AWS Solutions Architect Associate

Session 901

Database: DynamoDB and Computing: Lambda

August/2024

SERVERLESS !!!

DynamoDB


Amazon DynamoDB


Amazon DynamoDB Accelerator (DAX)


Attribute


Items


Table

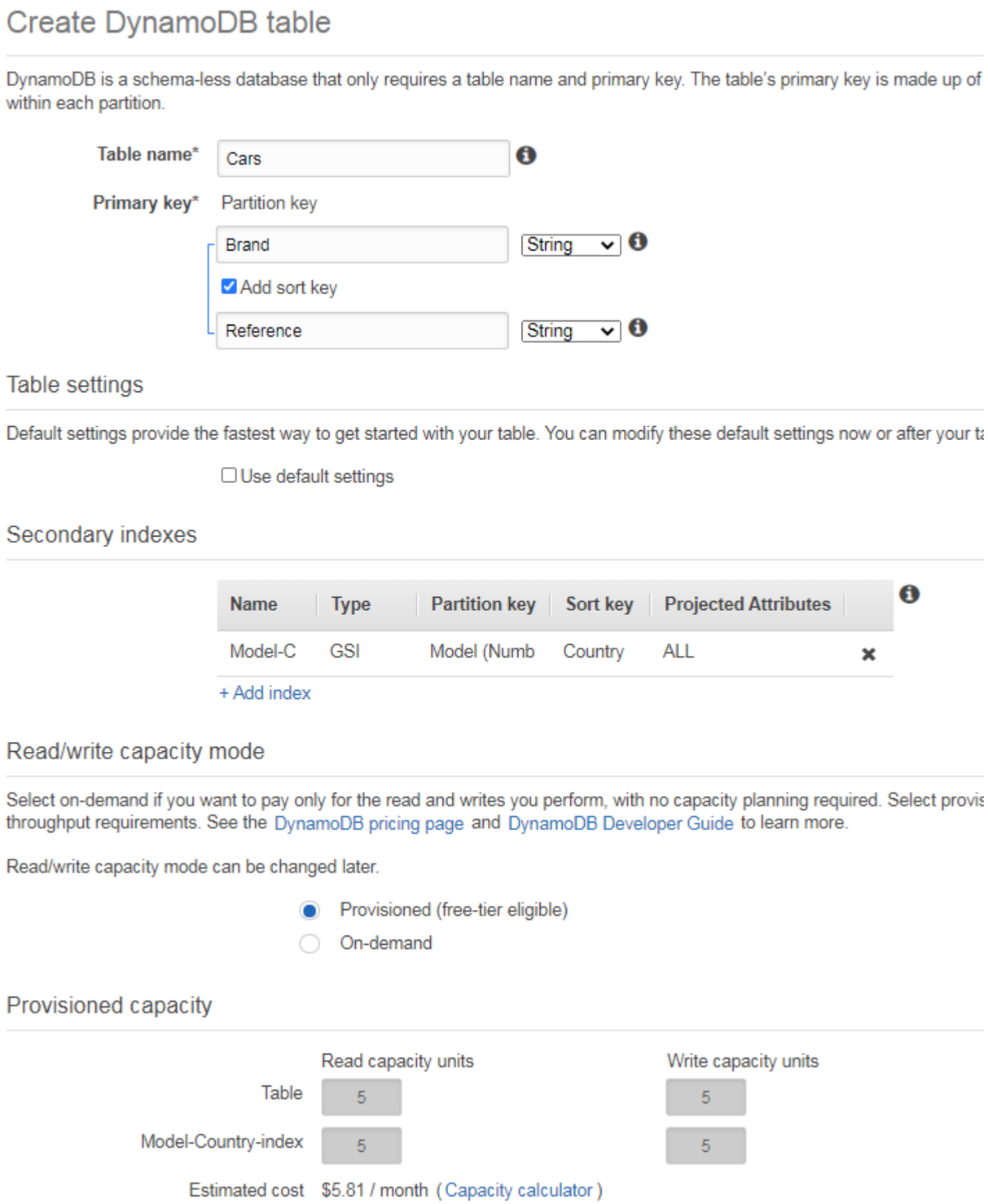

Global secondary index

- Full managed service for NoSQL DB.
- Better performance with simple/seamless scalability.
- ACID Transactions (Double Trx – One for preparing, other for commit).
- Encryption-at-rest. Options: DynamoDB/KMS/KMS-C.
- On-Demand Backup and Point-In-Time Recovery (PITR Continuous Backup up to 35 days). Cross-Region Table from Backup.
- Automatically delete records if you use TTL.
- HA using distributed SSD on Single Region (3 AZ) or you can use Global Tables.
- Usage Cases: Mobile, IoT, Web, Gaming, Session Mgmt., Realtime.

Taken from https://cloudacademy.com/course/amazon-dynamodb-high-availability/aws-dynamodb-ha-options/ (09/08/2020)

At creation moment, you have to define capacity (per second):
Read Capacity Units / RCU - Min item size for pricing 4 Kb
Write Capacity Units / WCU - Min item size for pricing 1 Kb
Primary Keys (**Partition Key** as mandatory)
Indexes (Optional)
Provisioned / On-Demand capacity Mode – Specifying pricing fixed or pay-as-you-go. Change every 24 hours, take some minutes.

Autoscaling: Take care of costs.



Consistency and Pricing

Music


```
{
    "Artist": "No One You Know",
    "SongTitle": "My Dog Spot",
    "AlbumTitle": "Hey Now",
    "Price": 1.98,
    "Genre": "Country",
    "CriticRating": 8.4
}
```

```
{
    "Artist": "No One You Know",
    "SongTitle": "Somewhere Down The Road",
    "AlbumTitle": "Somewhat Famous",
    "Genre": "Country",
    "CriticRating": 8.4,
    "Year": 1984
}
```

```
{
    "Artist": "The Acme Band",
    "SongTitle": "Still in Love",
    "AlbumTitle": "The Buck Starts Here",
    "Price": 2.47,
    "Genre": "Rock",
    "PromotionInfo": {
        "RadioStationsPlaying": [
            "KHCR",
            "KQBX",
            "WTNR",
            "WJJH"
        ],
        "TourDates": {
            "Seattle": "20150625",
            "Cleveland": "20150630"
        },
        "Rotation": "Heavy"
    }
}
```

```
{
    "Artist": "The Acme Band",
    "SongTitle": "Look Out, World",
    "AlbumTitle": "The Buck Starts Here",
    "Price": 0.99,
    "Genre": "Rock"
}
```


Read Consistency (Item Size 4 Kb):
Eventually Consistent Reads / Strongly Consistent Reads / Transactional (ACID – Nov/2018)
2 RCU per Transactional Read, 1 RCU per Strong CR, 0.5 RCU per Eventually CR
Issues with SCR: HTTP 500 Errors (Possible due to network), Latency, not Secondary Indexes.

Write Consistency (Item Size 1 Kb):
Standard / Transactional Writing.
1 WCU per Standard Writing and 2 WCU per Transactional Writing

If you don't have Autoscaling or OnDemand Capacity Mode, you get HTTP 400 Error Code (ProvisionedThroughputExceededException).
You can use locally for development and then, move to AWS.
NoSQL Workbench

- Perform strongly consistent reads of up to 24 KB per second (4 KB × 6 read capacity units)
- Perform eventually consistent reads of up to 48 KB per second (twice as much read throug
- Perform transactional read requests of up to 12 KB per second.
- Write up to 6 KB per second (1 KB × 6 write capacity units).
- Perform transactional write requests of up to 3 KB per second.

Service Limits:
32 Nested Attributes Deep
256 Tables per account
20 Global Secondary Index (soft) and 5 local Secondary Index
1 MB Per call on Query and Scan API
40k WRU and 40k RCU per table, and 80k WRU and 80k RCU per account
Provisioned, Unlimited per OnDemand Mode.

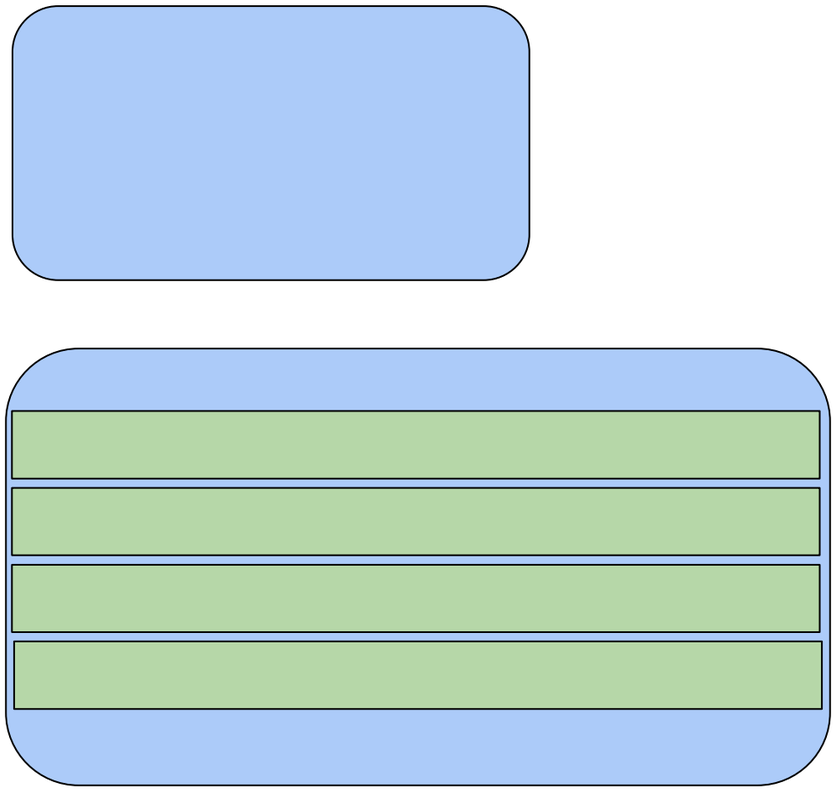
Taken from https://aws.amazon.com/blogs/aws/new-amazon-dynamodb-transactions/ (30/07/2024), https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html and https://tutorialsdojo.com/calculating-the-required-read-and-write-capacity-unit-for-your-dynamodb-table/ (30/07/2024)

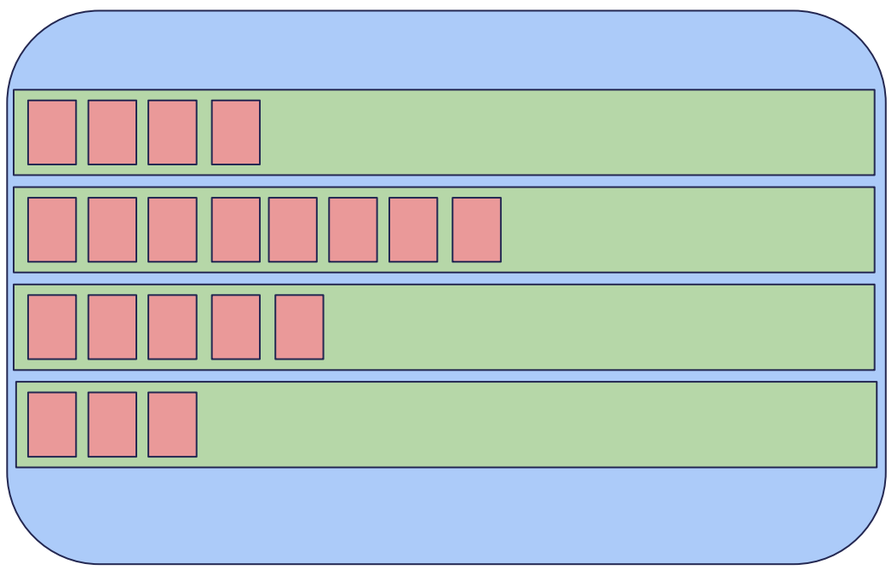
Key Concepts

Table

Items:
set of attributes

Attributes:
set of key-value



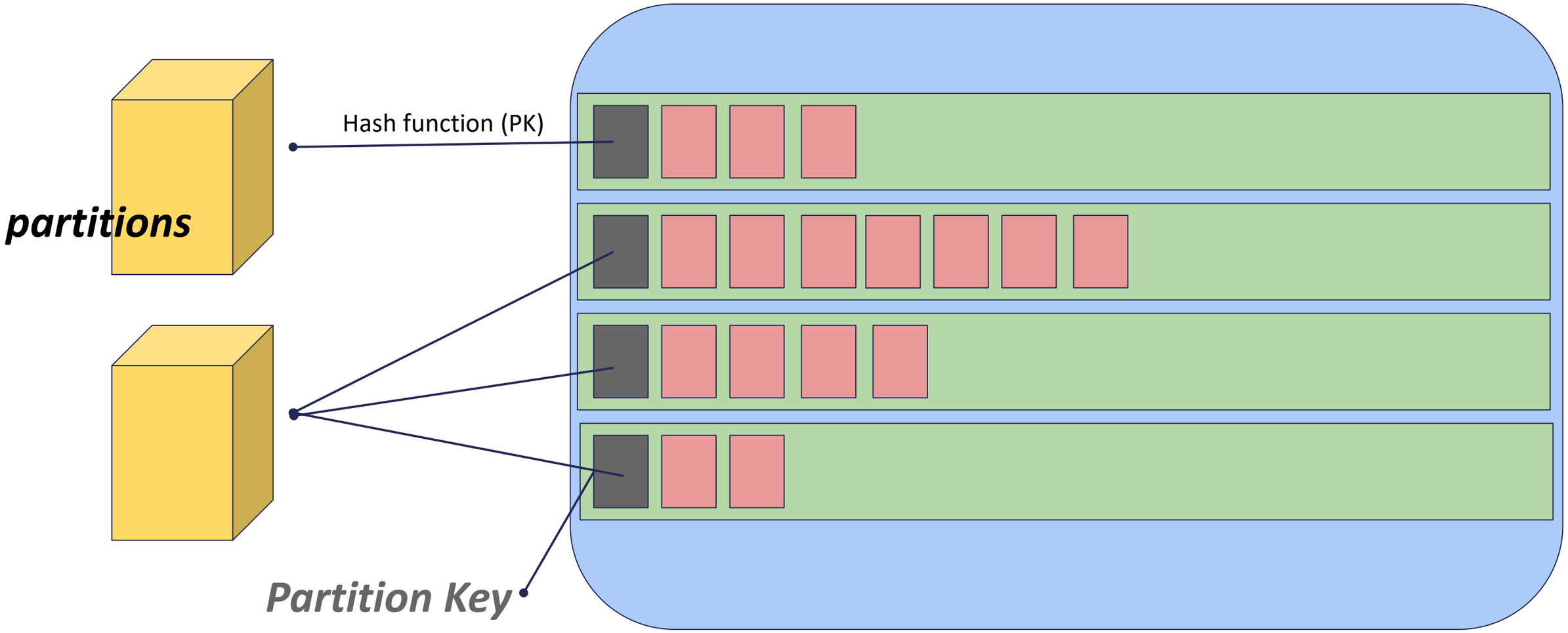



```
{
    "PersonID": 101,
    "LastName": "Smith",
    "FirstName": "Fred",
    "Phone": "555-4321"
}
```

```
{
    "PersonID": 102,
    "LastName": "Jones",
    "FirstName": "Mary",
    "Address": {
        "Street": "123 Main",
        "City": "Anytown",
        "State": "OH",
        "ZIPCode": 12345
    }
}
```

```
{
    "PersonID": 103,
    "LastName": "Stephens",
    "FirstName": "Howard",
    "Address": {
        "Street": "123 Main",
        "City": "London",
        "PostalCode": "ER3 5K8"
    },
    "FavoriteColor": "Blue"
}
```

Key Concepts



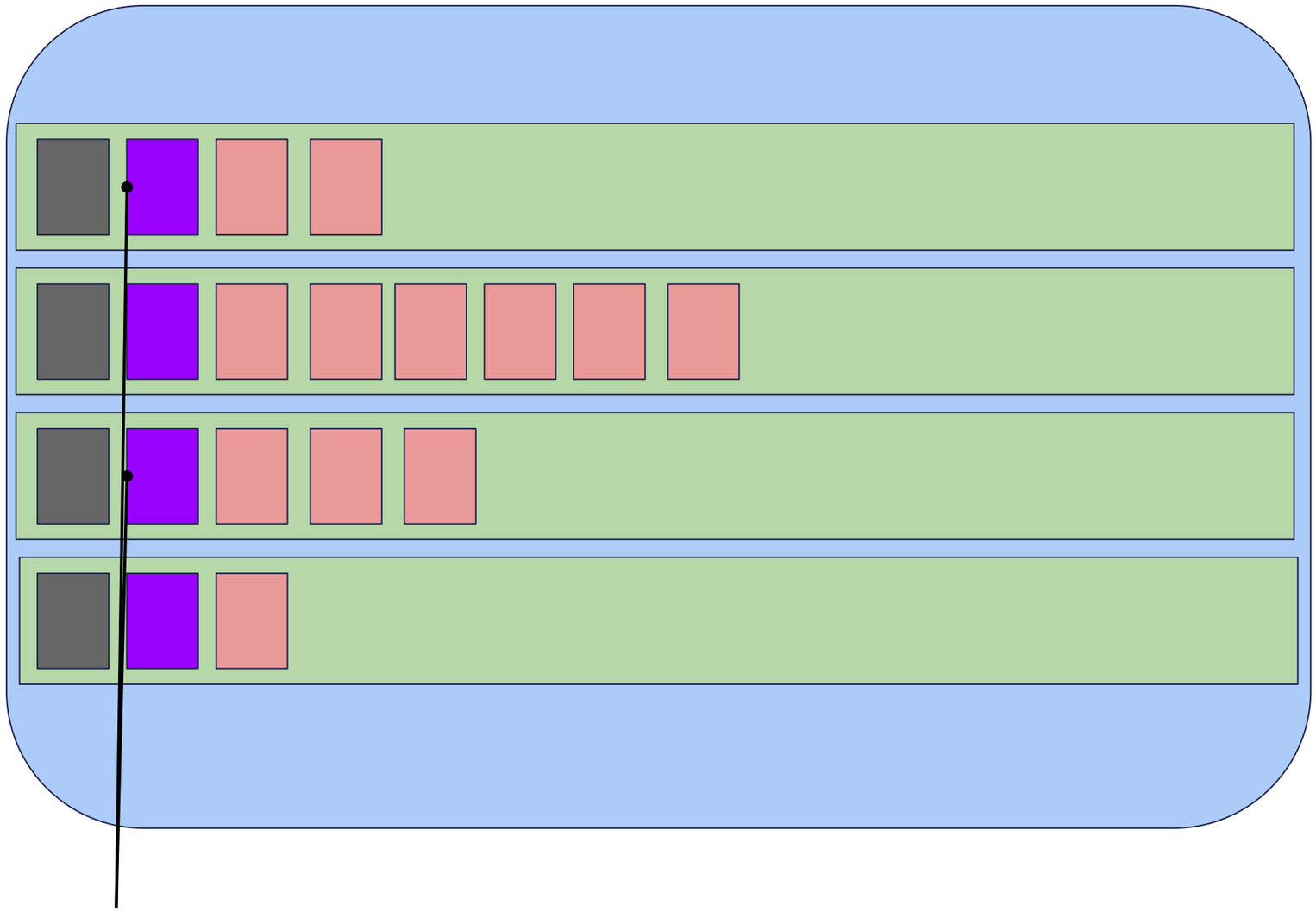
Partition key: Unique value for each item. It’s a simple primary key
Specify the physical location for the partition using internal hash function. So, it also called **hash attribute.**

Key Concepts

Partition keys

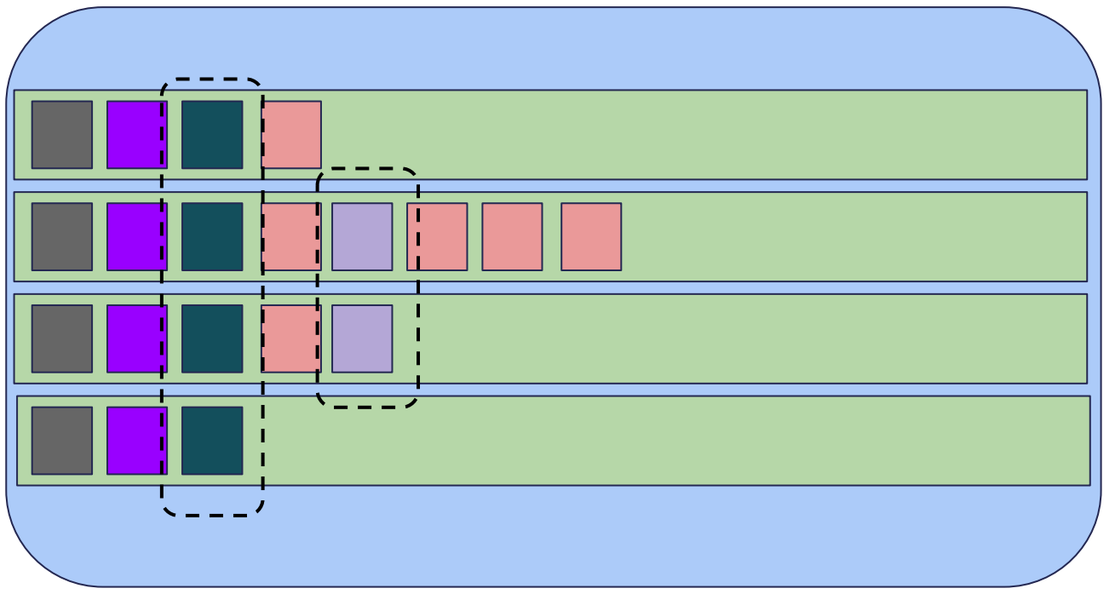
Sort (Range) keys

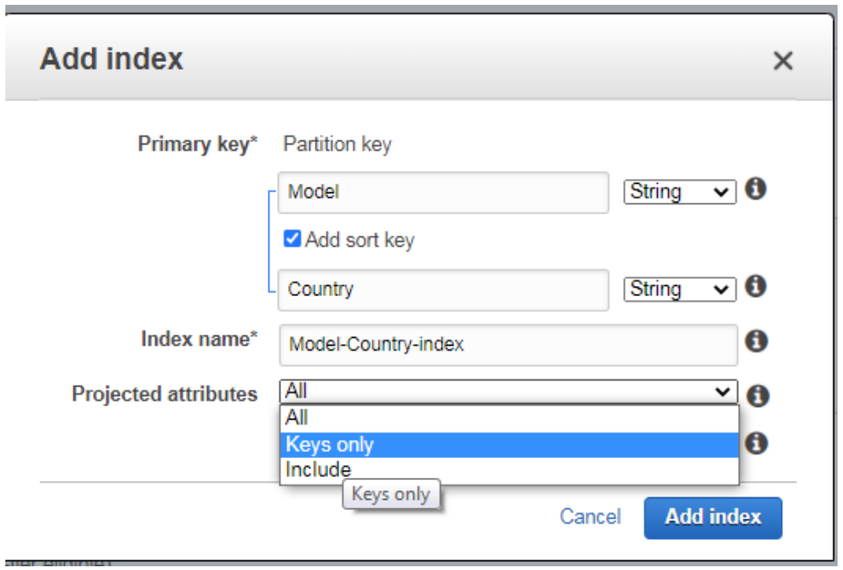
Composite Primary Key



Sort key: Composed primary key for avoid duplicate items with same composed primary key.
Items are ordered by sort key internally on the partition, in that way improve performance, so it’s called **range attribute.**

Global Secondary Index

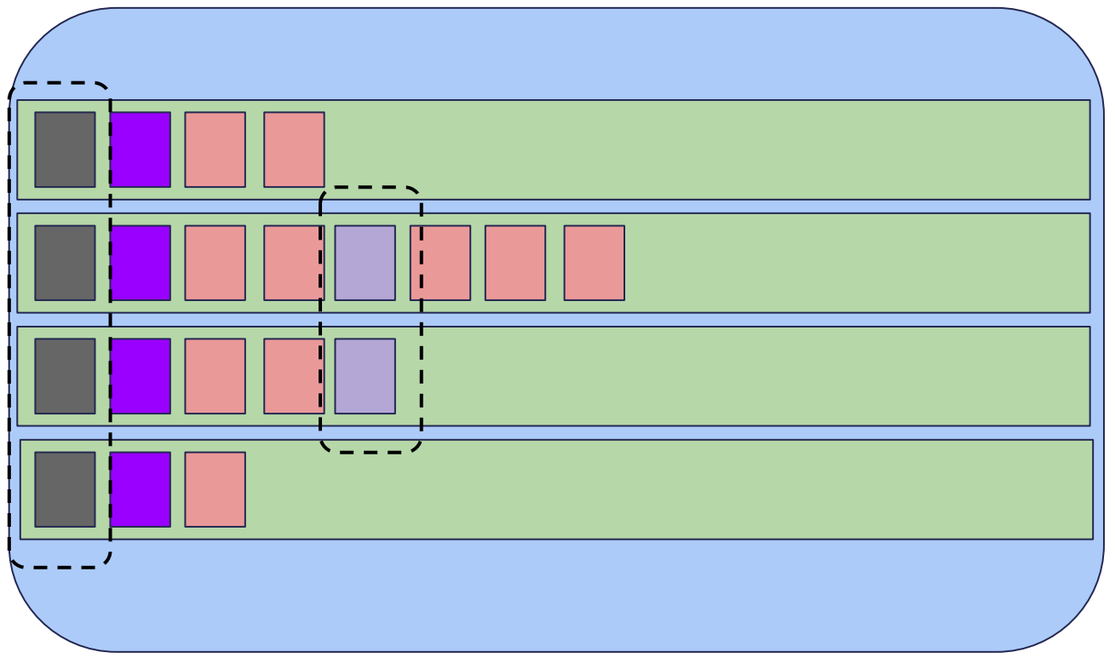


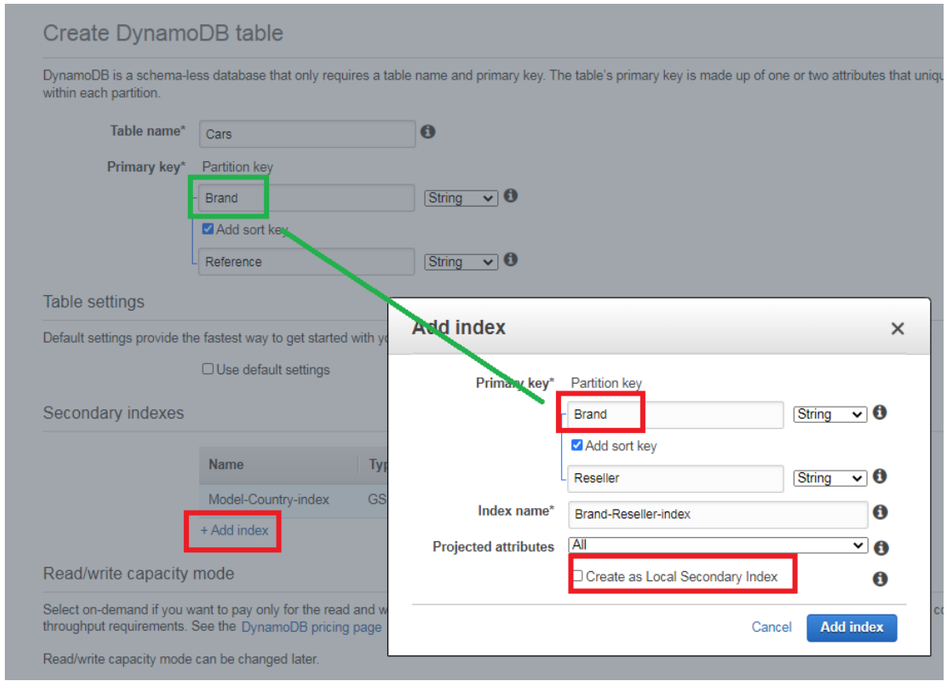


Partition and Sort Key can be different from Base Table. Can span all data from Base Table.

Base Table → Fields to be Projected to Index. DynamoDB Managed index

Local Secondary Index





Has to have the same Partition Key as Base Table and another sort key. It scope is the partition.

Indexes

Music


```
{
    "Artist": "No One You Know",
    "SongTitle": "My Dog Spot",
    "AlbumTitle": "Hey Now",
    "Price": 1.98,
    "Genre": "Country",
    "CriticRating": 8.4
}
```

```
{
    "Artist": "No One You Know",
    "SongTitle": "Somewhere Down The Road",
    "AlbumTitle": "Somewhat Famous",
    "Genre": "Country",
    "CriticRating": 8.4,
    "Year": 1984
}
```

```
{
    "Artist": "The Acme Band",
    "SongTitle": "Still in Love",
    "AlbumTitle": "The Buck Starts Here",
    "Price": 2.47,
    "Genre": "Rock",
    "PromotionInfo": {
        "RadioStationsPlaying": [
            "KHCR",
            "KQBX",
            "WTNR",
            "WJJH"
        ],
        "TourDates": {
            "Seattle": "20150625",
            "Cleveland": "20150630"
        },
        "Rotation": "Heavy"
    }
}
```

```
{
    "Artist": "The Acme Band",
    "SongTitle": "Look Out, World",
    "AlbumTitle": "The Buck Starts Here",
    "Price": 0.99,
    "Genre": "Rock"
}
```


Primary and Sort Key

Music


```
{
    "Artist": "No One You Know",
    "SongTitle": "My Dog Spot",
    "AlbumTitle": "Hey Now",
    "Price": 1.98,
    "Genre": "Country",
    "CriticRating": 8.4
}
```

```
{
    "Artist": "No One You Know",
    "SongTitle": "Somewhere Down The Road",
    "AlbumTitle": "Somewhat Famous",
    "Genre": "Country",
    "CriticRating": 8.4,
    "Year": 1984
}
```

```
{
    "Artist": "The Acme Band",
    "SongTitle": "Still in Love",
    "AlbumTitle": "The Buck Starts Here",
    "Price": 2.47,
    "Genre": "Rock",
    "PromotionInfo": {
        "RadioStationsPlaying": [
            "KHCR",
            "KQBX",
            "WTNR",
            "WJJH"
        ],
        "TourDates": {
            "Seattle": "20150625",
            "Cleveland": "20150630"
        },
        "Rotation": "Heavy"
    }
}
```

```
{
    "Artist": "The Acme Band",
    "SongTitle": "Look Out, World",
    "AlbumTitle": "The Buck Starts Here",
    "Price": 0.99,
    "Genre": "Rock"
}
```


Base Table

GenreAlbumTitle


```
{
    "Genre": "Country",
    "AlbumTitle": "Hey Now",
    "Artist": "No One You Know",
    "SongTitle": "My Dog Spot"
}
```

```
{
    "Genre": "Country",
    "AlbumTitle": "Somewhat Famous",
    "Artist": "No One You Know",
    "SongTitle": "Somewhere Down The Road"
}
```

```
{
    "Genre": "Rock",
    "AlbumTitle": "The Buck Starts Here",
    "Artist": "The Acme Band",
    "SongTitle": "Still in Love"
}
```

```
{
    "Genre": "Rock"
    "AlbumTitle": "The Buck Starts Here",
    "Artist": "The Acme Band",
    "SongTitle": "Look Out, World"
}
```


Global Secondary Index on Projected Space (Index)

Taken from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html (09/08/2020)

API

API Group	Creating	Reading	Updating	Delete
Control Plane	CreateTable	DescribeTable	UpdateTable	DeleteTable
Data Plane	PutItem, BatchWriteItem	GetItem (Consistency as Boolean), BatchGetItem, Query, Scan	UpdateItem	DeleteItem, BatchWriteItem
DynamoDB Streams	ListStreams, DescribeStream, GetShardIterator, GetRecords			
Transactions	TransactWriteItems	TransactGetItems	TransactWriteItems	

Adapted from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.API.html. More info on https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_GetItem.html (09/08/2020)

Indexes

Characteristic	Global Secondary Index	Local Secondary Index
Key Schema	The primary key of a global secondary index can be either simple (partition key) or composite (partition key and sort key).	The primary key of a local secondary index must be composite (partition key and sort key).
Key Attributes	The index partition key and sort key (if present) can be any base table attributes of type string, number, or binary.	The partition key of the index is the same attribute as the partition key of the base table. The sort key can be any base table attribute of type string, number, or binary.
Size Restrictions Per Partition Key Value	There are no size restrictions for global secondary indexes.	For each partition key value, the total size of all indexed items must be 10 GB or less.
Online Index Operations	Global secondary indexes can be created at the same time that you create a table. You can also add a new global secondary index to an existing table, or delete an existing global secondary index. For more information, see Managing Global Secondary Indexes.	Local secondary indexes are created at the same time that you create a table. You cannot add a local secondary index to an existing table, nor can you delete any local secondary indexes that currently exist.
Queries and Partitions	A global secondary index lets you query over the entire table, across all partitions.	A local secondary index lets you query over a single partition, as specified by the partition key value in the query.
Read Consistency	Queries on global secondary indexes support eventual consistency only.	When you query a local secondary index, you can choose either eventual consistency or strong consistency.
Provisioned Throughput Consumption	Every global secondary index has its own provisioned throughput settings for read and write activity. Queries or scans on a global secondary index consume capacity units from the index, not from the base table. The same holds true for global secondary index updates due to table writes.	Queries or scans on a local secondary index consume read capacity units from the base table. When you write to a table, its local secondary indexes are also updated; these updates consume write capacity units from the base table.
Projected Attributes	With global secondary index queries or scans, you can only request the attributes that are projected into the index. DynamoDB does not fetch any attributes from the table.	If you query or scan a local secondary index, you can request attributes that are not projected in to the index. DynamoDB automatically fetches those attributes from the table.

Best Practices:

Minimize use of Indexes, because it costs RCU/WCU.
Choose Projections Carefully
Optimize Frequent Queries to Avoid Fetches
Be Aware of Item-Collection Size Limits When Creating Local Secondary Indexes

Taken from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SecondaryIndexes.html and https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-indexes.html (30/07/2024)

DynamoDB Streams

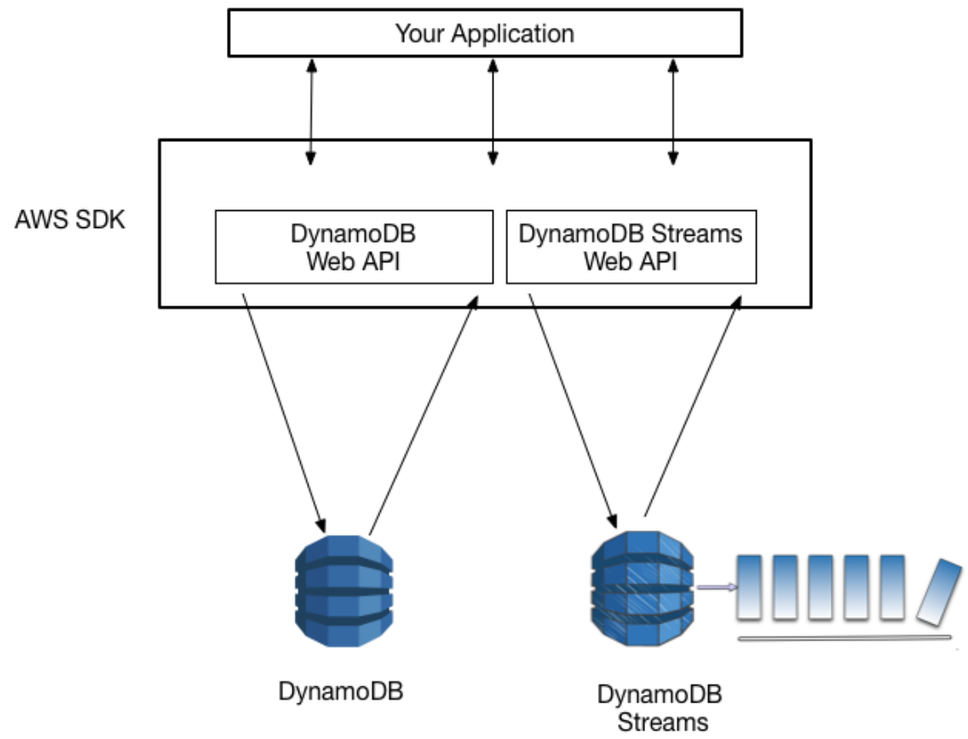
If you able on a table, you can have a ordered sequence of events on another table with source table information such as data change (before and after), timestamp, etc.

You increase the rang of modifying anything, because you reach an origin streams of Lambda, Kinesis, SNS, etc.

Shard retention 24 hours only. A Shard Iterator define the pointer to get new records.

Different Endpoints: DynamoDB, DynamoDB Streams.

Define record store about item change: Only key, old image (previous value), new image (new value) and both.

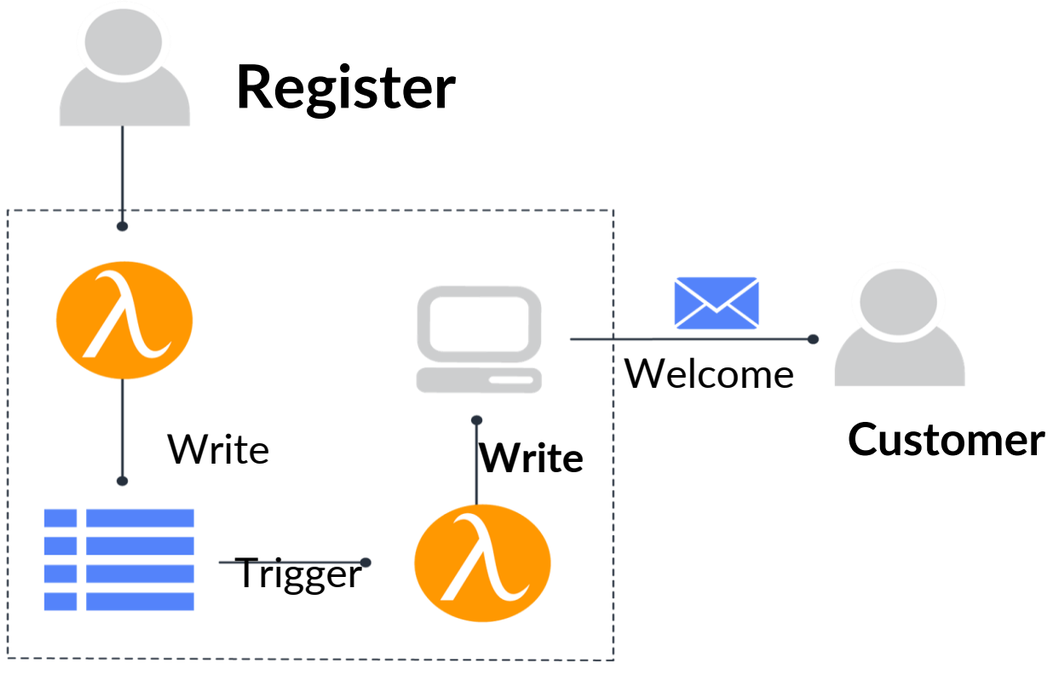


Taken from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.API.html. More info on https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_GetItem.html (30/07/2024)

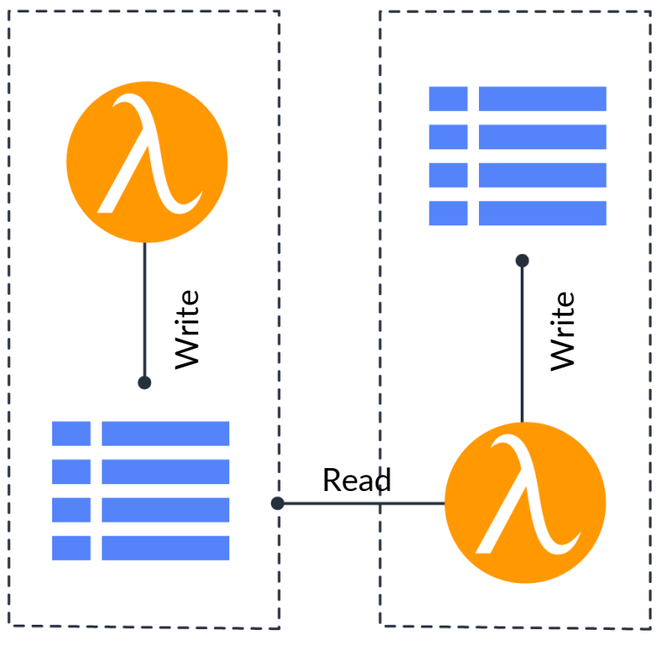
DynamoDB Streams – Use Cases

fmorenod.co

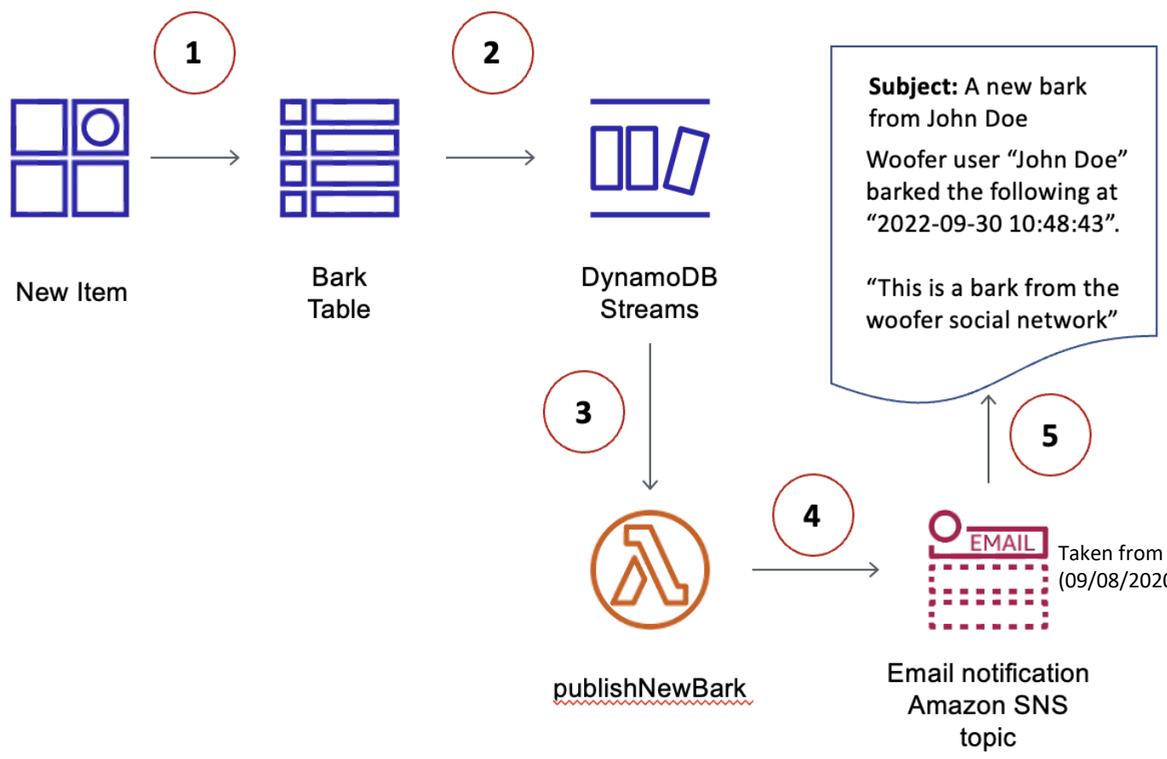
Customer Welcome



Cross-Region Replication and/or Transformation



Example as Social Network and subscribed by your nickname or topic

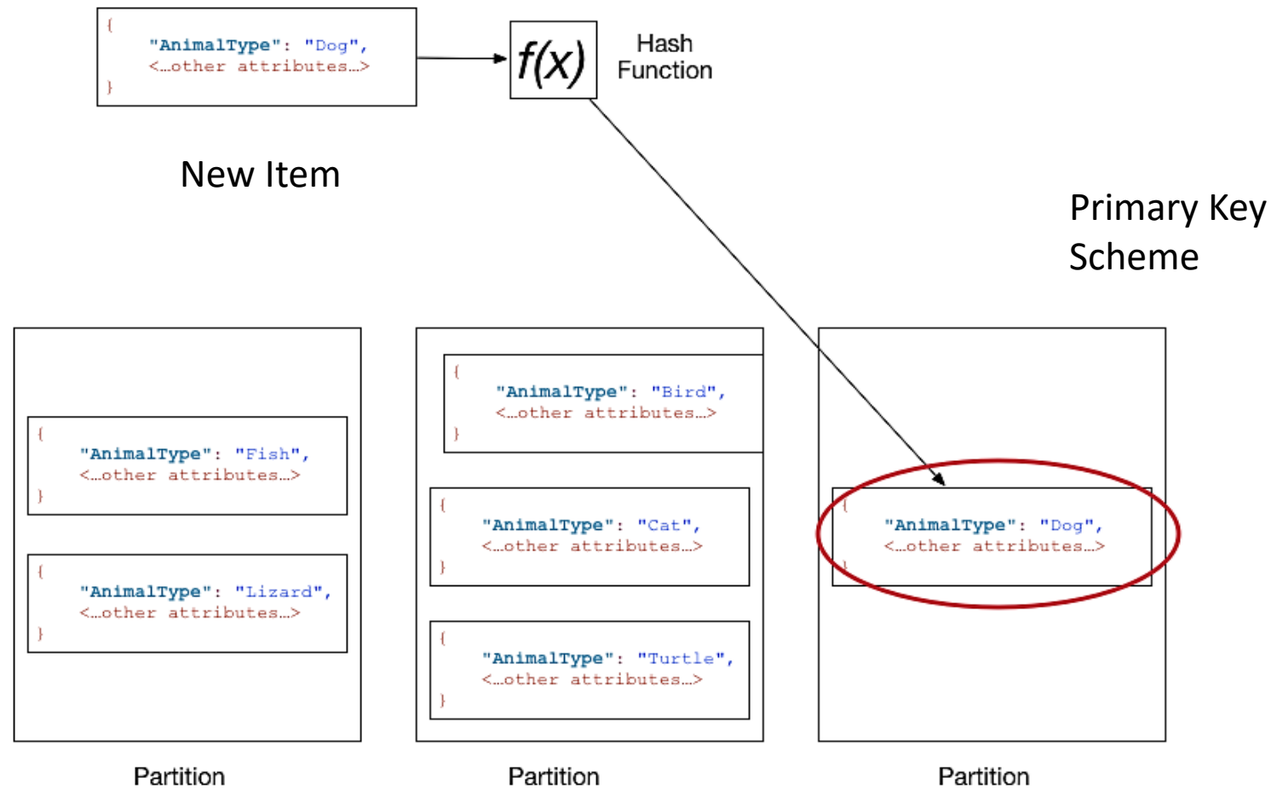


Taken from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.Lambda.Tutorial.html (09/08/2020)

Partitions

Partitions is the way to store data.
AWS manage it (creation, extend), you don't have access.
AWS recommend have a partition key with several values to have a wide range to define partitions and balanced (Hot Key).

Every partitions has 10GB as max size, and 3k RCU/1k WCU



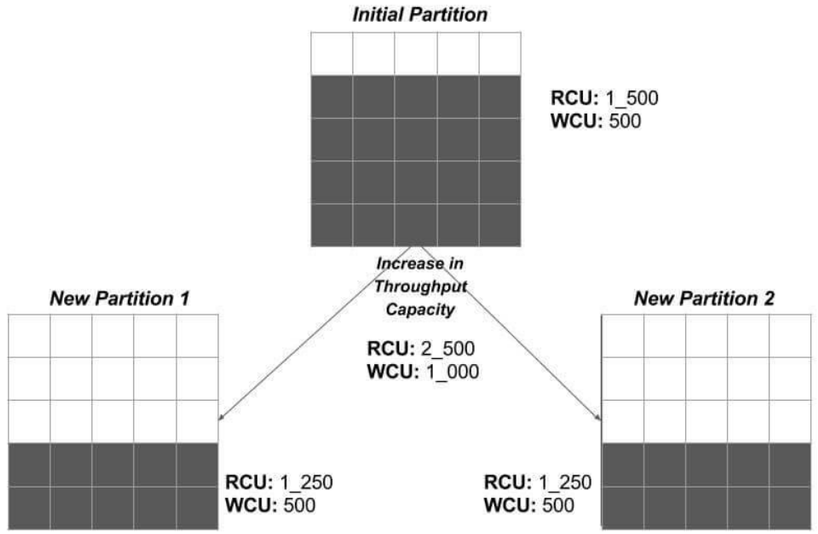
Best Practices:

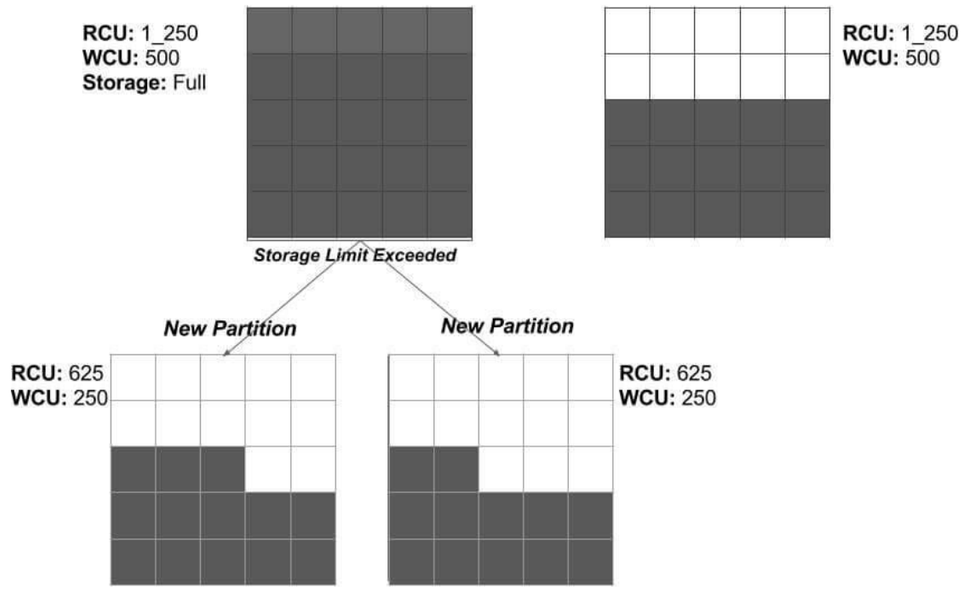
- Isolated Freq Load Items
- Choose a partition key smartly.
- Write Sharding: Pseudo-Randomized sort key suffix

"DynamoDB is optimized for uniform distribution of items across a table's partitions, no matter how many partitions there may be. We recommend that you choose a partition key that can have a large number of distinct values relative to the number of items in the table". More Info at Aws DynamoDB Doc, Best Practice > Partition Key Design

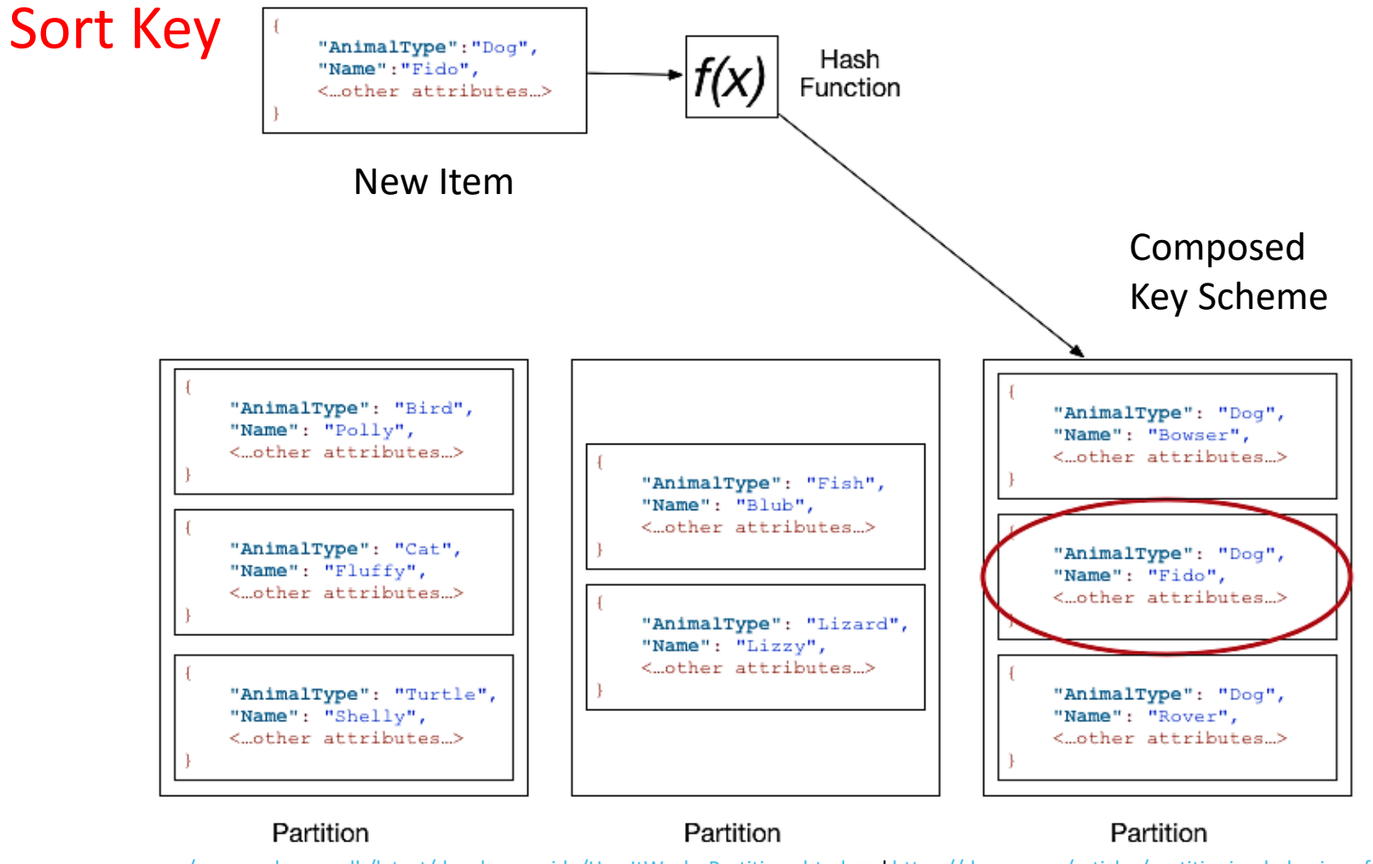
Taken from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.Partitions.html and https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-partition-key-design.html (30/07/2024)

Partitions





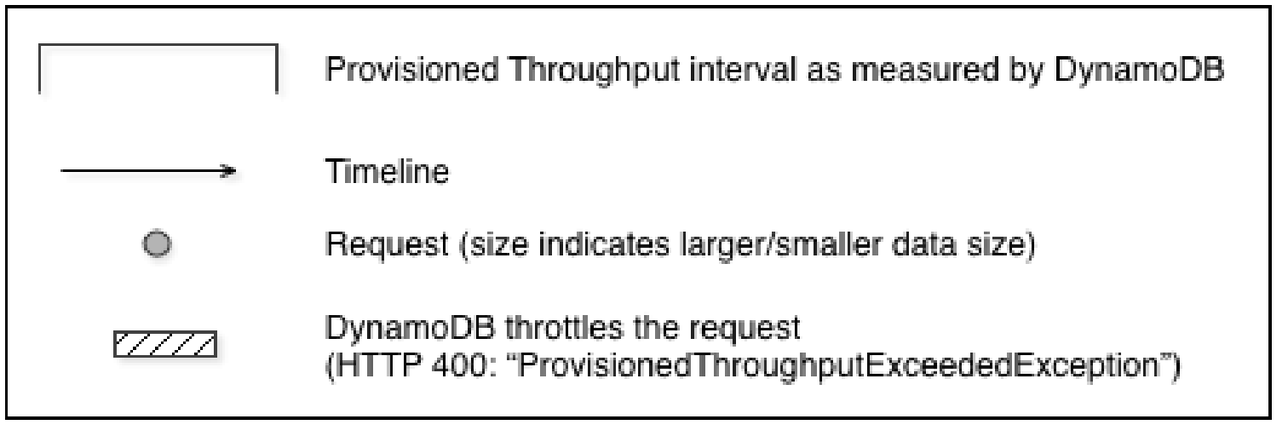
Information store on partition follow the ascendant order using Sort Key.



Taken from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.Partitions.html and https://dzone.com/articles/partitioning-behavior-of-dynamodb#:~:text=The%20partition%20can%20contain%20a,write%20capacity%20units%20(WCUs). (30/07/2024)

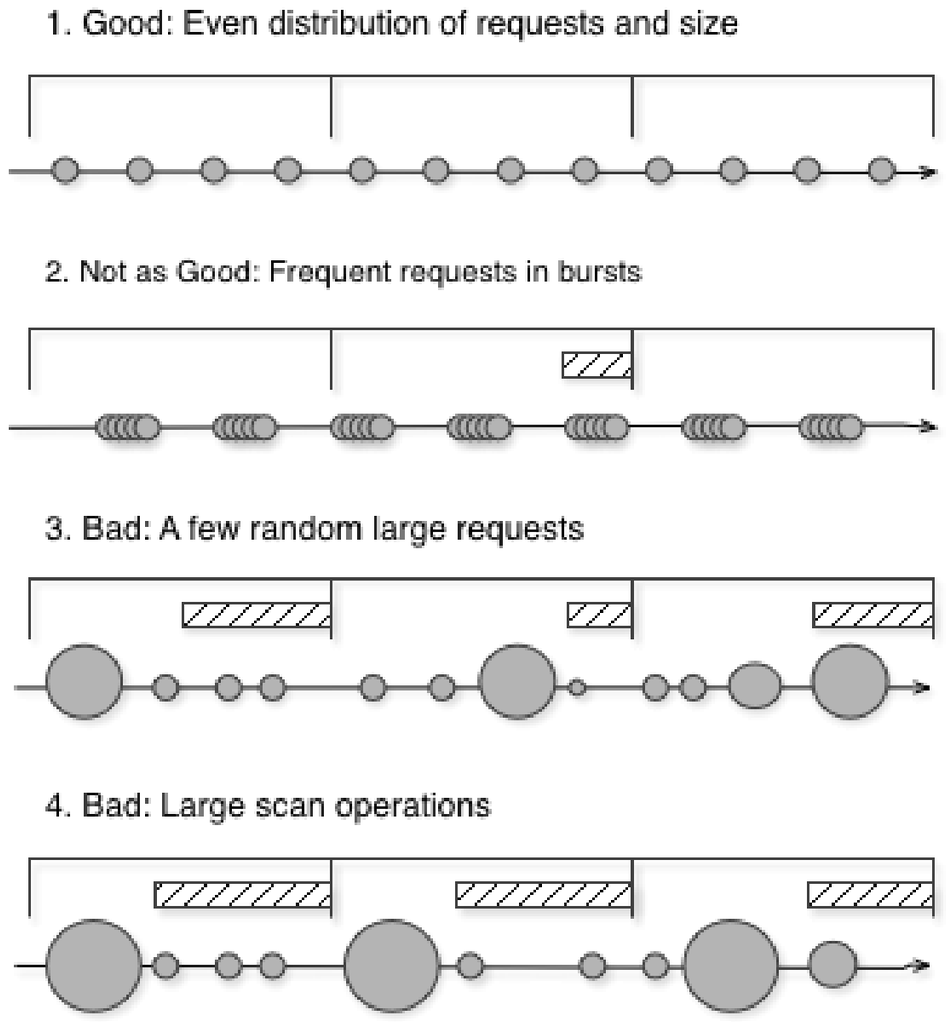
DynamoDB Scan

Asks for **all elements** on the table and secondary indexes about the user request, and its wide scope imply a less efficient read consumption.
It have a large latency and big page response (>1MB) due to immense table.
High consumption can imply a Exceed Throughput Capacity.



Best Practices:

Don't use scan, use query using a right partition key .
Reduce page size (limit option on query).
Isolate scan operations using replicated table.
Configure retries after a Throughput Exceed error.



Taken from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-query-scan.html (30/07/2024)

DynamoDB Query

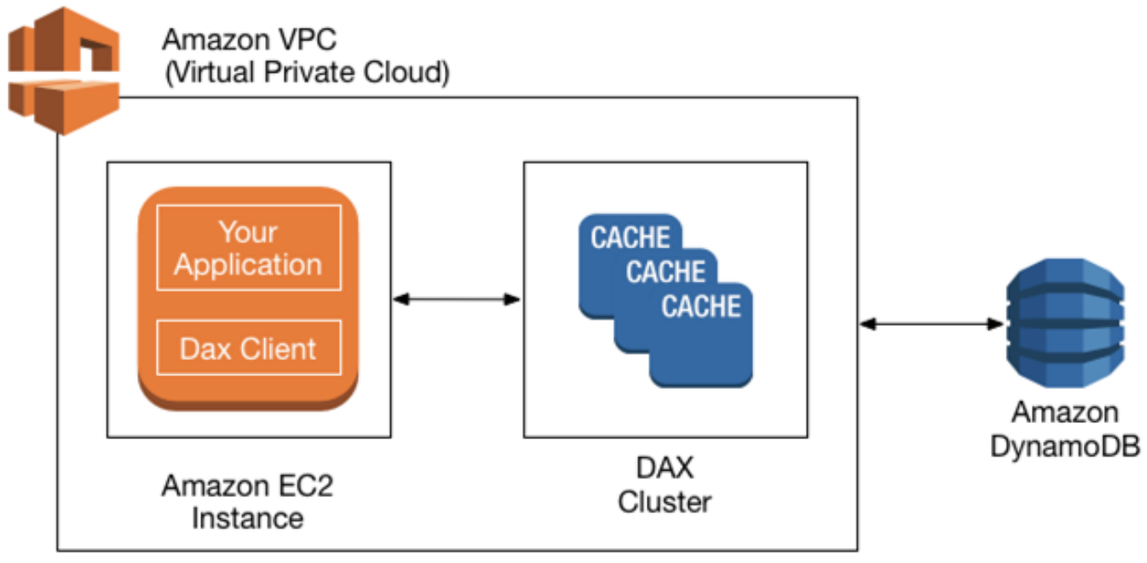
Asks for information using primary or composited key on tables and secondary indexes.
For Query using Equal Operator on Primary Key, and conditionals for Sort Key.
Page size using “Limit”. Consumption control using “ReturnConsumedCapacity”.
Used options for “KeyConditionExpression” for defined Key on Query API.

“DynamoDB calculates the number of read capacity units consumed based on item size, not on the amount of data that is returned to an application. The number of capacity units consumed will be the same whether you request all of the attributes (the default behavior) or just some of them (using a projection expression). The number will also be the same whether or not you use a FilterExpression.”

If Query is applied to	*DynamoDB use RCU*
Table	Provisioned on the table
Global secondary index	Provisioned on the index
Local secondary index	Provisioned on the base table

Taken from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-query-scan.html and https://medium.com/@amos.shahar/dynamodb-query-vs-scan-sql-syntax-and-join-tables-part-1-371288a7cb8f (30/07/2024)

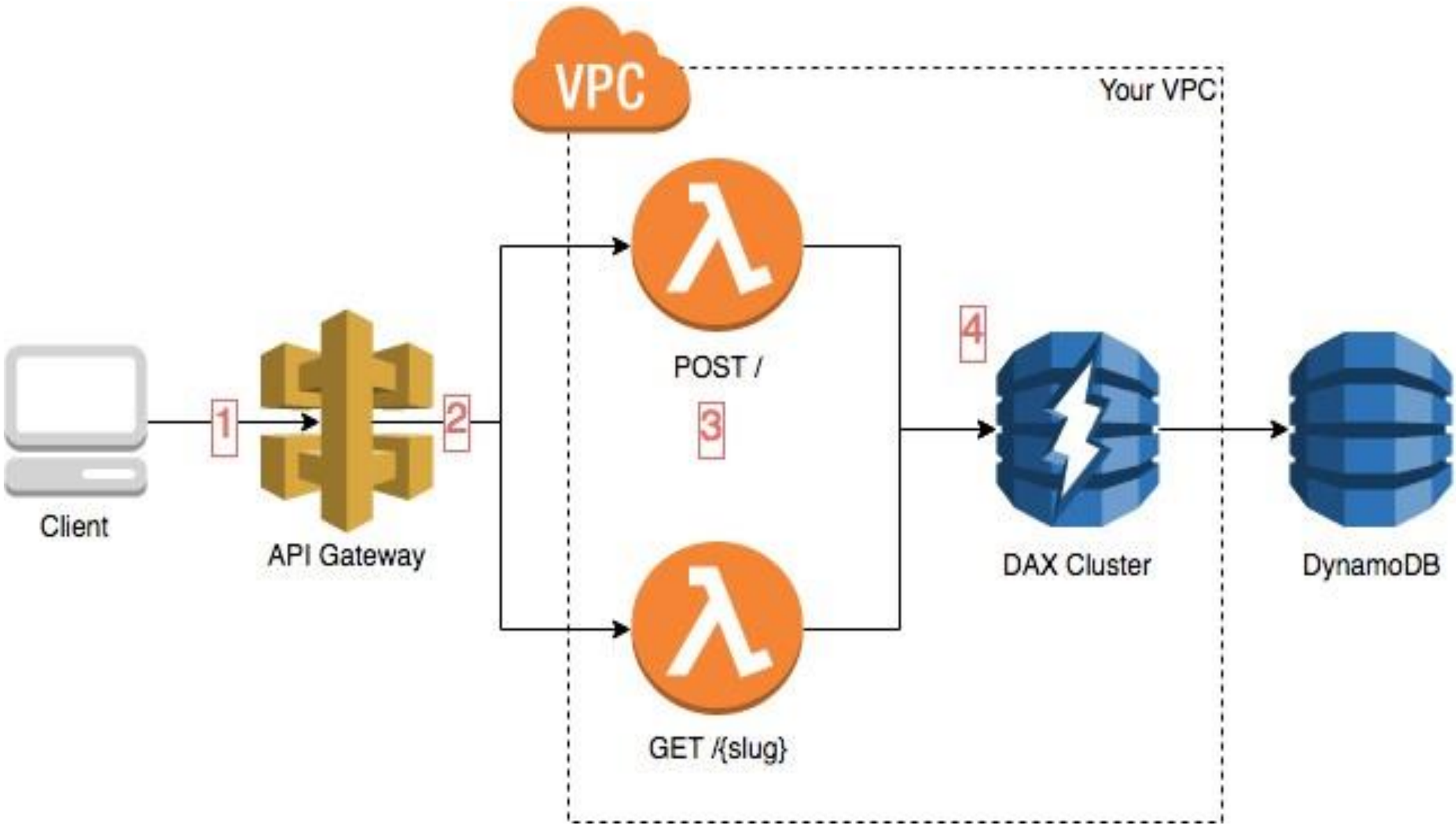
DynamoDB Accelerator - DAX



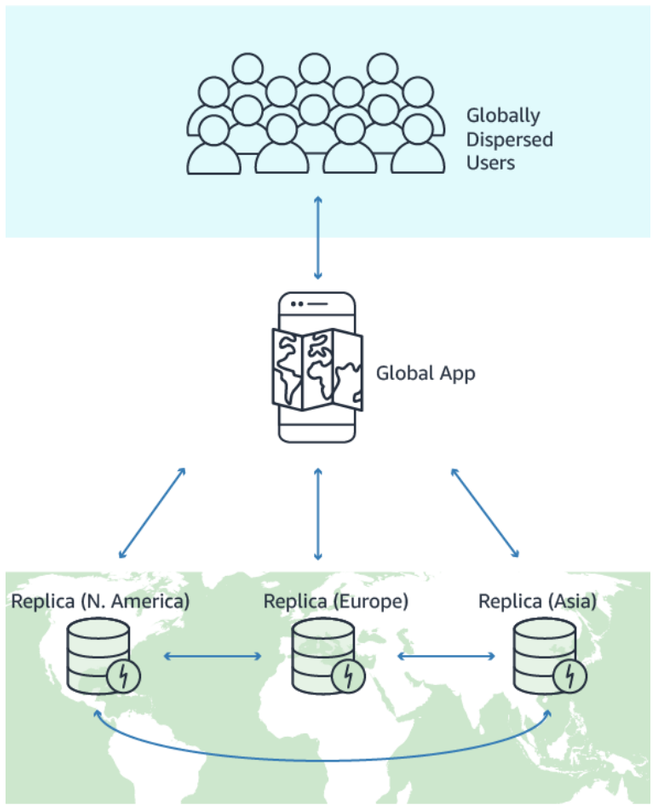
In-Memory Cache Solution for DynamoDB.
Compared response times from milliseconds from DynamoDB to microseconds on DAX.
Managed millions of requests.
Defined API to be simple administration.
Operational costs to avoid over-provisioned Capacity Units. Intensive read operations (small and large dataset).
Encryption-at-rest Support.
Up to 10 Nodes.
All API excepts Control Plane and Read Operations for Eventual Consistency.
Transactional Operations works as pass-through.
Allow stress and load test using T instances, otherwise you have to use R Instances (Memory-Optimized instances).

Not work for:
Strong consistent read.
Not need of faster responses, for cost.
Write-intensive.
Additional Cache Solution.

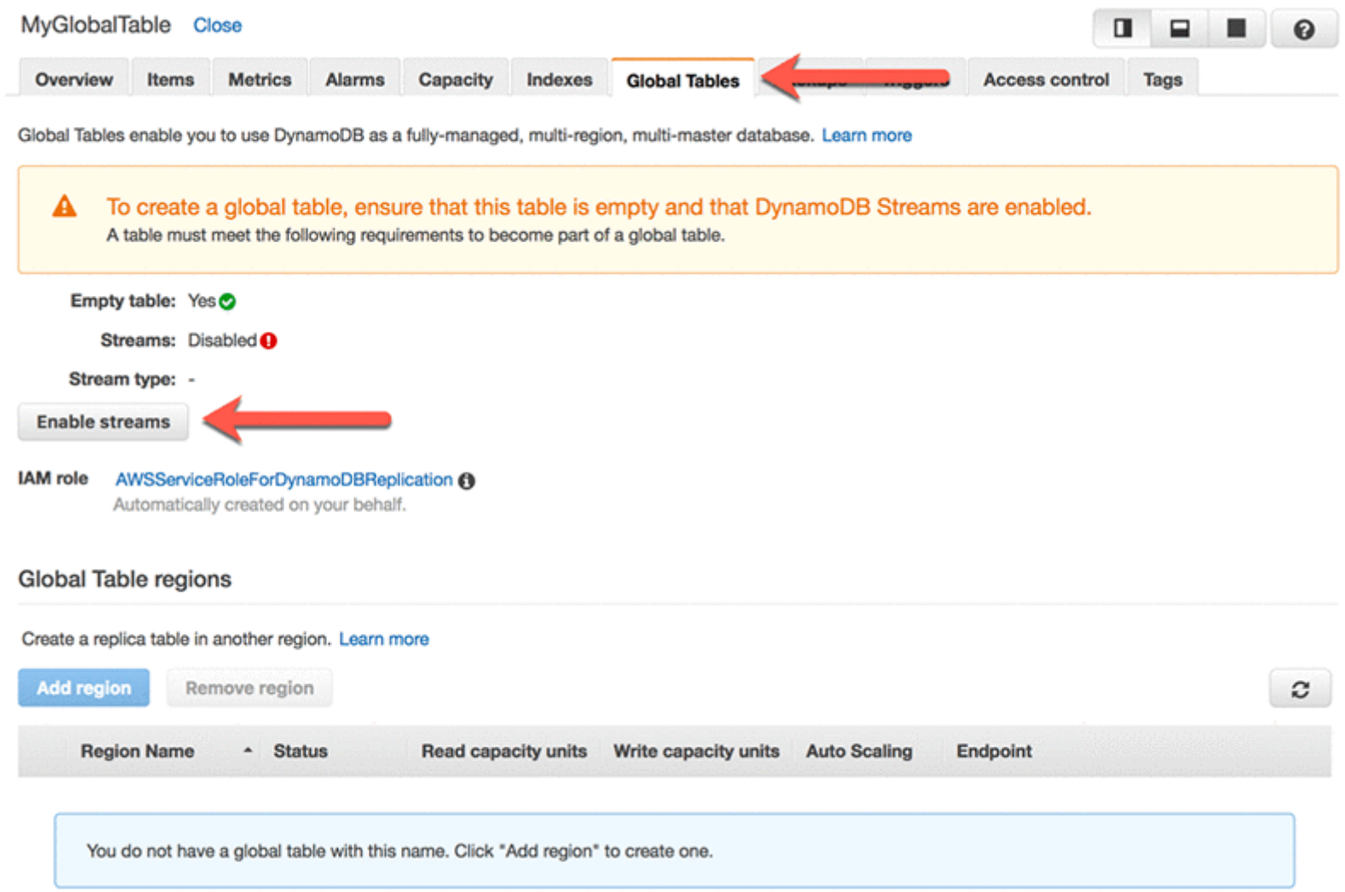
Taken from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DAX.concepts.html (30/07/2024)


VPC
Your VPC
POST /
GET /{slug}
Client
API Gateway
DAX Cluster
DynamoDB
1
2
3
4

DynamoDB Global Tables



Replication on several Regions: You create table and then use Streams and able this feature to replicate automatically.
Latency < 1 segs.
Works for Read Eventual Consistency and Stateless Services.



Usage Cases:
Low Latency for Global Apps
DR Scenarios

Taken from https://aws.amazon.com/dynamodb/global-tables/ and https://aws.amazon.com/blogs/database/how-to-use-amazon-dynamodb-global-tables-to-power-multiregion-architectures/ (30/07/2024)

Serverless Functions

Event driven | Code focused | Managed machines

Simplify: Server Management Overhead (FaaS)
Scale: Automatically and Continuously
Cost Effective: Serverless, Pay-as-you-go
Always Ready: Availability and Fault Tolerant
Service Integration: IAM Roles to integrate with AWS

Taken from https://app.pluralsight.com/library/courses/aws-developer-introduction-aws-lambda and https://app.pluralsight.com/library/courses/aws-developer-lambda-deep-dive (21/06/2021)

Market Leader Comparison

AWS Lambda	Azure Functions
Node, Python, Java, C#, PowerShell, Ruby, Go, user-provided runtimes	Node, Python, Java, C#, PowerShell, F# PHP, batch, bash, other executables
Built-in versioning	No built-in versioning
HTTP endpoints via API Gateway	HTTP endpoints via API Management
15 minute running time limit	10 minute limit (option for unlimited)
1000 concurrent functions (soft limit)	10 concurrent instances
512 MB Ephemeral Storage (/tmp folder)	
You can choose RAM between 128 MB to 10 GB	
CPU Scales with Memory	
IAM Roles to run Functions	
VPC or not VPC ? That’s is the questions	
Load and Save Files on S3, Logs on CloudWatch Logs	
Chain Functions Together	

Taken from https://app.pluralsight.com/library/courses/aws-developer-introduction-aws-lambda (21/06/2021) and https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html (23/06/2021)

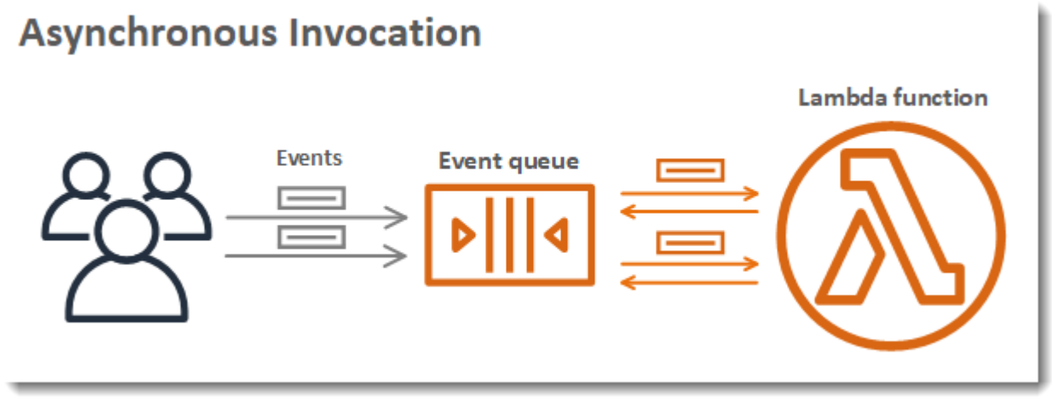
Limits

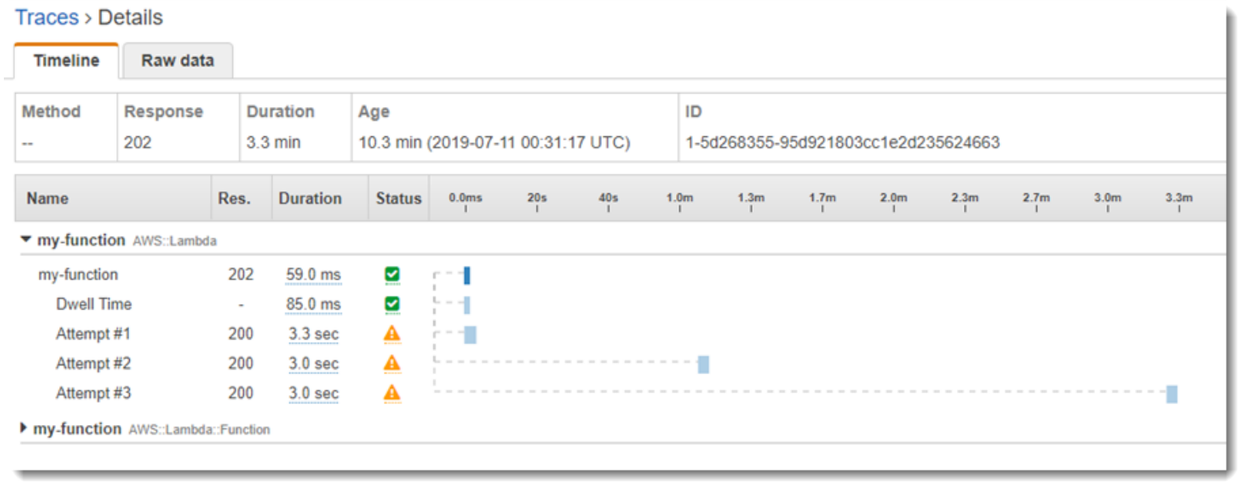
Resource	Quota
Function memory allocation	128 MB to 10,240 MB, in 1-MB increments.
Function timeout	900 seconds (15 minutes)
Function environment variables	4 KB
Function resource-based policy	20 KB
Function layers	five layers
Function burst concurrency	500 - 3000 (varies per Region)
Invocation payload (request and response)	6 MB (synchronous) 256 KB (asynchronous)
Deployment package (.zip file archive) size	50 MB (zipped, for direct upload) 250 MB (unzipped, including layers) 3 MB (console editor)
Container image code package size	10 GB
Test events (console editor)	10
/tmp directory storage	512 MB
File descriptors	1,024

Size of:

- Environment Variables (i.e. Path)
- Policy
- Layers
- Concurrency (per Region)
- Invocation payload (Input and Output of the Function)
- Code Size

Asynch Targets:
SQS, SNS, Lambda, EventBridge





Taken from https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html (23/06/2021)

Lambda > Functions > publishNewBark

publishNewBark

Throttle | Copy ARN | Actions

Function overview Info

Description
-

Last modified
18 seconds ago

Function ARN
arn:aws:lambda:us-east-1:768312754627:function:publishNewBark

Code | Test | Monitor | Configuration | Aliases | Versions

Code source Info

Upload from

File Edit Find View Go Tools Window | Test | Deploy | Changes deployed

Go to Anything (Ctrl-P)

Environment

publishNewBark - /
publishNewBark.js

publishNewBark

```
1  'use strict';
2  var AWS = require("aws-sdk");
3  var sns = new AWS.SNS();
4
5  exports.handler = (event, context, callback) => {
6
7      event.Records.forEach((record) => {
8          console.log('Stream record: ', JSON.stringify(record, null, 2));
9
10         if (record.eventName == 'INSERT') {
11             var who = JSON.stringify(record.dynamodb.NewImage.Username.S);
12             var when = JSON.stringify(record.dynamodb.NewImage.Timestamp.S);
13             var what = JSON.stringify(record.dynamodb.NewImage.Message.S);
```

Code properties

Package size	SHA256 hash	Last modified
869.0 byte	dj7Rv0732qgxO1w+7m3VzpesacVE8YkyBHgpdCW35XY=	June 21, 2021, 04:35 AM GMT-5

Runtime settings Info

Edit

Runtime	Handler Info
Node.js 10.x	publishNewBark.handler

Code | Test | Monitor | Configuration | Aliases | Versions

- General configuration
- **Triggers**
- Permissions
- Destinations
- Environment variables
- Tags
- VPC
- Monitoring and operations tools
- Concurrency
- Asynchronous invocation
- Code signing
- Database proxies
- File systems
- State machines

Triggers (7)

Enable | Disable

DynamoDB | 7 matches

Trigger
DynamoDB: BarkTable (Disabled) arn:aws:dynamodb:us-east-1:768312754627:table/BarkTable/stream/2020-11-09T12:15:40.344 ▶ Details
DynamoDB: BarkTable (Disabled) arn:aws:dynamodb:us-east-1:768312754627:table/BarkTable/stream/2020-08-16T10:51:27.835 ▶ Details
DynamoDB: BarkTable (Disabled) arn:aws:dynamodb:us-east-1:768312754627:table/BarkTable/stream/2020-08-15T17:13:01.591 ▶ Details
DynamoDB: BarkTable (Enabled) arn:aws:dynamodb:us-east-1:768312754627:table/BarkTable/stream/2021-06-21T09:33:53.013 ▶ Details

Same Table.
LAST: Latest Stream ARN

29.654

AWS Identity and Access Management

Users and Groups
Create users and assign security credentials

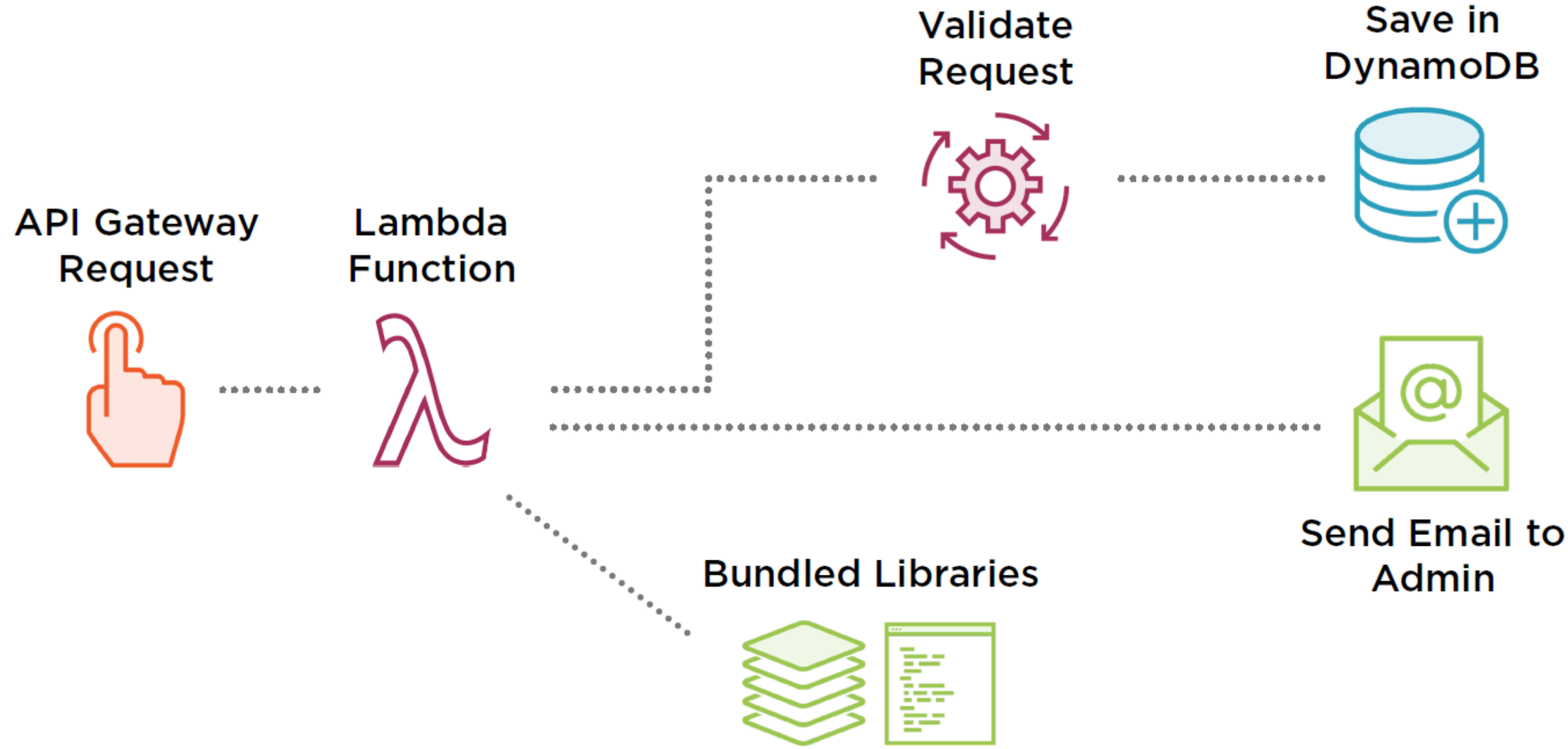
Roles
Grants permissions to entities or AWS services

Policies
Define permissions of an identity or resource

Event Source Mapping instead of Events for trigger: i.e. DynamoDB Streams, SQS, Kinesis Data Streams vs Cloudwatch Events

Lambda – Use Cases I

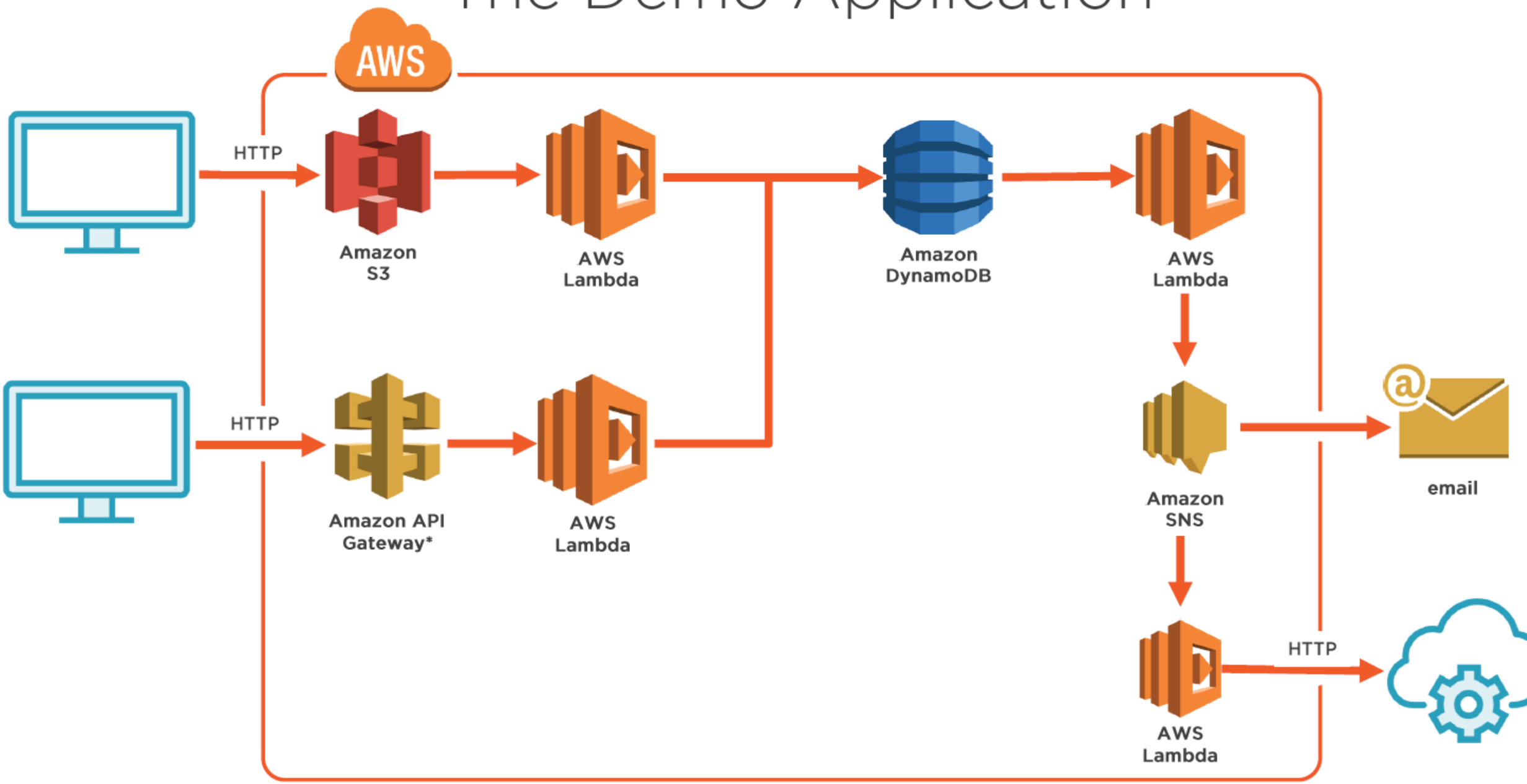
New Customers Service



Taken from https://app.pluralsight.com/library/courses/aws-developer-introduction-aws-lambda (21/06/2021)

Lambda – Use Cases II

The Demo Application



Taken from https://app.pluralsight.com/library/courses/aws-developer-lambda-deep-dive (21/06/2021)