

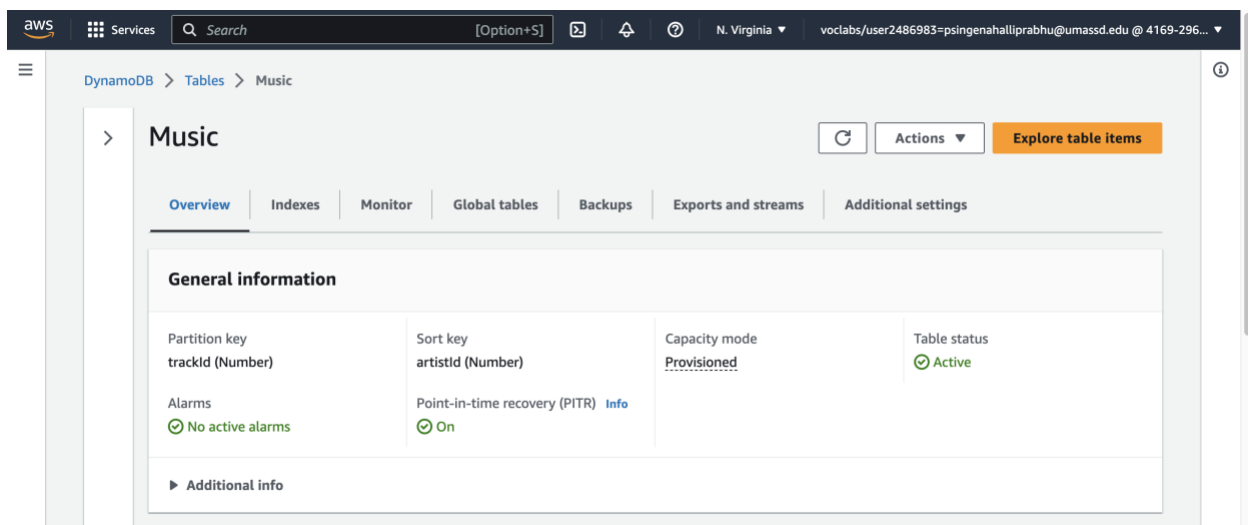
# CIS 552: Database Design – Homework 5

## Introduction:

I successfully created an AWS Academy for Students account and set up a DynamoDB database in the AWS cloud. To store my music-related data, I created a table named "Music" with a well-designed schema.

## Table Creation:

The "Music" table has a partition key named "trackId," which serves as the primary key for the table. Additionally, I added a Sort Key called "artistId" to the table, which allows for efficient querying of data related to specific artists.



## Populating table:

I leveraged the publicly available iTunes Search API provided by Apple Computer to retrieve relevant data for populating my database. This API provides a comprehensive set of endpoints for searching and retrieving music-related data, including track names, artist names, album names, and other metadata.

By utilizing this API, I was able to efficiently gather large amounts of data from the iTunes Store, and easily integrate it into my database. This approach allowed me to ensure the accuracy and consistency of the data, while also minimizing the time and effort required to manually enter the information.

I made sure to create items with varying fields to take advantage of the inherent flexibility and irregularity that non-SQL data and databases provide.

aws Services Search [Option+S] N. Virginia voclabs/user2486983=psingenahalliprabhu@umassd.edu @ 4169-296...

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** View DynamoDB JSON

```
1 {
2   "trackId": {
3     "N": "1677443584"
4   },
5   "artistId": {
6     "N": "909253"
7   },
8   "artistName": {
9     "S": "Jack Johnson"
10  },
11  "artistViewUrl": {
12    "S": "https://music.apple.com/us/artist/jack-johnson/909253?uo=4"
13  },
14  "artworkUrl100": {
15    "S": "https://is1-ssl.mzstatic.com/image/thumb/Music126/v4/04/ec/48/04ec488c-104f-aefb-b48f-6078d0511a79/23UMGIM28520.rgb.jpg/100x100bb.jpg"
16  },
17  "artworkUrl30": {
18    "S": "https://is1-ssl.mzstatic.com/image/thumb/Music126/v4/04/ec/48/04ec488c-104f-aefb-b48f-6078d0511a79/23UMGIM28520.rgb.jpg/30x30bb.jpg"
19  },
20  "artworkUrl60": {
21    "S": "https://is1-ssl.mzstatic.com/image/thumb/Music126/v4/04/ec/48/04ec488c-104f-aefb-b48f-6078d0511a79/23UMGIM28520.rgb.jpg/60x60bb.jpg"
22  }
23 }
```

JSON Ln 1, Col 1 Errors: 0 Warnings: 0

Cancel Save changes

CloudShell Feedback Language © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

We can see from the below image, that the date is saved in the table.

aws Services Search [Option+S] N. Virginia voclabs/user2486983=psingenahalliprabhu@umassd.edu @ 4169-296...

**Items returned (10)** Actions Create item

	trackId	artistId	amgArtistId	artistLinkUrl	artistName	artistType	ar
<input type="checkbox"/>	1677443584	909253			Jack Johnson		ht
<input type="checkbox"/>	986903480	961252454			Leon Bridges		ht
<input type="checkbox"/>	528437613	148662			LINKIN PARK		ht
<input type="checkbox"/>	283635002	203739			Brad Paisley		ht
<input type="checkbox"/>	36130192232	78500	5723	https://music.ap...	U2	Artist	
<input type="checkbox"/>	573962555	278873078			Bruno Mars		ht
<input type="checkbox"/>	1663974121	137057909			Miley Cyrus		ht
<input type="checkbox"/>	1079234240	961252454			Leon Bridges		ht
<input type="checkbox"/>	12345678	909253	468749	https://music.ap...	Jack Johnson	Artist	
<input type="checkbox"/>	1663973562	137057909			Miley Cyrus		ht

CloudShell Feedback Language © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Query Data:

To retrieve specific tracks from my DynamoDB database, I utilized the trackId field as the partition key for my table. By performing queries using this key, I was able to retrieve precise data from the database with minimal latency.

The following image illustrates the results of a typical query using trackId as the partition key:

The screenshot shows the AWS Management Console interface for a DynamoDB query. The 'Query' tab is selected, and the table 'Table - Music' is chosen. The partition key 'trackId' is set to '1663973562'. The attribute projection is set to 'All attributes'. The query is executed, and the results are displayed in a table with 7 columns: trackId, artistId, artistName, artistViewUrl, artworkUrl100, and artworkUrl130. The results show a single item for trackId '1663973562' by artist 'Miley Cyrus'.

trackId	artistId	artistName	artistViewUrl	artworkUrl100	artworkUrl130
1663973562	137057909	Miley Cyrus	https://music.ap...	https://is4-ssl.m...	https://is4-ssl.m...

## Creating a Secondary Index:

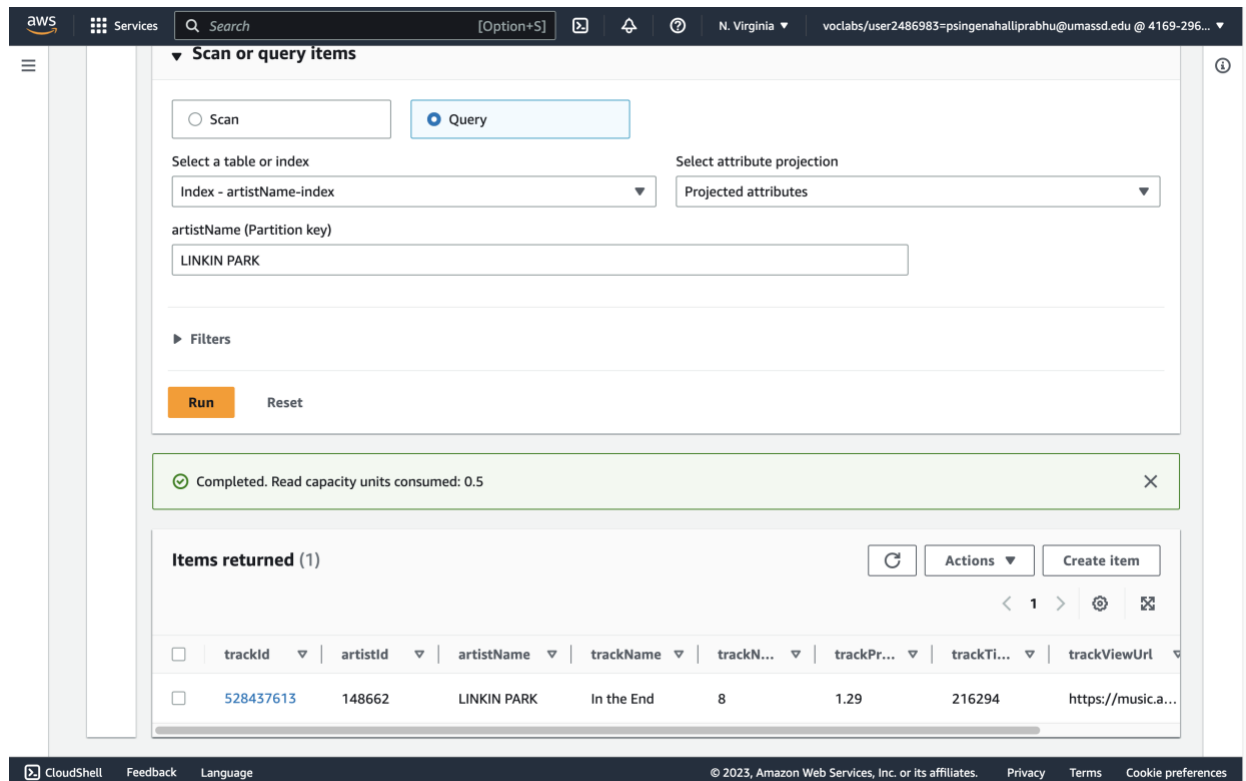
As part of my DynamoDB implementation, I recognized the need for a secondary index that would enable me to efficiently query data based on artist names. To address this requirement, I created a secondary index on the "artistName" field, which allows me to quickly retrieve all tracks associated with a particular artist.

The screenshot shows the AWS Management Console interface for the 'Music' table. The 'Indexes' tab is selected, and the 'Global secondary indexes' section is displayed. A table lists the existing secondary index 'artistName-index', which is active and has a partition key of 'artistName (String)'. The table also shows the read and write capacity for the index.

Name	Status	Partition key	Sort key	Read capacity	Write capacity	Projected attributes
artistName-index	Active	artistName (String)	-	5 Auto scaling is off	5 Auto scaling is off	All

## Querying using Secondary Index

With the addition of a secondary index on the "artistName" field in my DynamoDB database, I am now able to easily query data based on specific artist names.



The screenshot displays the AWS Management Console interface for querying a DynamoDB database. The 'Scan or query items' section is active, with the 'Query' tab selected. The 'Select a table or index' dropdown is set to 'Index - artistName-index', and the 'Select attribute projection' dropdown is set to 'Projected attributes'. The 'artistName (Partition key)' field contains the value 'LINKIN PARK'. The 'Run' button is highlighted in orange, and a 'Reset' button is also visible. Below the query configuration, a green status bar indicates 'Completed. Read capacity units consumed: 0.5'. Underneath, the 'Items returned (1)' section shows a table with the following data:

trackId	artistId	artistName	trackName	trackN...	trackPr...	trackTi...	trackViewUrl
528437613	148662	LINKIN PARK	In the End	8	1.29	216294	https://music.a...

## Conclusion

In conclusion, my assignment has successfully utilized DynamoDB, a NoSQL database provided by AWS, to store and retrieve music-related data. Through the use of efficient querying techniques and secondary indexes, I have been able to effectively manage a large and diverse dataset, and easily retrieve information based on various criteria such as track ID and artist name. The use of the publicly available iTunes Search API has also proved to be an effective means of populating the database with accurate and comprehensive data. Overall, this assignment has provided me with valuable experience working with cloud-based databases and exploring the unique capabilities and challenges of NoSQL data management.