BIG DATA on AWS

Data Storage

Agenda

01 What is Amazon DynamoDB?

02 Core Components of DynamoDB

03 DynamoDB Concepts

04 RDBMS vs DynamoDB

05 Dynamo Replication

06 Backup and Restore

07 DynamoDB Best Practices

08 What is S3 Glacier?

Agenda

09 Glacier Vaults and Archives

10 Introduction to RDS

11 Basics of RDS

12 Creating a MySQL Database

What is Amazon DynamoDB?

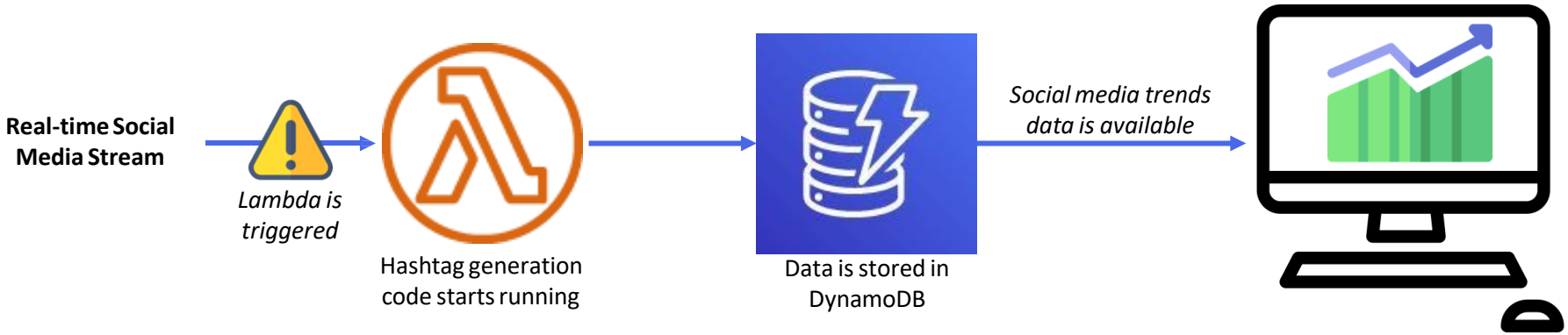
What is Amazon DynamoDB?

Amazon DynamoDB is a NoSQL database, which is fully managed by AWS to provide fast and predictable performance. It creates database tables that can store and retrieve any amount of data

- Key-value and document data models
- Automated global replication
- Read/Write capacity modes
- Point-in-time recovery

What is Amazon DynamoDB?

This is a sample use case on how DynamoDB can be used:



Core Components of DynamoDB

Core Components of DynamoDB

- Tables, Items, and Attributes
- Primary Key
- Secondary Indexes
- DynamoDB Streams

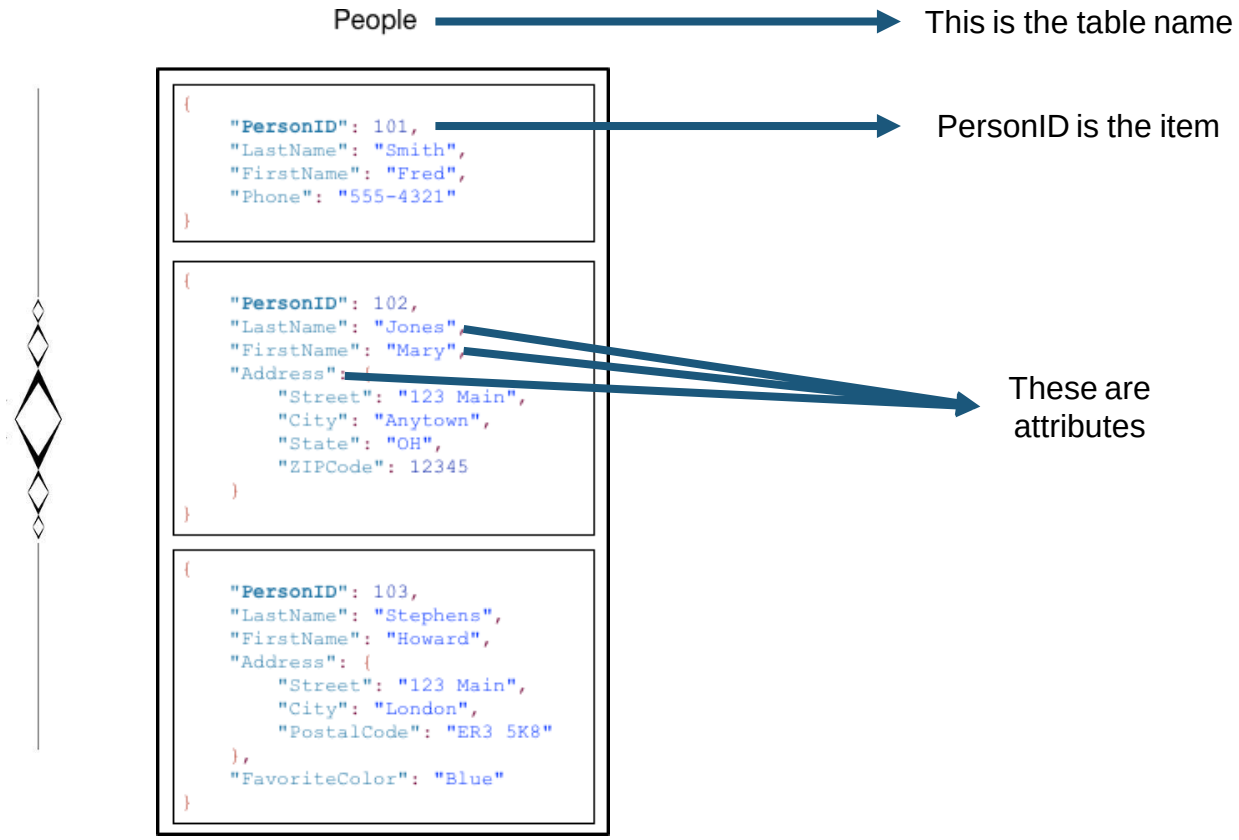
A **table** is a collection of data

Each table contains zero or more items. An **item** is a group of attributes uniquely identifiable among all of the other items

Each item is composed of one or more attributes. An **attribute** is a fundamental data element, something that does not need to be broken down any further

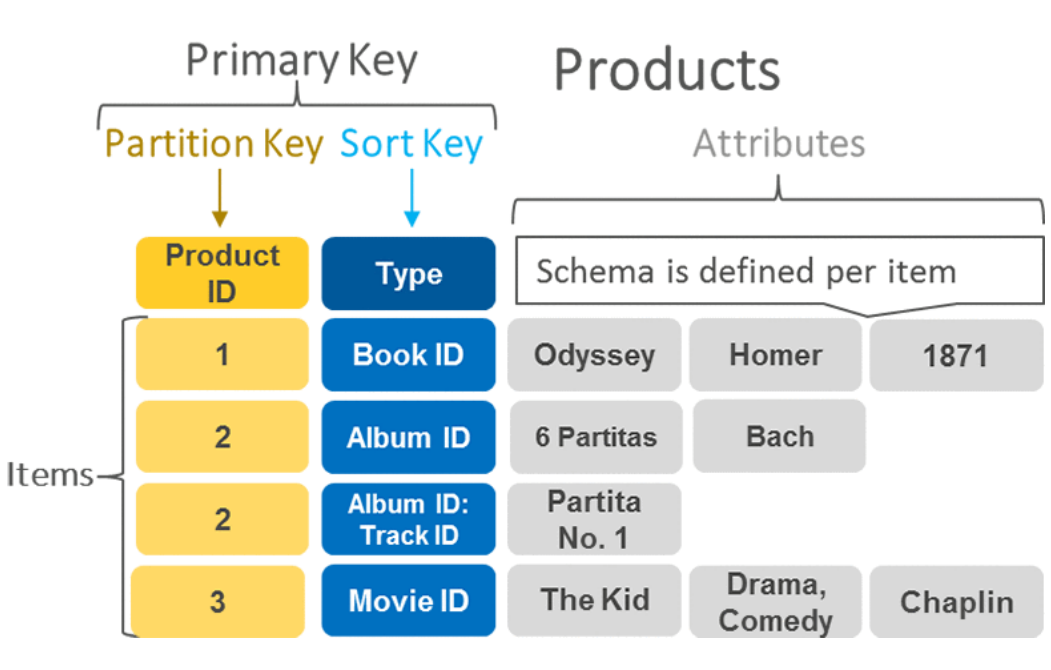
Core Components of DynamoDB

- Tables, Items, and Attributes
- Primary Key
- Secondary Indexes
- DynamoDB Streams



Core Components of DynamoDB

- Tables, Items, and Attributes
- Primary Key
- Secondary Indexes
- DynamoDB Streams



Core Components of DynamoDB

- Tables, Items, and Attributes
- Primary Key
- Secondary Indexes
- DynamoDB Streams

When we create a table, in addition to the table name, we must specify the primary key of the table

Partition Key	**Partition Key and Sort Key**
A simple primary key comprises one attribute known as the partition key. In the previous example, ‘PersonID’ is the partition key	It is possible for two items to have the same partition key value. However, those two items must have different sort key values

Core Components of DynamoDB

- Tables, Items, and Attributes
- Primary Key
- Secondary Indexes
- DynamoDB Streams

We can create one or more secondary indexes on a table. A secondary index lets us query the data in the table using an alternate key

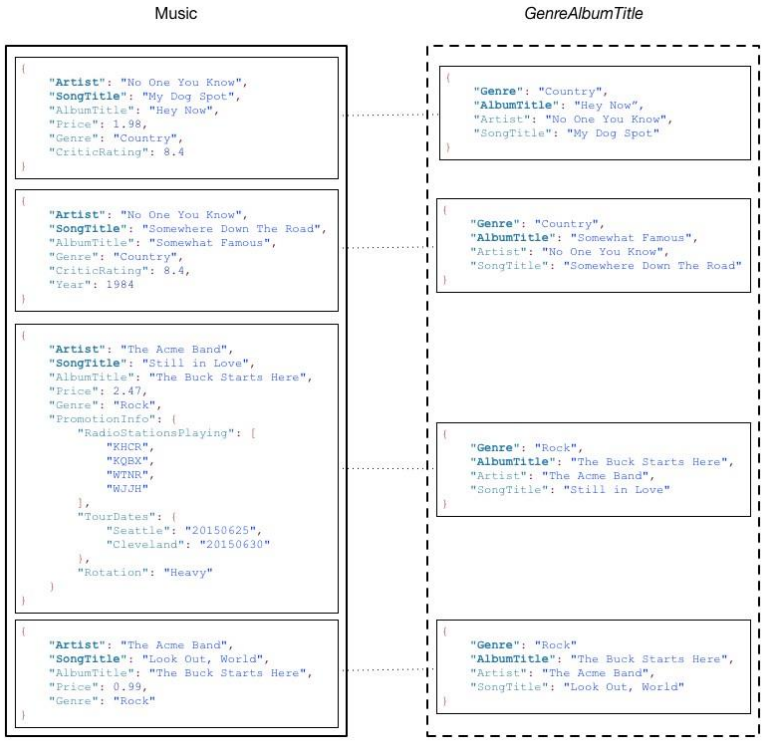
Types of Secondary Indexes

1. Global secondary index: An index with a partition key and a sort key that can be different from those on the table
2. Local secondary index: An index that has the same partition key as the table, but a different sort key

Core Components of DynamoDB

- Tables, Items, and Attributes
- Primary Key
- Secondary Indexes
- DynamoDB Streams

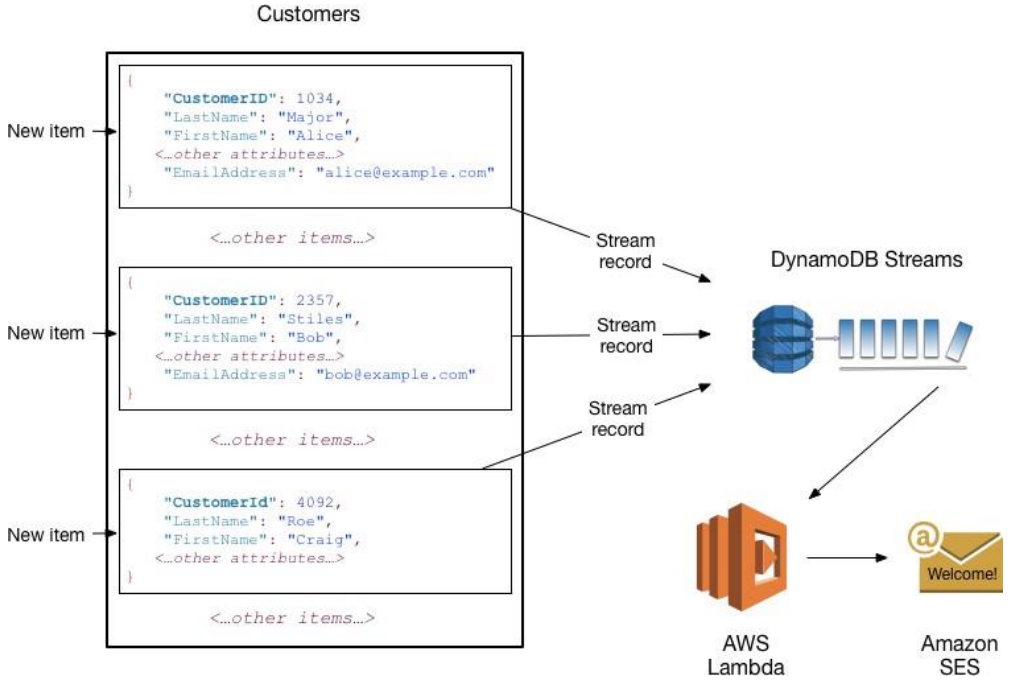
Limitations: 20 GSIs per table



Core Components of DynamoDB

- Tables, Items, and Attributes
- Primary Key
- Secondary Indexes
- DynamoDB Streams

It is a feature that captures data modification events in DynamoDB tables, and it is optional



DynamoDB Concepts

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

Names should be meaningful and concise—for example, names such as Products, Books, and Authors are self-explanatory

Characters we can use are as follows:

- a-z
- A-Z
- 0-9
- _ (underscore)
- - (dash)
- . (dot)

- Attribute names must be between 1 and 255 characters long

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

- **Scalar types**: They can represent exactly one value. The scalar types are number, string, binary, Boolean, and null
- **Document types**: They can represent a complex structure with nested attributes as we would find in a JSON document
- **Set types**: They can represent multiple scalar values. The set types include string set, number set, and binary set

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

Scalar Types

- **Number**: Numbers can be positive, negative, or zero
- **String**: It is a unicode with UTF-8 binary encoding, and the length should be greater than 0
- **Binary**: It can store any binary data, such as compressed text, encrypted data, or images
- **Boolean**: It can store either true or false
- **Null**: It is an attribute with an unknown or undefined state

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

Document Types

- **List**: A list type attribute can store an ordered collection of values. Lists are enclosed in square brackets: [...]
- **Map**: A map type attribute can store an unordered collection of name-value pairs. Maps are enclosed in curly braces: { ... }

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

Document Types

List

```
FavoriteThings: ["Cookies", "Coffee", 3.14159]
```

Map

```
{
    Day: "Monday",
    UnreadEmails: 42,
    ItemsOnMyDesk: [
        "Coffee Cup",
        "Telephone",
        {
            Pens: { Quantity : 3},
            Pencils: { Quantity : 2},
            Erasers: { Quantity : 1}
        }
    ]
}
```

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

Set Types

DynamoDB supports the types that represent sets of number, string, or binary values. All elements within a set must be of the same type

```
["Black", "Green", "Red"]

[42.2, -19, 7.5, 3.14]

["U3Vubnk=", "UmFpbnk=", "U25vd3k="]
```

For example, an attribute of the type Number Set can only contain numbers; a String Set can only contain strings; and so on

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

DynamoDB supports **Eventually Consistent** reads and **Strongly Consistent** reads

Eventually Consistent

When reading data from DynamoDB, this may give you results quicker because of low latency, but data might not be up-to-date

Strongly Consistent

In DynamoDB this offers updated data all the time, but has a very high latency.

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

Amazon DynamoDB has two read/write capacity modes for processing reads and writes on our tables:

- **On-demand**
- **Provisioned (default, free-tier eligible)**

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

Amazon DynamoDB on-demand mode is a flexible billing option capable of serving thousands of requests per second without capacity planning

Use on-demand mode only under these conditions:

- When we create new tables with unknown workloads
- When we have unpredictable application traffic
- When we prefer the ease of paying for only what we use

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

If we choose the provisioned mode, we specify the number of reads and writes per second that we require for our application

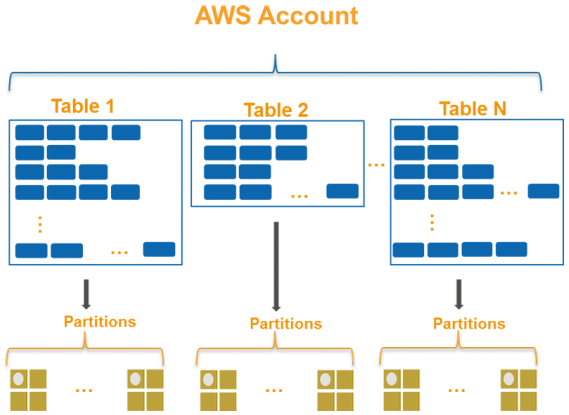
Use provisioned mode only under these conditions:

- When we have predictable application traffic
- When we run applications with traffic, which is consistent or ramps gradually
- When we can forecast capacity requirements to control costs

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

A **partition** is the allocation of storage for a table, backed by solid state drives (SSDs) and automatically replicated across multiple availability zones within an AWS region



Partition management is handled entirely by DynamoDB—we never have to manage partitions ourselves

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

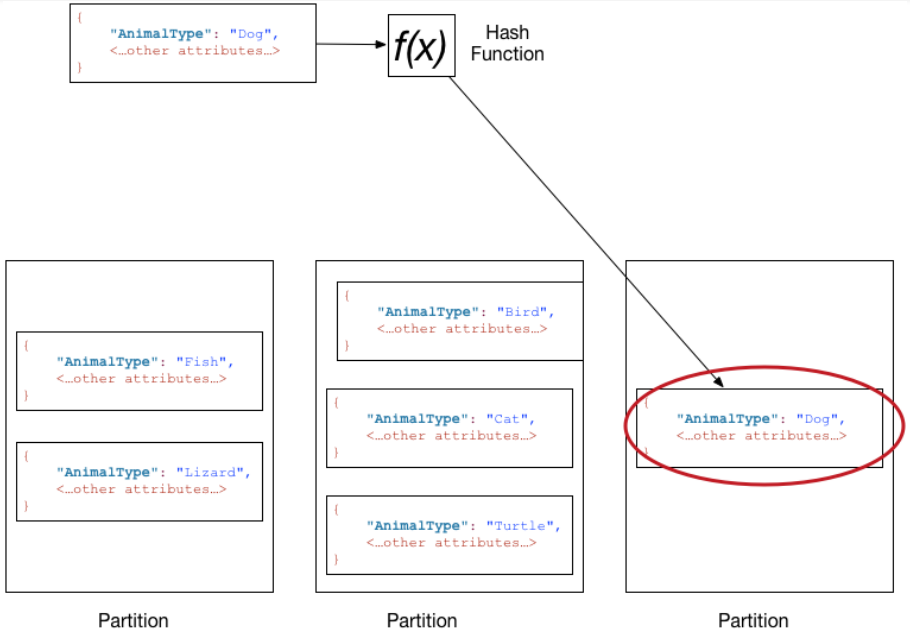
DynamoDB allocates additional partitions to a table in the following situations:

- If we increase the table's provisioned throughput settings beyond what the existing partitions can support
- If an existing partition fills to capacity and more storage space is required

DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

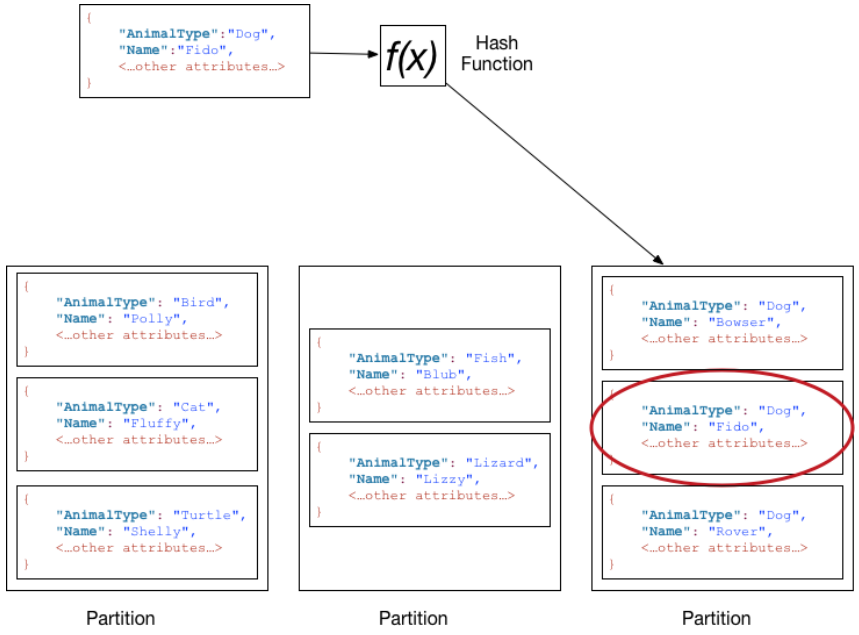
If our table has a simple primary key (partition key only), DynamoDB stores and retrieves each item based on its partition key value



DynamoDB Concepts

- Naming Rules
- DynamoDB Data Types
- Read Consistency
- Read/Write Capacity Modes
- Partitions & Data Distribution

If the table has a composite primary key, DynamoDB calculates the hash value of the partition key the same way as in simple primary key

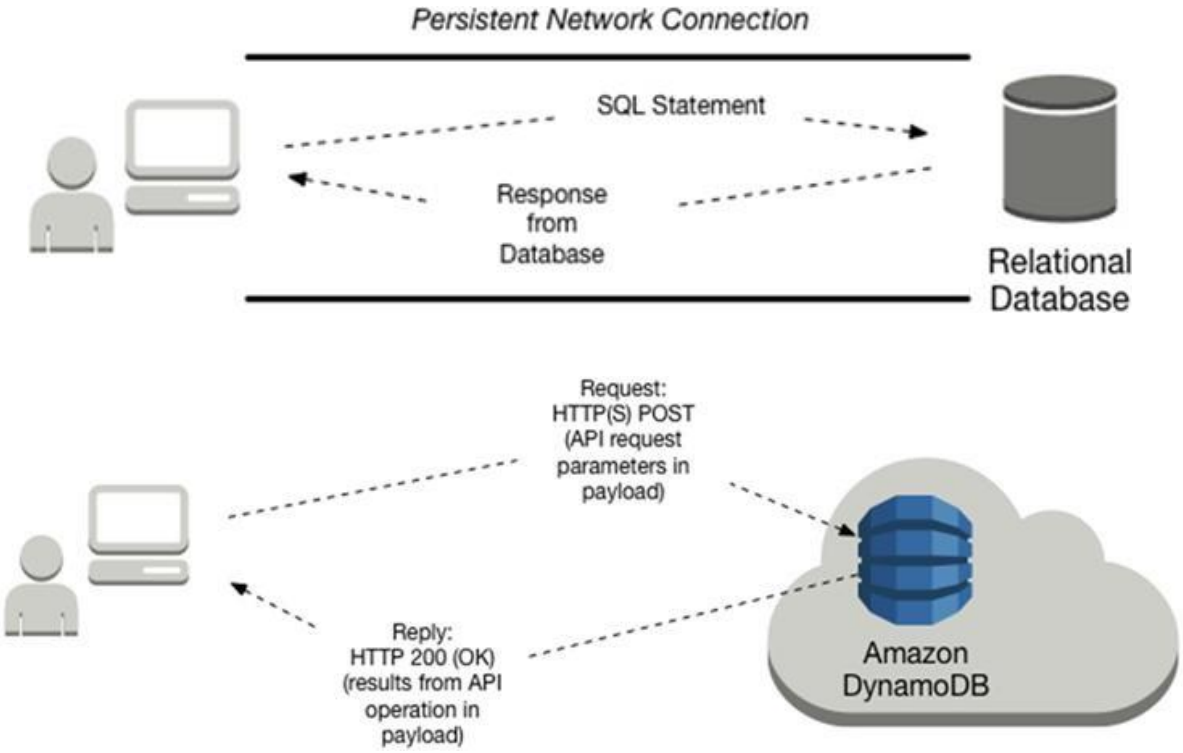


RDBMS vs DynamoDB

RDBMS vs DynamoDB: Characteristics

Characteristic	Relational Database Management System (RDBMS)	DynamoDB
Optimal Workloads	Ad-hoc queries, data warehousing, and OLAP (online analytical processing)	Web-scale applications, including social networks, gaming, etc.
Data Model	Requires a well-defined schema, where data is normalized into tables, rows, and columns	Schema less; it can manage structured or semi structured data, including JSON documents
Data Access	SQL is the standard for storing and retrieving data	We can use AWS Management Console, AWS CLI, or AWS SDKs to work with DynamoDB
Performance	Optimized for storage, so performance generally depends on the disk subsystem	Optimized for compute, so performance is mainly a function of the underlying hardware and network latency
Scaling	Easiest to scale up with faster hardware; it is possible for database tables to span across multiple hosts in a distributed system, but this requires additional investment	Designed to scale out using distributed clusters of hardware; this design allows increased throughput without increased latency

RDBMS vs DynamoDB: Client Interaction

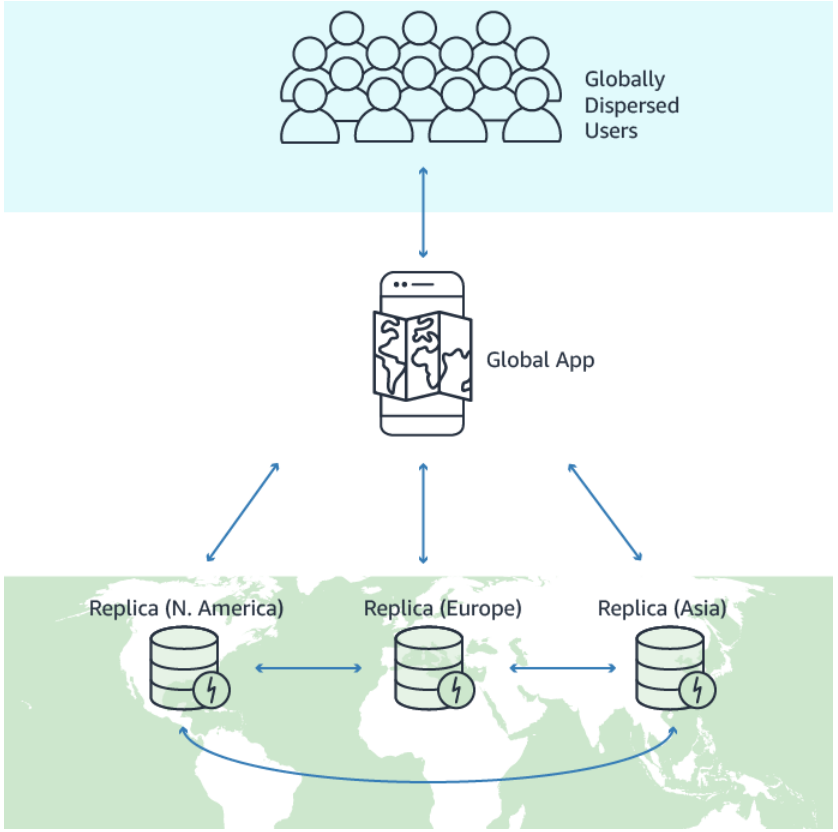


Dynamo Replication

Dynamo Replication

Global Tables builds upon DynamoDB's global footprint to provide a fully managed, multi-region, and multi-master database that gives faster local read and write performance for massively scaled global applications

Global Tables replicates our Amazon DynamoDB tables automatically across our choice of AWS regions



Backup and Restore

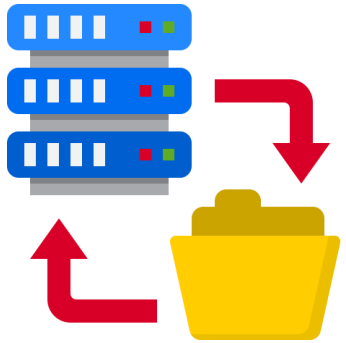
Backup and Restore

On-demand Backup

Restore

Point-in-time Recovery

When we create an on-demand backup, a time marker of the request is cataloged. The backup is created asynchronously by applying all changes until the time of the request to the last full table snapshot



No limit to the number of on-demand backups that can be taken

Backup and Restore

On-demand Backup

Restore

Point-in-time Recovery

We restore a table without consuming any throughput on the table; we can do a full table restore from our DynamoDB backup

While restore, we can change these settings:

- Global secondary indexes (GSIs)
- Local secondary indexes (LSIs)
- Billing mode
- Provisioned read and write capacity
- Encryption settings

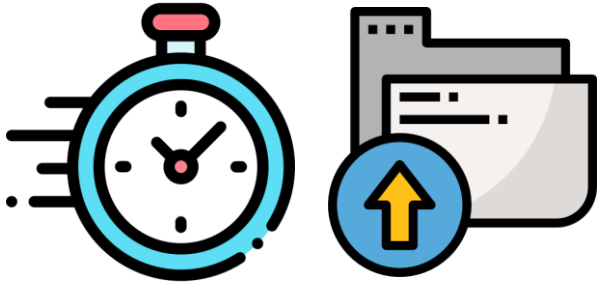
Backup and Restore

On-demand Backup

Restore

Point-in-time Recovery

We can enable continuous backups using point-in-time recovery. This helps protect our DynamoDB tables from accidental write or delete operations



For example, suppose, a test script writes accidentally to a production DynamoDB table. With point-in-time recovery, we can restore that table to any point in time during the last 35 days

DynamoDB Best Practices

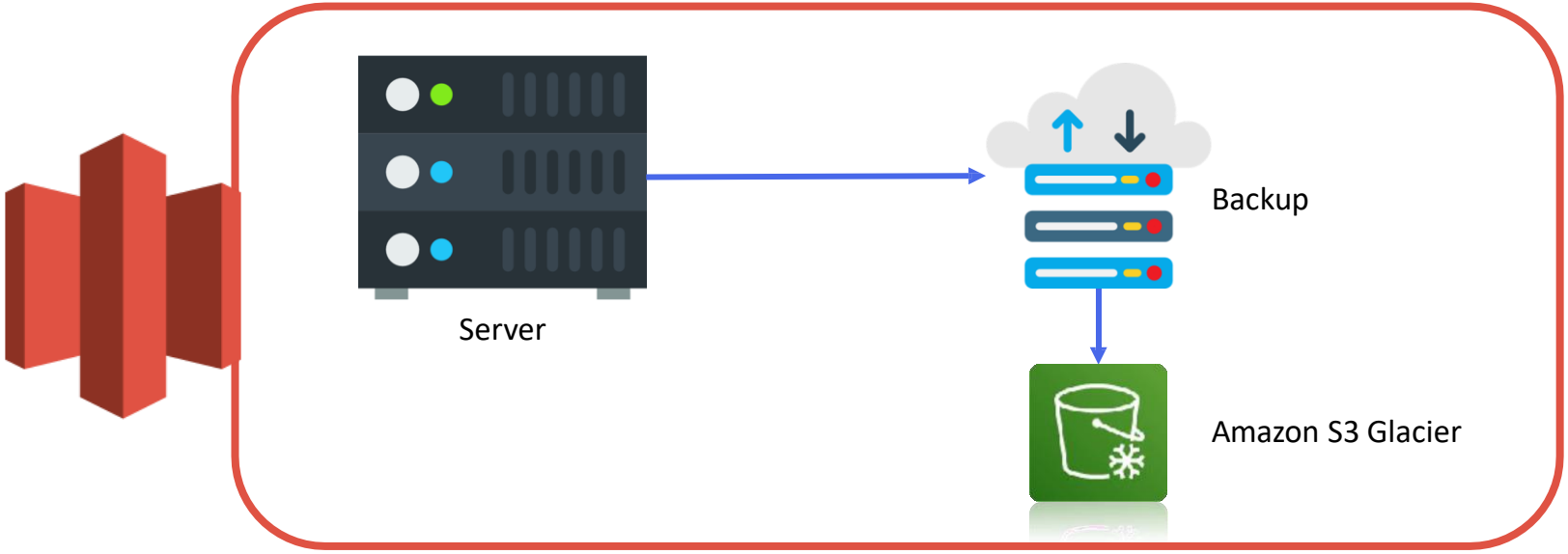
DynamoDB Best Practices

- Keep the number of indexes to a minimum
- Set different provisioned read capacity for different readers
- Store large attribute values in Amazon S3 and store the object identifier in DynamoDB
- Maintain as few tables as possible in a DynamoDB application

What is S3 Glacier?

What is S3 Glacier?

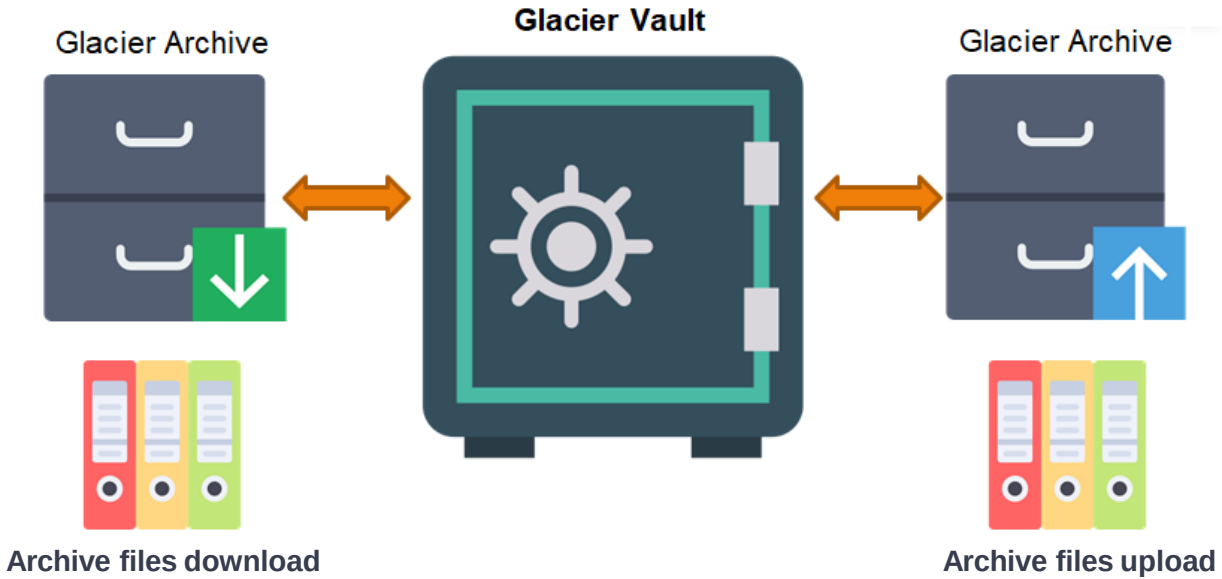
Amazon Glacier is designed for long-term data storage for cold data that is not needed to be retrieved very frequently and quickly. It is also a backup service used to back up the infrequently accessed data in S3



Glacier Vaults and Archives

Glacier Vaults and Archives

Archives and Vaults are at the core of the Glacier data model



Glacier Vaults and Archives

The actual data is stored in archives. We can store any type of data in an archive, such as images or video. We can directly upload a single file and create an archive, or we can ‘TAR’ or ‘ZIP’ multiple files and upload it as a single archive

Vaults are containers where we can store multiple archives. 1,000 vaults are allowed for a single AWS account

Use Cases of S3 Glacier

- **Applications that have a lot of multimedia data** coming in can use Amazon Glacier as the amount of data can sometimes reach petabytes too. For example, Twitter needs Glacier
- **Research and Development** organizations use huge datasets for their scientific analysis. They need a way to store all the input, training, and output data

Introduction to R D S

Introduction to RDS

Amazon RDS (Relational Database Service) is used to set up, manage, and scale a relational database instance in the cloud. Amazon RDS manages backups, software patching, failure detection, and many more tasks

Amazon Aurora

MariaDB

PostgreSQL

Microsoft SQL Server

MySQL

ORACLE

Basics of RDS

Basics of RDS

A **DB instance** is an isolated database environment, running in the cloud. It is the basic building block of Amazon RDS

Once we launch our instance, we will get an **Endpoint and a Port number** that can be used to connect with it

Hands-on: Creating a MySQL Database

+919030485102

rganesh0203@gmail.com

https://topmate.io/rganesh_0203