage());
        }
    }

    public void update(String name, String region, Map<String, AttributeValue>
campsites) {
        UpdateItemRequest request = new UpdateItemRequest();
        request.setTableName("Parks_NovaScotia");
```

```

        request.setReturnConsumedCapacity(ReturnConsumedCapacity.TOTAL);

        Map<String, AttributeValue> keysMap = new HashMap<>();
        keysMap.put("name", new AttributeValue(name));
        keysMap.put("region", new AttributeValue(region));
        request.setKey(keysMap);

        /* Create a Map of attributes to be updated */
        Map<String, AttributeValueUpdate> map = new HashMap<>();
        map.put("types_of_campsites", new AttributeValueUpdate(new
        AttributeValue().withM(campsites), "PUT"));
        request.setAttributeUpdates(map);

        try {
            UpdateItemResult result = this.client.updateItem(request);
            System.out.println("Status: " +
            result.getSdkHttpMetadata().getHttpStatusCode());
            System.out.println("Consumed Capacity: " +
            result.getConsumedCapacity().getCapacityUnits());

            if (result.getAttributes() != null) {
                result.getAttributes().entrySet().stream()
                    .forEach(e -> System.out.println(e.getKey() + " " +
            e.getValue()));
            }
        } catch (AmazonServiceException e) {
            System.out.println(e.getErrorMessage());
        }
    }
}

```

Usage:

```

private static void update() {
    DynamoDB dynamoDB = new DynamoDB();

    Map<String, AttributeValue> campsitesOne = new HashMap<>();
    campsitesOne.put("Unserviced", new AttributeValue("Mixture of wooded,
    partially wooded and open campsites with parking, "));
    campsitesOne.put("Serviced", new AttributeValue("A mixture of open and
    partially wooded campsites (sites 4, 5, 6, 7, 8 & 26 - 41)"));
}

```

```

    dynamoDB.update("Battery", "Saint Peters", campsitesOne);
}

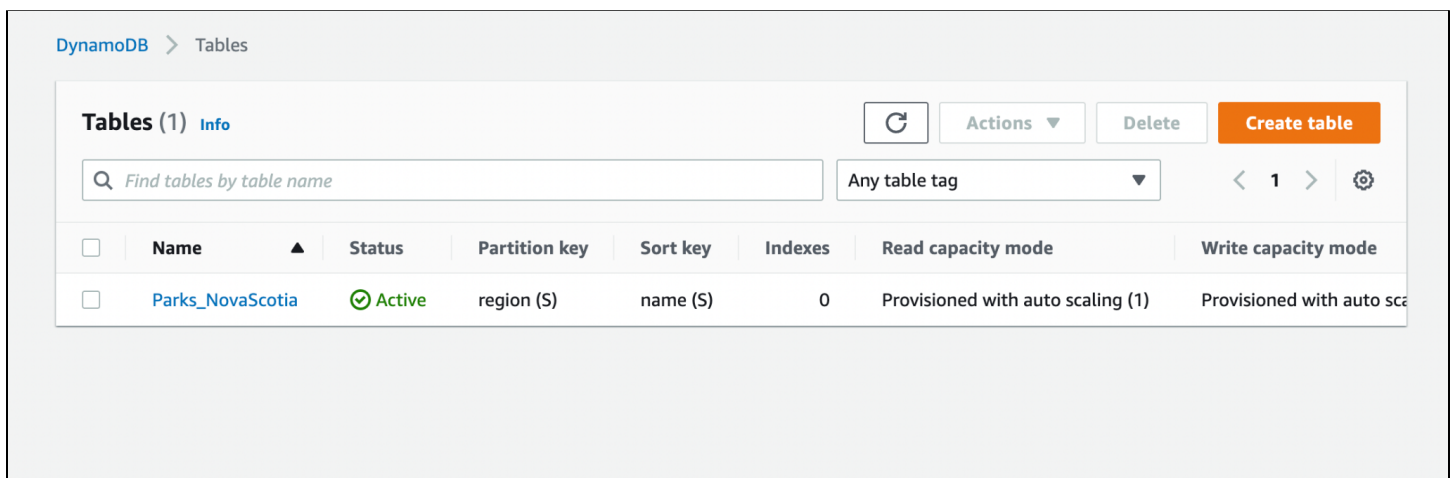
private static void add() {
    DynamoDB dynamoDB = new DynamoDB();
    Map<String, String> attributesOne = new HashMap<>();
    attributesOne.put("camper_shelter", "Picnic tables, picnic shelters,
water tap, vault toilets and beach access overlooking");
    attributesOne.put("change_house", "NA");
    attributesOne.put("park_office", "Deck is accessible by a wooden ramp
with a 6% grade however the entrance door ");
    attributesOne.put("swimming", "Unsupervised swimming at the beach");
    attributesOne.put("trailer_dump", "Dump station");
    attributesOne.put("wifi", "Available at the entry building and
surrounding area.");
    dynamoDB.add("Battery", "Saint Peters", attributesOne);
}

```

Gitlab Repository:

<https://git.cs.dal.ca/adimurthy/csci5410-b00886154-adesh-nalpet-adimurthy/-/blob/master/src/main/java/A1/DynamoDB.java>

3. Outputs



The screenshot shows the AWS DynamoDB console interface. At the top, there's a breadcrumb 'DynamoDB > Tables'. Below this, a header bar contains 'Tables (1) Info', a refresh button, an 'Actions' dropdown, a 'Delete' button, and a 'Create table' button. A search bar with the placeholder 'Find tables by table name' and a dropdown for 'Any table tag' are also present. The main content is a table with the following columns: Name, Status, Partition key, Sort key, Indexes, Read capacity mode, and Write capacity mode. One table is listed: 'Parks_NovaScotia' with a status of 'Active' (indicated by a green checkmark), a partition key of 'region (S)', a sort key of 'name (S)', 0 indexes, and both read and write capacity modes set to 'Provisioned with auto scaling (1)'.

	Name	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode
<input type="checkbox"/>	Parks_NovaScotia	Active	region (S)	name (S)	0	Provisioned with auto scaling (1)	Provisioned with auto scaling (1)

Figure 3a: After creating a collection (empty).

DynamoDB > Tables > Parks_NovaScotia

Tables (1) X

Any table tag

Find tables by table name

< 1 > ⚙

Parks_NovaScotia

Parks_NovaScotia

Actions Explore table items

< Monitor Global tables Backups Exports and streams Additional settings >

General information

Partition key region (String)	Sort key name (String)	Capacity mode Provisioned	Table status ✔ Active ✔ No active alarms
----------------------------------	---------------------------	------------------------------	--

► Additional info

Items summary

DynamoDB updates the following information approximately every six hours.

Get live item count

Item count 0	Table size 0 bytes	Average item size 0 bytes
-----------------	-----------------------	------------------------------

Table capacity metrics

View all metrics

Figure 3b: After creating a collection (empty).

DynamoDB > Items: Parks_NovaScotia > Item editor

Item editor

Form JSON

Attributes

Add new attribute

Attribute name	Value	Type	
region - Partition key	Northumberland Shore	String	New
name - Sort key	Amherst Shore	String	New
camper_shelter	Open-air style camper shelter centrally located in the camping area	String	Remove
change_house	Located near the beach	String	Remove
park_office	Accessible entrance from a solid wooden ramp, door has a 12mm bevelled threshold and clear width of entry	String	Remove
swimming	Unsupervised swimming at beach	String	Remove
trailer_dump	Dump station located near park office.	String	Remove
wifi	Available at the park office and surrounding area.	String	Remove

Cancel Save changes

Figure 4: After creating a record - Park.

1

Parks_NovaScotia

Scan

Query

Table or index

Parks_NovaScotia

▼ Filters

Attribute name

Type

Condition

Value

region

String

Equal to

Northumberl...

Remove

Add filter

Run

Reset

✔ Completed

Read capacity units consumed: 0.5

Items returned (2)

< 1 >

<input type="checkbox"/>	region	name	camper_...	change_...
<input type="checkbox"/>	Northumberland Shore	Amherst Shore	Open-air st...	Located ne...
<input type="checkbox"/>	Northumberland Shore	Caribou-Munroes Island	Parking, be...	Located ne...

Figure 5: After creating multiple items and querying on region (partition key).

DynamoDB > Items: Parks_NovaScotia > Item editor

Item editor

FormJSON

Attributes

Add new attribute ▼

Attribute name	Value	Type	
change_house	Located near the beach	String	Remove
park_office	Accessible entrance from a solid wooden ramp, door has a 12mm bevelled threshold and clear width of entry	String	Remove
swimming	Unsupervised swimming at beach	String	Remove
trailer_dump	Dump station located near park office.	String	Remove
types_of_campsites	Insert a field ▼	Map	Remove
Serviced	Wooded sites with two way hook up (30 amp electrical service and drinking water)	String	Remove
Unserviced	Wooded sites with parking, picnic table and campfire ring/grill	String	Remove
wifi	Available at the park office and surrounding area.	String	Remove

CancelSave changes

Figure 6: After updating the item from java code (types of campsites).

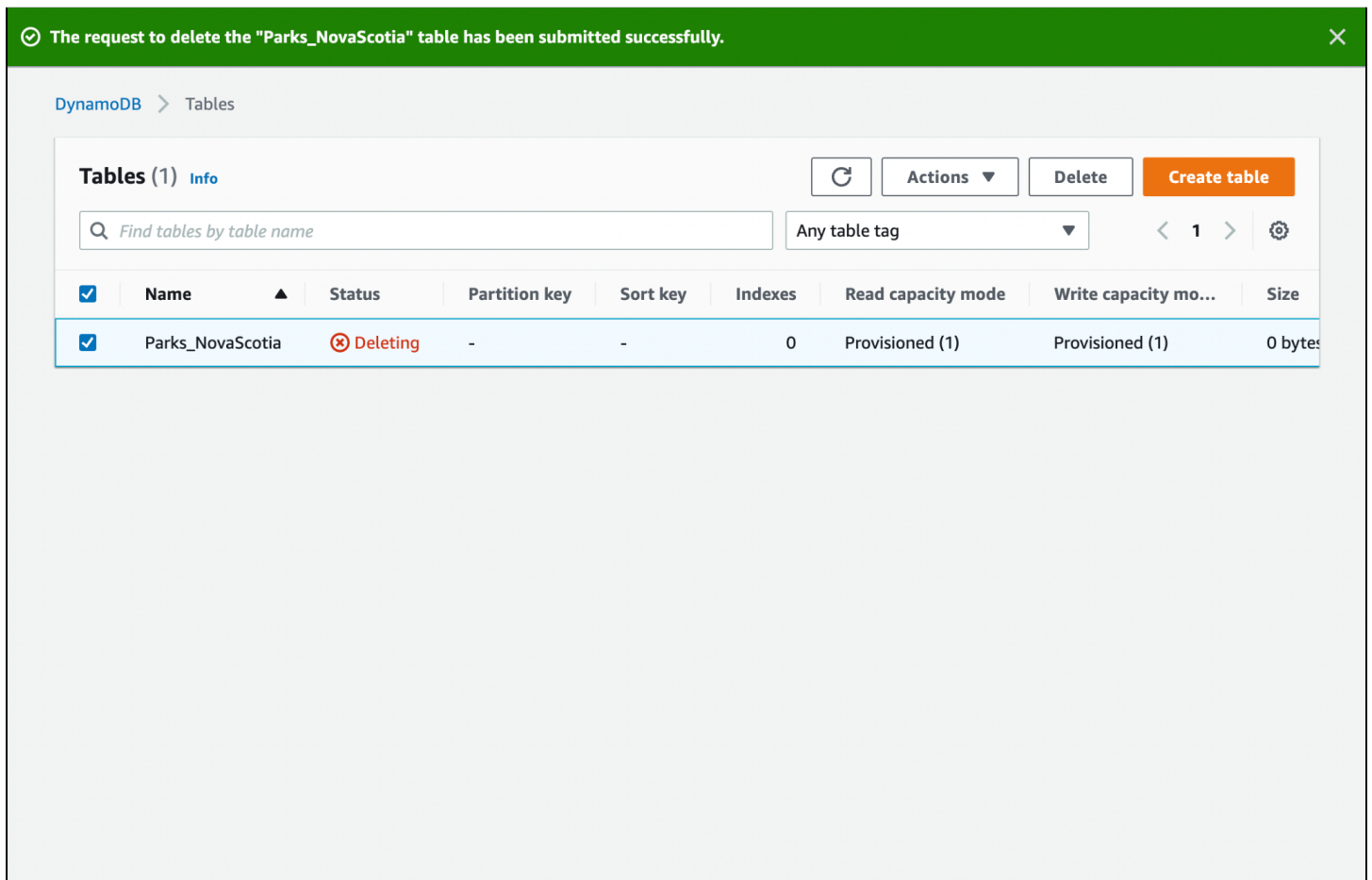


Figure 7: Deleting the table after completing the assignment.

4. References

[1] "AWS SDK for Java," Amazon Web Services, Inc. <https://aws.amazon.com/sdk-for-java/> (accessed May 27, 2022).

[2] "DynamoDB Examples Using the AWS SDK for Java - AWS SDK for Java," docs.aws.amazon.com.

<https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/examples-dynamodb.html> (accessed May 27, 2022).

[3] "Maven Repository: com.amazonaws» aws-java-sdk," mvnrepository.com.

<https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk> (accessed May 27, 2022).