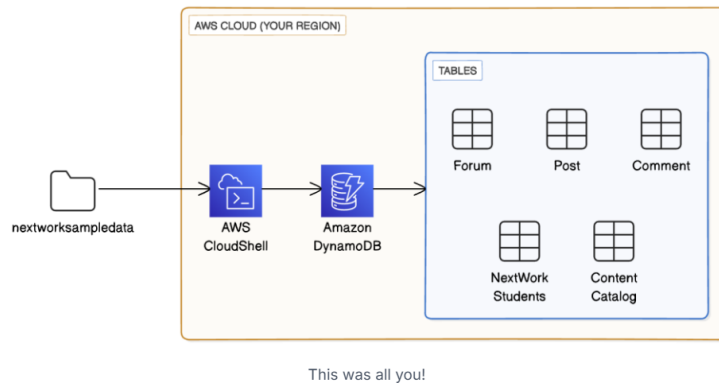


Load and Query DynamoDB Tables



Introducing Today's Project!

What is Amazon DynamoDB?

Amazon DynamoDB is a fully managed NoSQL database service offering fast, scalable, and low-latency performance. It's useful for applications that need high availability and quick data access, handling large amounts of data efficiently.

How I used Amazon DynamoDB in this project

In today's project, I used Amazon DynamoDB to create tables with specific attributes and keys. I loaded data into the tables using the `batch-write-item` command and interacted with the data by viewing and updating it, all through AWS CloudShell.

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

One thing I didn't expect in this project was how easy it was to load data into DynamoDB using the `batch-write-item` command. The process was seamless, and I didn't have to manually input each item, making it more efficient than anticipated.

This project took me...

This project took me around **30 to 45 minutes** to complete. It involved creating tables in DynamoDB, loading data using the AWS CLI, and interacting with the data. The time may vary depending on familiarity with AWS services and the specific tasks.

Create a DynamoDB table.

DynamoDB tables organize data using items and attributes. Each item has a primary key (partition key and optionally a sort key) and associated attributes. Items can have different attributes, providing flexibility compared to relational databases.

An attribute is a piece of data that describes an item in DynamoDB. For example, if an item is "Santhosh," an attribute could be the number of projects completed. Items can have different attributes, offering more flexibility than relational databases.

Items returned (1)				Actions ▼	Create item
<input type="checkbox"/>	StudentName (String) ▼	ProjectsComplete			
<input type="checkbox"/>	Santhosh	4			

Read and Write Capacity

Read capacity units (RCUs) and write capacity units (WCUs) are measures of throughput in DynamoDB. RCUs handle read operations per second, and WCUs manage write operations per second. More units increase capacity but also raise costs based on usage.

Amazon DynamoDB's Free Tier covers **25 GB of data storage**, **25 RCUs**, and **25 WCUs**, supporting up to **200 million requests per month** for free. I turned off auto scaling to avoid unexpected charges from automatic capacity adjustments.

The screenshot shows the AWS Management Console interface for configuring a DynamoDB table. The top navigation bar includes the AWS logo, a search bar, and user information for 'Santhosh-IAM-Admin' in the 'Asia Pacific (Mumbai)' region. The main content area is titled 'DynamoDB > Tables > Create table'. Below this, the 'Capacity calculator' section is expanded, showing input fields for 'Average item size (KB)' (1), 'Item read/second' (1), and 'Item write/second' (1). It also has dropdowns for 'Read consistency' (Eventually consistent) and 'Write consistency' (Standard). A summary row shows 'Read capacity units' (1), 'Write capacity units' (1), 'Region' (ap-south-1), and 'Estimated cost' (\$0.67 / month). Below the calculator, the 'Read/write capacity settings' section shows 'Capacity mode' with 'On-demand' selected and 'Provisioned' highlighted. The 'Read capacity' section has 'Auto scaling' selected.

Using CLI and CloudShell

AWS CloudShell is a browser-based terminal with a pre-configured environment for running AWS CLI commands. It allows you to manage AWS resources directly from the console, simplifying automation and tasks without needing to install tools locally.

AWS CLI is a command-line interface that lets you manage AWS services and resources using commands instead of the AWS Management Console. It enables automation, making tasks like creating, updating, and managing resources easier from the terminal.

I ran a CLI command in AWS CloudShell that created four DynamoDB tables with specific attributes and keys. Each table was provisioned with 1 read and 1 write capacity unit. The command also queried the table's status to confirm successful creation.

```
aws dynamodb create-table \
  --table-name Forum \
  --attribute-definitions \
  --key-scheme \
  --provisioned-throughput \
  --query "TableDescription.TableStatus"
"CREATING"
~$ aws dynamodb create-table \
  --table-name Post \
  --attribute-definitions \
  --key-scheme \
  --provisioned-throughput \
  --query "TableDescription.TableStatus"
"CREATING"
~$ aws dynamodb create-table \
  --table-name Comment \
  --attribute-definitions \
  --key-scheme \
  --provisioned-throughput \
  --query "TableDescription.TableStatus"
"CREATING"
```

Loading Data with CLI

I ran a CLI command in AWS CloudShell that loaded data into DynamoDB tables using the 'batch-write-item' command. The command uploaded data from JSON files (ContentCatalog.json, Forum.json, Post.json, and Comment.json) into the respective tables.

```
nextworksampdata $ aws dynamodb batch-write-item --request-items file://ContentCatalog.json
{
  "UnprocessedItems": {}
}
nextworksampdata $
nextworksampdata $ aws dynamodb batch-write-item --request-items file://Forum.json
{
  "UnprocessedItems": {}
}
nextworksampdata $
nextworksampdata $ aws dynamodb batch-write-item --request-items file://Post.json
{
  "UnprocessedItems": {}
}
nextworksampdata $
nextworksampdata $ aws dynamodb batch-write-item --request-items file://Comment.json
{
  "UnprocessedItems": {}
}
nextworksampdata $
```

Observing Item Attributes

Edit item

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)

Attributes

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

I checked a ContentCatalog item, which had the following attributes: Id(Partition key), Authors(String), ContentType(Project), Difficulty(Easy peasy), Price(Number), ProjectCategory(Storage), Published(Boolean), and Title(Host a Website on Amazon S3).

I checked another ContentCatalog item, which had a different set of attributes: Id (Partition key, Number), ContentType (Video, String), Price (0, Number), Services (List), Title (AWS x Databases project, String), URL (String), and VideoType (String)

Benefits of DynamoDB

A benefit of DynamoDB over relational databases is flexibility, because it allows each item to have its own set of attributes. Unlike relational databases, where every row must have the same columns, DynamoDB supports dynamic schemas for diverse data.

Another benefit over relational databases is speed, because DynamoDB is designed for low-latency performance at scale. Its distributed architecture enables faster read and write operations, without the need for complex joins or queries.




Items returned (6)

Actions Create item

< 1 > ⚙

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

Today you've learnt how to:

1.  **Create a DynamoDB table:** You could do this with both the AWS Management Console and AWS CLI.
2.  **Load data into DynamoDB:** Using AWS CLI, you loaded four files into matching DynamoDB tables. Such a time saver!
3.  **Edit data in your DynamoDB tables** This came with lots of learnings about the differences between DynamoDB and relational databases!