**My Notes & References**

**New Dataset**

It consists of nearly 47k restaurants. Each restaurants having following 10 attributes.

|  |  |
| --- | --- |
| id | Unique |
| name |  |
| rating |  |
| Rating count |  |
| cost |  |
| address |  |
| cuisine |  |
| License no | Not unique |
| link | Unique |
| city |  |

menu – Removed attribute

**Example:**

data = {

"name": "AB FOODS POINT",

"rating": "--",

"rating\_count": "Too Few Ratings",

"cost": "\u20b9 200",

"address": "AB FOODS POINT, NEAR RISHI NARANG DENTAL CLINIC , NEAR IDBI BANK, ABOHAR",

"cuisine": "Beverages,Pizzas",

"lic\_no": "22122652000138",

"link": "https://www.swiggy.com/restaurants/ab-foods-point-central-abohar-abohar-567335",

"city": "Abohar",

"id": "567335"

}

**Pagination:**

|  |  |  |
| --- | --- | --- |
| Previous | 1 2 3 4 5 | Next |

|  |  |  |
| --- | --- | --- |
| Previous | 4 | Next |

|  |
| --- |
| Load More.. |

**Three Ways of Data Retrieval From DynamoDB:**

Scan - Useful when notknowing the specific partition key in advance.

Query **-** when you know the partition key and potentially the sort key for the items you want to retrieve.

get\_item **-** operation is designed to retrieve a single item from a table based on its primary key

**Design 1**

Partition Key – ID - Unique

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Address | City | Link |
| 84681 | Restaurant A |  | Abohar | Link1 |
| 79214 | Restaurant B |  | Coimbatore | Link2 |
| 70404 | Restaurant C |  | chennai | Link3 |
| 561629 | Restaurant D |  | bangalore | Link4 |
| 556095 | Restaurant E |  | Goa | Link5 |
| 553626 | Restaurant F |  | Tirupati | Link6 |
| 548372 | Restaurant G |  | Sivakasi | Link7 |

Query – no need to use exclusive start key because each partition has only one item.

Scan – can use exclusive start key when partition key is unknown.

Problems:

If we use this design, then we can’t paginate to previous page because this design doesn’t have sort key.

We can only use scan to get items. Query is not effective for this.

ID is Unique but not ordered.

Solutions:

1. All Ids should be stored in somewhere like Cache System or other table.
2. Should create secondary index for different access patterns.
3. Need to scan all items then we can do pagination but not cost effective.
4. Should change Ids in incremental Order.

list = [id1, id2, id3, id4..............]

**Design 2**

Partition Key – Link - Unique

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Link | Name | Address | City | ID |
| Link1 | Restaurant A |  | Abohar | 84681 |
| Link2 | Restaurant B |  | Coimbatore | 79214 |
| Link3 | Restaurant C |  | chennai | 70404 |
| Link4 | Restaurant D |  | bangalore | 561629 |
| Link5 | Restaurant E |  | goa | 556095 |
| Link6 | Restaurant F |  | Tirupati | 553626 |
| Link7 | Restaurant G |  | Sivakasi | 548372 |

Problems: This Design is not suitable for our requirements

**Design 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| City | ID | Link | Name | Address |
| Abohar | 84681 | Link1 | Restaurant A |  |
| 79214 | Link2 | Restaurant B |  |
| 70404 | Link3 | Restaurant C |  |
| 561629 | Link4 | Restaurant D |  |
| 556095 | Link5 | Restaurant E |  |
| 553626 | Link6 | Restaurant F |  |
| 548372 | Link7 | Restaurant G |  |
| Coimbatore | 84681 | Link1.1 | Restaurant A |  |
| 79214 | Link1.2 | Restaurant B |  |
| 70404 | Link1.3 | Restaurant C |  |
| 561629 | Link1.4 | Restaurant D |  |
| 556095 | Link1.5 | Restaurant E |  |
| 553626 | Link1.6 | Restaurant F |  |
| 548372 | Link1.7 | Restaurant G |  |

Partition Key – City

Sort Key – ID – Unique

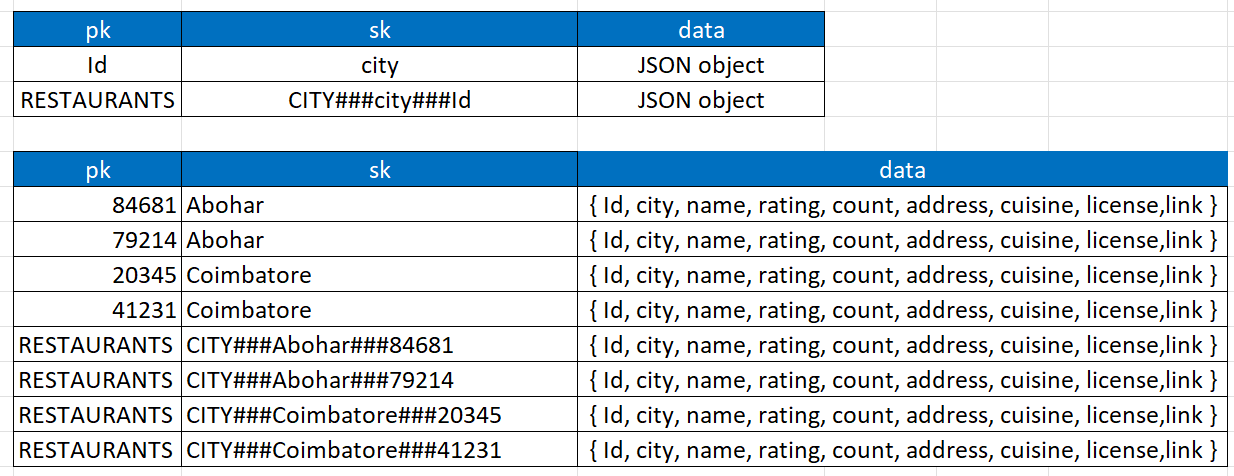
Problems:

1. Need to get specific Location first, then we can find, Update, Delete Restautrant by ID
2. Can create secondary index (id) to Update, Delete Restautrant by ID.

Advantages:

* Can use LastEvaluatedKey in Query.
* Can use ScanIndexForward – This design table have a sort key defined.
* Can easily navigate to previous and next pages.

**DESIGN 4**



PK – ID

SK – City

**Advantage:** Read Optimized

**Pagination:**

Page 1 page2 page3

|  |  |  |
| --- | --- | --- |
| previous – ‘’ | previous-156601 | previous-158203 |
|  |  |  |
| 156542 | 156601 | 158203 |
|  |  |  |
| 156587 | 156602 | 158204 |
|  |  |  |
| 156588 | 158192 | 161396 |
|  |  |  |
| 156590 | 158193 | 161405 |
|  |  |  |
| 156592 | 158195 | 161408 |
|  |  |  |
| next-156592 | next-158195 | next-161408 |

**Index**

An index is a data structure that allows you to query and access your data more efficiently based on specific attributes other than the table's primary key. DynamoDB supports two types of indexes:

**Global Secondary Index (GSI)** – Different Primary Key [Partition optional Sort Key]

**Local Secondary Index (LSI)** – Same Partition key but different sort key

Indexes in DynamoDB are used to improve query performance and enable more flexible access patterns for your data.

**Result:**

Publicly Accessible Urls:

**Paginations:**

* <http://ec2-13-51-165-179.eu-north-1.compute.amazonaws.com/api/restaurants>
* <http://ec2-13-51-165-179.eu-north-1.compute.amazonaws.com/api/restaurants?limit=5000>
* <http://ec2-13-51-165-179.eu-north-1.compute.amazonaws.com/api/restaurants?limit=5000&city=Agra>

**GET:**

* <http://ec2-13-51-165-179.eu-north-1.compute.amazonaws.com/api/restaurants/156588>
* <http://ec2-13-51-165-179.eu-north-1.compute.amazonaws.com/api/restaurants/156590>

**POST** - To create an item

* <http://ec2-13-51-165-179.eu-north-1.compute.amazonaws.com/api/restaurants/1>

Req Body:

{

"city": "coimbatore",

"name":"Restaurant Coimbatore",

"address": "Thadagam Road, Coimbatore",

"cost":"₹ 200",

"cuisine":"Indian",

"lic\_no":"Issued",

"link":"Not yet",

"rating": "5",

"rating\_count":"100"

}

**PUT** - Update an item

* http://ec2-13-51-165-179.eu-north-1.compute.amazonaws.com/api/restaurants/1

Req Body:

{

"city": "coimbatore",

"name":"Restaurant Updated"

}

**DELETE:**

* <http://ec2-13-51-165-179.eu-north-1.compute.amazonaws.com/api/restaurants/1>

Github: <https://github.com/girijeswaran/flask_api>

**References:**

1. [**https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Query.Pagination.html**](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Query.Pagination.html)
2. <https://dynobase.dev/dynamodb-pagination/>
3. <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html>
4. <https://medium.com/tilicholabs/dynamodb-pagination-7b9742d1078a>