

# Load and Query DynamoDB Tables

MA

HARI NATH REDDY

The screenshot shows the AWS Lambda console interface for querying a DynamoDB table named 'ContentCatalog'. The top navigation bar includes 'Find tables', 'Scan', 'Query', 'Select a table or index', 'Table - ContentCatalog', 'Select attribute projection', 'All attributes', 'Filters', 'Run', and 'Reset'. A green status bar at the bottom indicates 'Completed. Read capacity units consumed: 0.5'. The main area displays a table titled 'Items returned (5)' with columns: ID (Number), Authors, ContentType, Difficulty, Price, ProjectCategory, Published, Services, Title, and URL. The data rows are:

ID (Number)	Authors	ContentType	Difficulty	Price	ProjectCategory	Published	Services	Title	URL
3	[{"S": "Ne..."}]	Project	Easy peasy	0	AI/ML	true	(Build a Ch...)	aws-ai-fex1	
2	[{"S": "Ne..."}]	Project	Easy peasy	0	Analytics	true	(Visualize da...)	aws-analytic...)	
203	Video			0			[{"S": "Am..."}]	AWS Data...	https://you...
202	Video			0				Don't miss ...	https://you...
201	Video			0			[{"S": "Am..."}]	AWS Relati...	https://you...
1	[{"S": "Nat..."}]	Project	Easy peasy	0	Storage	true	(Host a Web...)	aws-host-a...	

# Introducing Today's Project!

## What is Amazon DynamoDB?

Amazon DynamoDB is a non-relational database service. Non-relational databases use structures other than rows and columns to organise data.

## How I used Amazon DynamoDB in this project

In today's project, I used Amazon DynamoDB to create live tables! I used CloudShell and CLI to load data into my tables quickly.

## One thing I didn't expect in this project was...

one thing I didn't expect in this project was that we can perform a lot of actions using AWS CLI instead of the console. e.g creating tables, loading data, deleting tables.

## This project took me...

This project took me 2 Hours to complete include documentation.

# Create a DynamoDB table

DynamoDB tables organises data using items and attributes! Every single item is recorded with a set of attributes. Items can have any number of attributes (minimum 1, for the partition key value.)

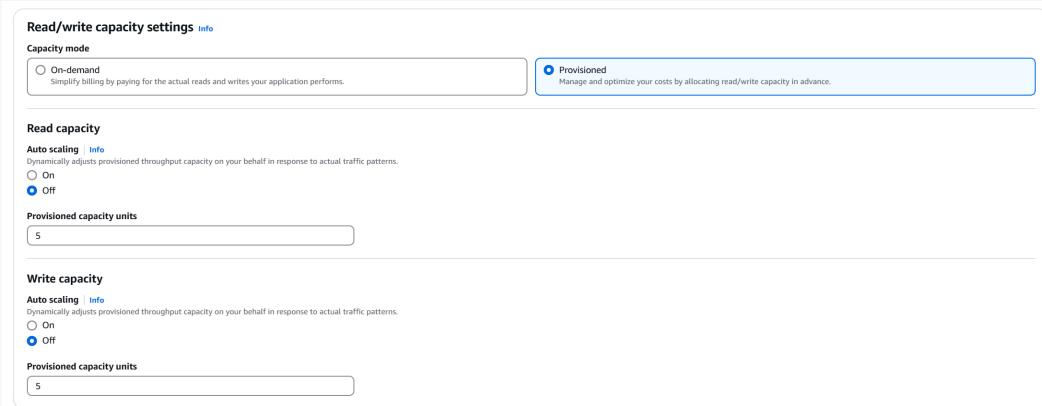
In DynamoDB, an attribute is like a piece of data about an item. In this case, our item is Nikko and the attribute is the number of projects Nikko completed.

The screenshot shows the AWS DynamoDB console interface for the 'NextWorkStudents' table. At the top, there are tabs for 'Scan' (selected) and 'Query'. Below these are dropdown menus for 'Select a table or index' (set to 'Table - NextWorkStudents') and 'Select attribute projection' (set to 'All attributes'). There is also a 'Filters' section with a 'Run' button and a 'Reset' link. A message bar at the bottom indicates 'Completed. Read capacity units consumed: 0.5'. The main area displays the results of the scan, titled 'Items returned (1)'. It shows one item with attributes: 'StudentName (String)' with value 'Nikko' and 'ProjectsComplete' with value '4'. On the right side of the results table are 'Actions' and 'Create item' buttons, along with navigation controls for pages 1 and 2.

# Read and Write Capacity

Think of read capacity units (RCUs) as a way to measure how many engines DynamoDB is using to operate. Write capacity units (WCUs) are just like read capacity units - they give your DynamoDB tables the engines to edit/update/delete data!

Amazon DynamoDB's Free Tier covers 25GB of data storage, plus 25 Write and 25 Read Capacity Units (WCU, RCU) I turned off auto scaling because this can result in higher charges if DynamoDB automatically scales up my operations.



# Using CLI and CloudShell

AWS CloudShell is shell in your AWS Management Console, which means it's a space for you to run code! The awesome thing about AWS CloudShell is that it already has AWS CLI pre-installed.

AWS CLI (Command Line Interface) is a software that lets you create, delete and update AWS resources with commands instead of clicking through your console.

I ran a CLI command in AWS CloudShell that created four new tables in AWS DynamoDB, each with specific attributes and settings.



```
[cloudshell]:~$ aws dynamodb create-table \
>   --table-name "Table1" \
>   --attribute-definitions \
>     AttributeDefinitionAttributeName="A" \
>     AttributeDefinitionAttributeType="S" \
>   --key-schema \
>     KeySchemaAttributeName="A" \
>     KeySchemaKeyType="HASH" \
>   --provisioned-throughput \
>     ProvisionedThroughputReadCapacityUnits=1 \
>     ProvisionedThroughputWriteCapacityUnits=1 \
>   --query "TableDescription.TableStatus"
[cloudshell]:~$ aws dynamodb create-table \
>   --table-name "Table2" \
>   --attribute-definitions \
>     AttributeDefinitionAttributeName="B" \
>     AttributeDefinitionAttributeType="S" \
>   --key-schema \
>     KeySchemaAttributeName="B" \
>     KeySchemaKeyType="HASH" \
>   --provisioned-throughput \
>     ProvisionedThroughputReadCapacityUnits=1 \
>     ProvisionedThroughputWriteCapacityUnits=1 \
>   --query "TableDescription.TableStatus"
[cloudshell]:~$ aws dynamodb create-table \
>   --table-name "Table3" \
>   --attribute-definitions \
>     AttributeDefinitionAttributeName="C" \
>     AttributeDefinitionAttributeType="N" \
>   --key-schema \
>     KeySchemaAttributeName="C" \
>     KeySchemaKeyType="HASH" \
>   --provisioned-throughput \
>     ProvisionedThroughputReadCapacityUnits=1 \
>     ProvisionedThroughputWriteCapacityUnits=1 \
>   --query "TableDescription.TableStatus"
[cloudshell]:~$ aws dynamodb create-table \
>   --table-name "Table4" \
>   --attribute-definitions \
>     AttributeDefinitionAttributeName="D" \
>     AttributeDefinitionAttributeType="N" \
>   --key-schema \
>     KeySchemaAttributeName="D" \
>     KeySchemaKeyType="HASH" \
>   --provisioned-throughput \
>     ProvisionedThroughputReadCapacityUnits=1 \
>     ProvisionedThroughputWriteCapacityUnits=1 \
>   --query "TableDescription.TableStatus"
[cloudshell]:~$
```

# Loading Data with CLI

I ran a CLI command in AWS CloudShell(aws dynamodb batch-write-item) that load multiple pieces of data into the DynamoDB tables. I set up in the previous step.

```
}
[cloudshell-user@ip-10-132-68-181 nextworksampleddata]$ aws dynamodb batch-write-item --request-items file://ContentCatalog.json
{
    "UnprocessedItems": {}
}
[cloudshell-user@ip-10-132-68-181 nextworksampleddata]$
[cloudshell-user@ip-10-132-68-181 nextworksampleddata]$ aws dynamodb batch-write-item --request-items file://Forum.json
{
    "UnprocessedItems": {}
}
[cloudshell-user@ip-10-132-68-181 nextworksampleddata]$
[cloudshell-user@ip-10-132-68-181 nextworksampleddata]$ aws dynamodb batch-write-item --request-items file://Post.json
{
    "UnprocessedItems": {}
}
[cloudshell-user@ip-10-132-68-181 nextworksampleddata]$
[cloudshell-user@ip-10-132-68-181 nextworksampleddata]$ aws dynamodb batch-write-item --request-items file://Comment.json
{
    "UnprocessedItems": {}
}
[cloudshell-user@ip-10-132-68-181 nextworksampleddata]$ █
```

# Observing Item Attributes

Attributes		
Attribute name	Value	Type
Id - Partition key	1	Number
Authors	<input type="button" value="Insert a field ▾"/>	List
ContentType	Project	String
Difficulty	Easy peasy	String
Price	0	Number
ProjectCategory	Storage	String
Published	<input checked="" type="radio"/> True <input type="radio"/> False	Boolean
Title	Host a Website on Amazon S3	String
URL	aws-host-a-website-on-s3	String

[Add new attribute ▾](#)

[Cancel](#) [Save](#) [Save and close](#)

I checked a ContentCatalog item, which had the following attributes: Id, Authors, ContentType, Difficulty, Price, ProjectCategory, Published, Title, and URL.

I checked another ContentCatalog item, which had a different set of attributes: Services, Title, VideoType etc.

# Benefits of DynamoDB

A benefit of DynamoDB over relational databases is flexibility, because every item having their own unique set of attributes is a huge advantage when items in a table could look different from each other.

Another benefit over relational databases is speed, because DynamoDB tables can use partition keys to split up a table and quickly find the items they're looking for. Relational databases have to scan through the entire table to find data, which can

The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with a 'Find tables' search bar and a list of tables: Comment, ContentCatalog (selected), Forum, NextWorkStudents, and Post. The main area has tabs for 'Scan' and 'Query'. Under 'Scan', there's a 'Select a table or index' dropdown set to 'Table - ContentCatalog' and a 'Select attribute projection' dropdown set to 'All attributes'. Below these are 'Filters' and 'Run' and 'Reset' buttons. A green status bar at the bottom says 'Completed. Read capacity units consumed: 0.5'. The bottom section is titled 'Items returned (6)' and contains a table with columns: Id (Number), Authors, ContentType, Difficulty, Price, ProjectCategory, Published, Services, Title, and URL. The table rows show various items like 'Build a Chatbot', 'Visualize data', and 'Host a Web...'. Each row has an 'Actions' button and a 'Create item' button.

Id (Number)	Authors	ContentType	Difficulty	Price	ProjectCategory	Published	Services	Title	URL
3	[{"S": "Ne..."}]	Project	Easy peasy	0	AI/ML	true		Build a Cha...	aws-ai-lex1
2	[{"S": "Ne..."}]	Project	Easy peasy	0	Analytics	true		Visualize da...	aws-analyti...
203		Video		0				[{"S": "Am..."}]	AWS x Data...
202		Video		0					Don't miss ...
201		Video		0				[{"S": "Am..."}]	AWS Relati...
1	[{"S": "Nat..."}]	Project	Easy peasy	0	Storage	true		Host a Web...	aws-host-a...



NextWork.org

# Everyone should be in a job they love.

Check out nextwork.org for  
more projects

