Exercise: DynamoDB - Typescript

In this exercise you will:

1. Create a DynamoDB table named “follows” that keeps track of which users are following each other (as in Twitter).
2. Create a secondary index named “follows_index” on the “follows” table.
3. Write a program that uses the AWS SDK to:
    a. Insert, update, and delete items in the “follows” table
    b. Query the “follows” table with and without pagination
    c. Query the “follows_index” index with and without pagination

Assumptions

You have already configured your AWS credentials on your computer. This should have been done when you installed the AWS CLI on your computer.

Node.js and ts-node are installed on your computer.

Steps

1. Go the the AWS DynamoDB web console

2. Create a table named “follows”.
    a. Partition Key: a string named “follower_handle”. This attribute stores the Twitter handle of the user who is following another user.
    b. Sort Key: a string named “followee_handle”. This attribute stores the Twitter handle of the user who is being followed.
    c. This Primary Key lets you query the table by “follower_handle” and sort the results by “followee_handle”.

3. Select the “Indexes” tab and create an index named “follows_index”.
    a. Partition Key: a string named “followee_handle. This is the same “followee_handle” attribute you created in the previous step.
    b. Check “Add sort key”
    c. Sort Key: a string named “follower_handle”. This is the same “follower_handle” attribute you created in the previous step.
    d. This Primary Key lets you query the index by “followee_handle” and sort the results by “follower_handle”.
    e. For “Projected attributes”, keep the “All” setting.

4. Next, you will write a Typescript program that performs operations on the “follows” table.
5. Create and initialize a new Node.js Typescript project

6. Install the AWS SDK
   > npm install --save aws-sdk

7. Write Typescript code to put, get, update, and delete items in the "follows" table. The examples on this page demonstrate how to put, get, update, and delete items in a DynamoDB table: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Js.03.html. Also, the complete online documentation for the AWS Javascript SDK can be found here: https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/

   a. "Put" 100 items into the "follows" table with the following attributes:
      i. "follower_handle": string handle of the user who is following (e.g., "@FredFlintstone")
      ii. "follower_name": string name of the user who is following (e.g., "Fred Flintstone")
      iii. "followee_handle": string handle of the user being followed (e.g., "@ClintEastwood")
      iv. "followee_name": string name of the user being followed (e.g., "Clint Eastwood")
      v. In the items that you create, make sure that each user is following many other users, and that each user is being followed by many other users.

   b. "Get" one of the items from the "follows" table using its primary key

   c. "Update" the "follower_name" and "followee_name" attributes of one of the items in the "follows" table

   d. "Delete" one of the items in the "follows" table using its primary key

8. Write Typescript code to query the "follows" table and the "follows_index" index. The example on this page demonstrates how to query a table: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Js.04.html.

   a. "Query" the "follows" table to return all of the users being followed by a user, sorted by "followee_handle"
      i. Hint: Use the ScanIndexForward: true query param to sort the result in ascending order
      ii. Hint: You might need to ensure your ExpressionAttributeValues are in the correct format like so: ExpressionAttributeValues: {<Property>:{<Type Key>:<Value>} } if typescript complains about your params argument.

b. “Query” the “follows_index” index to return all of the users following a user, reverse sorted by “follower_handle”
    i. Hint: Use the IndexName: ‘follows_index’ query param to specify the name of the index to query (in addition to specifying the TableName: ‘follows’ query param)
    ii. Hint: Use the ScanIndexForward: false query param to sort the result in descending order

9. Make copies of the previous two queries, and modify them to return the query results in pages of size ten (i.e., ten at a time)
    a. This means you will need to re-execute the query multiple times in order to fetch all of the results.
    b. Hint: Use the Limit query param to indicate how many items should be returned in each page
    c. Hint: Use the ExclusiveStartKey query param to tell DynamoDB the primary key value of the last item returned in the previous page (this is not necessary for the first page). This is how DynamoDB knows where the next page should start.
    d. Hint: The LastEvaluatedKey attribute of the query result contains the primary key value for the last item returned in the current page. If it’s undefined, the end of the query result has been reached.

10. Submit your Typescript code through Canvas.

More Information

The full AWS SDK documentation can be found here in the SDKs section:
https://aws.amazon.com/tools/

Amazon provides a way to automatically serialize and deserialize the data in the DynamoDB format (with the data type serialize) through the Converter:
https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/DynamoDB/Converter.html