



nextwork.org

Load Data into a DynamoDB Table



ampahben3@gmail.com

Table: ContentCatalog - Items returned (6)									
Scan started on May 07, 2025, 16:25:37									
	ID (Number)	Authors	Content Type	Difficulty	Price	Project Category	Published	Services	Title
<input type="checkbox"/>	3	[{"\$": "Nextwork"}]	Project	Easy peasy	0	AI/ML	true	Build a Chatbot	Build a Chatbot
<input type="checkbox"/>	2	[{"\$": "Nextwork"}]	Project	Easy peasy	0	Analytics	true	Visualize data	Visualize data
<input type="checkbox"/>	203		Video		0			AWS x Data	AWS x Data
<input type="checkbox"/>	202		Video		0			Don't miss	Don't miss
<input type="checkbox"/>	201		Video		0			AWS Relation	AWS Relation
<input type="checkbox"/>	1	[{"\$": "Natural Language Processing"}]	Project	Easy peasy	0	Storage	true	Host a Website	Host a Website

Introducing Today's Project!

What is Amazon DynamoDB?

Amazon DynamoDB is a NoSQL database that provides fast performance, automatic scaling, and high availability. It's great for real-time applications. Your project involved loading data into DynamoDB for efficient storage and retrieval.

How I used Amazon DynamoDB in this project

Amazon DynamoDB was used to load data into tables like ContentCatalog, Forum, Post, and Comment using batch-write-item operations. This ensured efficient storage and fast retrieval while supporting scalability.

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

The data-loading process seemed like it wasn't working, and I was stuck for a long time. Later, I realized it had actually worked all along. That was unexpected

This project took me...

This project took more than an hour, mostly spent figuring out why there was no meaningful output while loading the database. I did a lot of Googling and prompting before realizing the issue.

Create a DynamoDB table

DynamoDB tables organize data using items, each with unique attributes. Unlike traditional databases, items aren't fixed in rows and columns. Instead, every item can have different attributes, making DynamoDB highly flexible for dynamic data storage.

An attribute is an item that provides additional information about something, in this case, a key-value pair in Amazon DynamoDB. Attributes define an item's properties, helping store and organize data efficiently for easy querying and management.

The screenshot shows the AWS DynamoDB console interface. On the left, there is a sidebar titled "Tables (1)" with a dropdown menu for "Any tag key" and "Any tag value". Below it is a search bar with placeholder text "Find tables" and a "NextWorkStudents" button. The main area is titled "NextWorkStudents" and contains a "Scan or query items" section. It has two tabs: "Scan" (selected) and "Query". Under "Scan", there is a "Select a table or index" dropdown set to "Table - NextWorkStudents" and a "Select attribute projection" dropdown set to "All attributes". Below these are "Filters - optional" fields and "Run" and "Reset" buttons. A green status bar at the bottom indicates "Completed - Items returned: 1 - Items scanned: 1 - Efficiency: 100% - RCU consumed: 0.5". Below this is a table titled "Table: NextWorkStudents - Items returned (1)". It shows a single row with "StudentName (String)" and "ProjectsComplete". The row data is "Nikko" and "4". There are "Actions" and "Create item" buttons at the top of the table, and navigation buttons at the bottom.

Read and Write Capacity

Read capacity units (RCUs) and write capacity units (WCUs) are Amazon DynamoDB metrics defining database throughput. RCUs measure reads per second, while WCUs determine writes. These units help optimize performance and manage costs efficiently.

Amazon DynamoDB's Free Tier covers 25 RCUs, 25 WCUs, and 25GB of storage per month. I turned off auto scaling to prevent unexpected costs, as it automatically adjusts throughput. Managing capacity manually keeps expenses low.

Read/write capacity settings [Info](#)

Capacity mode

On-demand Simplify billing by paying for the actual reads and writes your application performs.

Provisioned Manage and optimize your costs by allocating read/write capacity in advance.

Read capacity

[Auto scaling](#) | [Info](#)
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.

On

Off

Provisioned capacity units

Write capacity

[Auto scaling](#) | [Info](#)
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.

On

Off

Minimum capacity units

Maximum capacity units

Target utilization (%)

Using CLI and CloudShell

AWS CloudShell is an environment that uses commands to help perform tasks like accessing resources in AWS. It provides a browser-based terminal for running AWS CLI commands, making it easy to manage services without installing tools locally.

AWS CLI is a command-line tool for managing AWS services. It allows users to run commands to configure, control, and automate resources without using a graphical interface, making cloud management more efficient and flexible.

I ran a CLI command in AWS CloudShell that created four tables: ContentCatalog (Id attribute), Forum (partition key Name), Post (partition key ForumName, sort key Subject), and Comment (partition key Id, sort key CommentDateTime).

```
$ aws dynamodb create-table \
>   --table-name contentcatalog \
>   --attribute-definitions \
>     AttributeDefinition \
>       AttributeName=Id,AttributeType=N \
>   --key-schema \
>     KeySchemaElement \
>      AttributeName=Id,KeyType=HASH \
>   --provisioned-throughput \
>     ReadCapacityUnits=1,WriteCapacityUnits=1 \
> $ aws dynamodb create-table \
>   --table-name contentcatalog \
>   --attribute-definitions \
>     AttributeDefinition \
>       AttributeName=Id,AttributeType=N \
>   --key-schema \
>     KeySchemaElement \
>      AttributeName=Name,KeyType=PARTITION \
>   --provisioned-throughput \
>     ReadCapacityUnits=1,WriteCapacityUnits=1 \
>   --query "TableDescription.tableStatus"
"CREATING"
$ aws dynamodb create-table \
>   --table-name forum \
>   --attribute-definitions \
>     AttributeDefinition \
>       AttributeName=name,AttributeType=S \
>   --key-schema \
>     KeySchemaElement \
>      AttributeName=name,KeyType=PARTITION \
>   --provisioned-throughput \
>     ReadCapacityUnits=1,WriteCapacityUnits=1 \
>   --query "TableDescription.tableStatus"
"CREATING"
$ aws dynamodb create-table \
>   --table-name forum \
>   --attribute-definitions \
>     AttributeDefinition \
>       AttributeName=name,AttributeType=S \
>   --key-schema \
>     KeySchemaElement \
>      AttributeName=name,KeyType=PARTITION \
>   --provisioned-throughput \
>     ReadCapacityUnits=1,WriteCapacityUnits=1 \
>   --query "TableDescription.tableStatus"
```

Loading Data with CLI

I ran a CLI(Command Line Interface) command in AWS CloudShell that executed multiple batch-write-item operations to insert data into ContentCatalog, Forum, Post, and Comment tables in DynamoDB. These commands were run twice to ensure data insertion.

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

Observing Item Attributes

The screenshot shows a table-based interface for managing item attributes. The columns are labeled 'Attribute name', 'Value', and 'Type'. There is also a header row with 'Add new attribute' and a 'Remove' button.

Attribute name	Value	Type	Add new attribute
Id - Partition key	3	Number	Remove
Authors	<input type="button" value="Insert a field"/>	List	Remove
ContentType	Project	String	Remove
Difficulty	Easy peasy	String	Remove
Price	0	Number	Remove
ProjectCategory	AI/ML	String	Remove
Published	<input checked="" type="radio"/> True <input type="radio"/> False	Boolean	Remove
Title	Build a Chatbot with Amazon Lex	String	Remove
URL	aws-ai-lex1	String	Remove

Buttons at the bottom include 'Cancel', 'Save', and 'Save and close'.

I checked a ContentCatalog item, which had the following attributes: Id, Authors, ContentType, Difficulty, Price, ProjectCategory, Published, Title, and URL. It describes a project titled Build a Chatbot with Amazon Lex, categorized under AI/ML.

I checked another ContentCatalog item, which had a different set of attributes: Id, Authors, ContentType, Duration, Price, Published, Title, and URL. It describes a video resource with specific metadata, including its duration and publication status.

Benefits of DynamoDB

DynamoDB allows items to have unique attributes, unlike relational databases where all rows must follow the same schema. It also uses partition keys for fast retrieval, while relational databases scan entire tables, which can slow performance.

Another benefit over relational databases is speed because DynamoDB uses partition keys to quickly locate data, while relational databases must scan entire tables, slowing performance.

Table: ContentCatalog - Items returned (6)								
Scan started on May 07, 2025, 16:25:37								
	Id (Number)	Authors	ContentType	Difficulty	Price	ProjectCategory	Published	Services
<input type="checkbox"/>	3	[{"S": "Nex..."}]	Project	Easy peasy	0	AI/ML	true	Build a Cha...
<input type="checkbox"/>	2	[{"S": "Nex..."}]	Project	Easy peasy	0	Analytics	true	Visualize da...
<input type="checkbox"/>	203		Video		0			[{"S": "Am..."}]
<input type="checkbox"/>	202		Video		0			Don't miss ...
<input type="checkbox"/>	201		Video		0			[{"S": "Am..."}]
<input type="checkbox"/>	1	[{"S": "Nat..."}]	Project	Easy peasy	0	Storage	true	Host a Web...



nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects

