International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

A Guide to DynamoDB Global Tables for Effective Multi-Region Replication

Krishna Mohan Pitchikala

Abstract: ***In today's world, businesses must effectively reach a global audience. For corporations whose clients are scattered across the globe, ensuring quick and steady access to information is not only an edge over competitors but a basic requirement. This is where data replication comes into play. To achieve this, tables in a database should be copied and saved in several locations worldwide so that people can have a low latency experience from wherever they may be physically located knowing that their demand is always met. DynamoDB-Amazon's fully managed NoSQL database service-have earned its repute as the best option available for effective global data management by many organizations. Its scalability, reliability, and performance make DynamoDB a popular choice for organizations needing efficient global data management. Large companies like Samsung, Netflix, and Dropbox use DynamoDB to deliver consistent user experiences worldwide [1]. Replicating data across different regions while keeping it consistent, reducing delays, and ensuring it works even if there are issues can be tough. DynamoDB's built-in feature, Global Tables, makes this process easy with little to no human effort. Once set up, the owners only need to focus on their business logic, as DynamoDB takes care of the replication process. This paper gives a detailed guide to understand, set up, and manage DynamoDB Global Tables for efficient multi-region data replication.***

Keywords: global audience, data replication, DynamoDB, multi-region data, Global Tables

1. Definitions

DynamoDB

DynamoDB is a NoSQL database service managed by Amazon. They scale better than SQL databases with large volumes of data, easily store different types of data and allow for simple updating which makes them ideal for global systems. SQL databases are not very friendly to developers because they do not align well with application models and their integration into cloud infrastructure requires downtime during scaling whereas NoSQL does not. These features make NoSQL databases highly suitable for modern, dynamic applications [2].

In DynamoDB, data elements are called items, which are collections of attributes that uniquely identify them. A table is a collection of these items. Typically, DynamoDB tables are deployed in a single AWS region, with data replicated across multiple availability zones for reliability. However, this setup can cause high latency for users far from the region. To address this, DynamoDB introduced Global Tables, which replicate data across multiple regions to reduce latency for users located in different parts of the world.

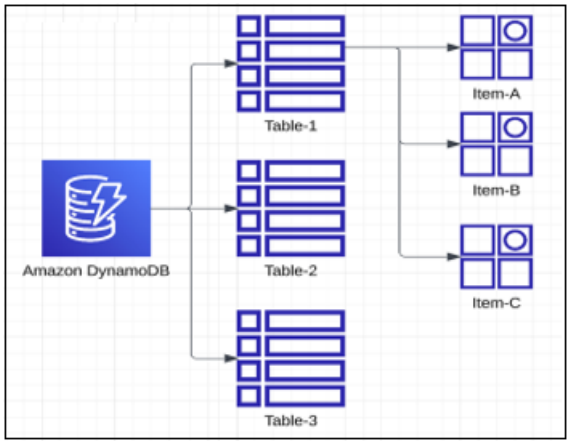

Figure 1: Amazon DynamoDB Structure Overview

Dynamo DB Global Tables

DynamoDB Global Tables extend the capabilities of DynamoDB Tables to support multi-region replication, making applications highly available and providing low-latency access for users worldwide. This reduces latency by keeping data close to users, improving performance for a fast and reliable experience. Global Tables also offer high availability and fault tolerance with automatic failover and data replication, ensuring continuous operation even if parts of the system fail.

Understanding Data Replication with Global Tables

Data replication is the process of copying data from one location to another to ensure that the same data is available in multiple places. This process is crucial for enhancing data availability and reliability across different systems and regions. If one system fails due to hardware issues, or other problems, users can still access the replicated data, minimizing downtime and data loss.

There are multiple benefits of data replication, one of which is availability and accessibility. For global applications with users in different locations, data replication helps reduce latency by fetching data from the nearest location. It also improves load balancing by distributing data across multiple servers, which allows the system to handle more requests efficiently and enhances overall performance.

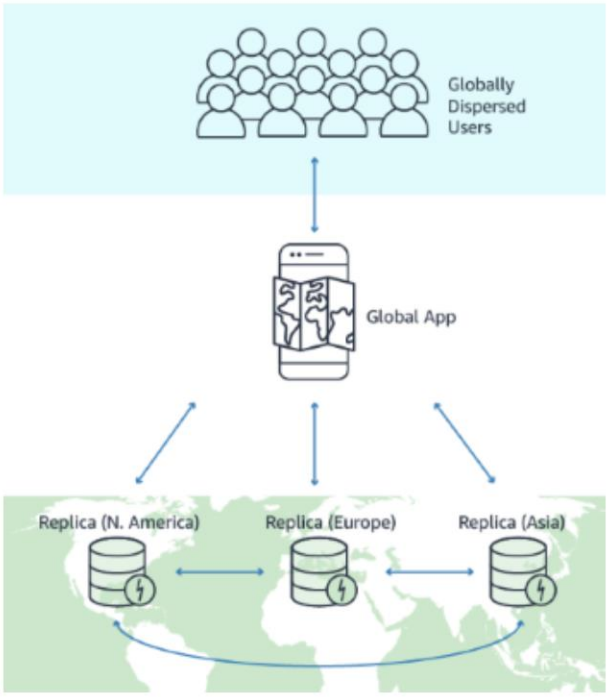

Figure 2: How Replication works [3]

Recognizing the use cases and benefits of data replication, DynamoDB introduced a native feature called Global Tables in 2017. Global Tables offer multi-region data replication, ensuring high availability and low latency. Once replication is enabled, DynamoDB manages everything, so users don't need to worry. All tables in replicated regions become active, allowing data to be read from the nearest DynamoDB instance, improving performance and accessibility. You can easily turn any table into a Global Table with one click, and you can also add or remove regions just as easily. The only requirement is that the table must use either on-demand capacity mode or, if using provisioned capacity mode, auto-scaling must be enabled.

DynamoDB global table replicas are connected but are not dependent. This means they share data, but their operation doesn't rely on each other. If an issue occurs in AWS region A, it won't affect the replicas in other regions. The writes to DynamoDB are eventually consistent, meaning there will be conflicts if same item is updated in multiple regions. In such cases, the "last writer wins" rule applies, where the latest update overwrites others. Therefore, we should avoid writing to the same item in different regions simultaneously unless the order of updates doesn't matter.

With global tables, writing to the table costs more because the data is copied to other replicas. Instead of a regular write capacity unit or request unit, replicated write capacity unit or replicated request unit is used depending on the table's capacity mode. Read costs remain the same. Global Tables for DynamoDB improve application's availability, durability, and fault tolerance. In disaster recovery terms, the database can achieve a near-zero recovery time and a recovery point of about one second. Additionally, AWS guarantees a 99.999% database uptime with global tables [4].

2. Converting DynamoDB Tables to Global Tables

DynamoDB offers several methods to convert a single-region table into a Global Table. Here are some ways to do it:

1) **Using AWS Console**
a) Go to the DynamoDB console
b) Choose the table you want to convert to a Global Table.
c) Click on the "Global Tables" tab.
d) Click on "Create Replica" and select the AWS regions where you want to replicate your table.
e) Follow the prompts to confirm the addition of regions.

2) **Using AWS CLI:**
a) Open your terminal and run the following command to convert the existing table into Global table [8]

```
aws dynamodb update-table \
  --region <existingRegion> \
  --table-name <tableName> \
  --replica-updates '[{"Create":{"RegionName": "<replicationRegion>"}}]'
```

3) Using CDK:
a) Add the property replicationRegions to the CDK where the table is created [9]
b) Example code would look like

```
const globalTable = new dynamodb.Table(this, "<tableName>", {
partitionKey: { name: "<partitionKey>", type: dynamodb.AttributeType.STRING }
replicationRegions: ["<replicationRegion>"],
});
```

4) Using CloudFormation:
a) The AWS::DynamoDB::GlobalTable resource enables you to create and manage a Version 2019.11.21 global table.
b) You cannot convert a resource of type AWS::DynamoDB::Table into a resource of type AWS::DynamoDB::GlobalTable by changing its type in your template. Doing so might result in the deletion of your DynamoDB table.
c) You can instead use the GlobalTable resource to create a new table in a single Region. This will be billed the same as a single Region table. If you later update the stack to add other Regions, then Global Tables pricing will apply [10].
d) Example code would look like

```
Type: AWS::DynamoDB::GlobalTable
Properties:
  AttributeDefinitions:
    - AttributeDefinition
  BillingMode: String
  GlobalSecondaryIndexes:
    - GlobalSecondaryIndex
  KeySchema:
    - KeySchema
  LocalSecondaryIndexes:
    - LocalSecondaryIndex
  Replicas:
    - ReplicaSpecification
  SSESpecification:
    SSESpecification
  StreamSpecification:
    StreamSpecification
  TableName: String
  TimeToLiveSpecification:
    TimeToLiveSpecification
  WriteOnDemandThroughputSettings:
    WriteOnDemandThroughputSettings
  WriteProvisionedThroughputSettings:
    WriteProvisionedThroughputSettings
```

Each of these methods provides a flexible way to convert your single-region DynamoDB table into a Global Table, ensuring that you can choose the one that best fits your workflow and technical preferences.

3. Conclusion

DynamoDB Global Tables are ideal for creating fault-tolerant applications and ensuring a high-quality user experience. They keep application data available and consistent across multiple AWS regions, ensuring data is safe even if one region fails. They are also cost-efficient, as costs are incurred only based on usage, making them economical for both small and large systems. Setting up Global Tables is straightforward with various AWS tools, and they do not disrupt existing applications, facilitating a smooth transition. Although they handle data conflicts effectively, it is advisable to design applications to minimize conflicts when writing data simultaneously across multiple regions.

References

[1] https://aws.amazon.com/dynamodb/customers/
[2] https://www.mongodb.com/resources/basics/databases/nosql-explained/advantages
[3] https://aws.amazon.com/dynamodb/global-tables/
[4] https://repost.aws/questions/QUEhA2ULoQSG-giUoZ5CA4CA/dynamodb-race-conditions-with-transactwriteitems-and-condition-check
[5] https://aws.amazon.com/dynamodb/
[6] https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html
[7] https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GlobalTables.html
[8] https://aws.amazon.com/blogs/aws/new-convert-your-single-region-amazon-dynamodb-tables-to-global-tables/
[9] https://dev.to/aws-builders/aws-cdk-and-amazon-dynamodb-global-tables-3p0b
[10] https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-dynamodb-globaltable.html