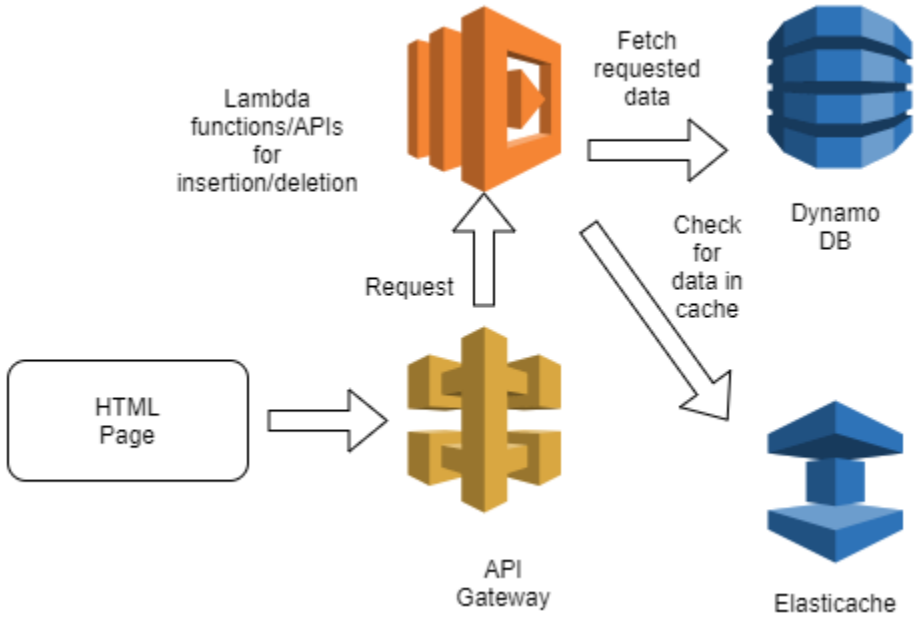
APIs for insertion/deletion and viewing can be implemented using serverless architecture, i.e., using AWS Lambda functions. Here, we won't require an application server to run our web application; in fact, our client application will comprise of a single HTML page.

Individual functions carrying out the insertion/deletion operations are deployed as Lambda functions. The API gateway service is used to send requests to the deployed functions.

DynamoDB enables storage of data in key value format, hence adding flexibility to the logical structure of the data. It scales well and hence has good query speeds.

However, considering a scenario where there are millions of requests to be serviced, a fractional reduction in the fetching time would mean a significant optimization in the running time.



Languages/Technologies Used:

Python, HTML5, jQuery

AWS:

1. Lambda
2. DynamoDB
3. Elasticache Redis
4. API Gateway

Python Libraries:

1. Boto3
2. Hashlib

Environment required/setup:

Simply run the HTML page on a modern browser such as chrome/Mozilla etc. You can use the interface to test the various functions of the API.

Structure of Data:

The restaurant data can be represented as following:

1. ID
2. Name
3. Location
4. Zip

Here, I have used the combination of name, location and zip to produce a unique hash and used it as an ID of the restaurant.

As DynamoDB is a key-value store, ID is used as the partition key and rest of the data is used as the 'Value' for that partition key.

Similarly, Menu items can be represented as:

1. ID
2. Name
3. Price
4. Restaurant ID

Components of the API:

The **Insert** component:

This is a basic component for inserting new restaurants or dishes. It simply takes all the details from the user and pushes the data to DynamoDB.

Parameters (required):

a. Name
b. Location
c. Zip

URL: https://dbws4i6jxd.execute-api.us-east-2.amazonaws.com/v1/insert

The **Search** Component:

This API is at the heart of most functionalities that the API hopes to fulfill. Here, the user can search a restaurant or menu item by their ID, or other attributes such as location, name etc.

Elasticache Redis has been used as the caching layer above DynamoDB:

a. Every time a search is made by the user, the cache is first checked for any match with the search query.
b. If no match is found, then DynamoDB is queried and the resulting set of values and pushed onto the cache memory to speed up future operations

Parameters:

a. Name
   OR
b. Location/Restaurant ID if it's a menu item
   OR
c. ID

URL: https://dbws4i6jxd.execute-api.us-east-2.amazonaws.com/v1/search

The **Delete** Component:

1. The delete component gives the user a choice to delete single restaurant or menu items by ID.
2. The user can also perform a search of tuples having certain attribute values.
3. For menu items, deletion has to be done using the id of the item so as to avoid wrong deletions from the table.

Parameters:

a. Name
   OR
b. Location/Restaurant ID if it's a menu item
   OR
c. ID

URL: https://dbws4i6jxd.execute-api.us-east-2.amazonaws.com/v1/delete

Scenario:

1. Suppose, the user wants to delete all restaurants in Brooklyn, then he/she can simply enter the location.
2. The Search component will return all elements matching with the location. The user can then select a batch of items to be deleted from the table.
3. The items are first flushed from the cache and later deleted from DynamoDB.

Additional Possible Developments:

We can add a MySQL database which is updated at regular intervals from the DynamoDB.