



Amazon DynamoDB

Database Fundamentals

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Welcome to Amazon DynamoDB.

What you will learn

At the core of the lesson

You will learn how to do the following:

- Describe the purpose and function of Amazon DynamoDB.

Key terms:

- Nonrelational database
- NoSQL database
- Table item and attribute
- Primary, partition, and sort keys
- Table partition
- Global tables



In this module, you will discover key concepts that are related to database solutions. The module will highlight the differences between relational and nonrelational databases in general and explore concepts and terminology related to the Amazon DynamoDB NoSQL database.

You will learn how to describe the purpose and function of DynamoDB.



Introduction to DynamoDB

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Welcome to an introduction to DynamoDB, a NoSQL Amazon Web Services (AWS) database service.

Relational and nonrelational NoSQL databases: Comparison

Relational databases

- Data is stored in tables by using predefined columns of specific data types.
- Relationships can be defined between tables by using table foreign keys.
- Better performance is achieved by adding compute or memory capacity to a single database server.

Nonrelational NoSQL databases

- Data is stored in tables with a flexible column structure.
- Each item stored in a table can have a different number and type of data elements.
- Better performance is achieved by adding a new server to an existing pool of database servers.

Relational databases are used to store data for a wide variety of applications. Relational databases use **normalized** database table designs, which means that data is stored across multiple tables with defined relationships between the tables.

The data stored in a relational database table must match the design for that table. Every record layout is essentially the same; only the data is different.

Most relational databases run on a single, large database server with a fixed amount of memory and compute capacity. If better performance is needed, a bigger server with more memory or compute capacity is used to replace the existing server.

Nonrelational, or NoSQL, databases are more flexible in the way that they store data.

Data for one record in the database can be of a different layout, type, and size from data for another record in the database.

NoSQL databases run on a group of servers called a cluster, each with its own fixed amount of memory and compute capacity. If better performance is needed, additional servers can be added to the existing cluster group.

DynamoDB is a NoSQL database

What is DynamoDB?

A fully managed, serverless, **key-value NoSQL database** that does the following:

- Improves performance by keeping data in memory
- Keeps data secure by encrypting data at rest
- Protects data with backups and automated copying of data between AWS Regions



DynamoDB offers the following advantages:

- Fully managed service: AWS manages all of the underlying compute and storage needed to store your data.
- Scalability: DynamoDB automatically adds more compute and storage capacity as your data grows.
- Redundancy: DynamoDB copies your data across multiple AWS Regions to avoid data loss.
- Recoverability: DynamoDB can restore your data from automatic backup operations.
- Low latency: Data can be read from a DynamoDB table in a few milliseconds.
- Security: You can use DynamoDB with AWS Identity and Access Management (IAM) to control access to DynamoDB tables.
- Flexibility: DynamoDB can store several types of data, and each record can have varying numbers and types of data. JSON is one popular way to store data in DynamoDB.



How it works

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Next, you'll look at how DynamoDB works.

Key concepts: Tables

Core concepts behind DynamoDB

- Tables In DynamoDB: Similar to relational database systems, DynamoDB stores data in tables.
 - The table name and primary key must be specified at table creation.
 - Each DynamoDB table has at least one column that acts as the primary key.
- The primary key is the only data that is required when storing a row in a DynamoDB table. Any other data is optional.

Example

- Suppose that you want to create a DynamoDB table to store data about your Friends. The Friends table can do the following:
 - Use an ID number to identify each friend.
 - Use this ID number as the table's primary key.



Each DynamoDB table must have a primary key.

Each row in a DynamoDB table must have a distinct value for the primary key.

Key concepts: Attributes

Core concepts behind DynamoDB

- A column in a DynamoDB table is called an attribute.
 - An attribute represents a fundamental data element, something that does not need to be broken down any further.
 - An attribute is similar in concept to table columns or fields in a relational database.
- A primary key can consist of one or two attributes.

Example

- The Friends table will use the FriendID attribute for the primary key, but other attributes might include the following:
 - Name
 - Hobbies
 - Favorite colors



Relational databases use tables, made up of columns, to store data.

DynamoDB uses tables, made up of attributes, to store data.

The primary key for a DynamoDB table can consist of one or two attributes.

Key concepts: Items

Core concepts behind DynamoDB

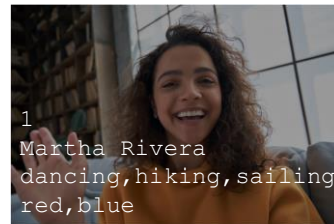
- An item is a group of attributes that is uniquely identifiable among all of the other items.
 - Each item consists of one or more attributes.
 - Each item is uniquely identified by its primary key attribute.
 - This concept is similar to a table row in a relational database.

Difference from a relational database

- The number and type of non-primary key attributes can be different for each item in a table.

Example

- Data for each friend is stored as an item.
 - Each friend has an ID plus any other attributes that you want to store about that person.
 - The amount and type of data stored for each attribute can vary for each friend.



Relational databases use tables, made up of columns, to store data as rows.

DynamoDB uses tables, made up of attributes, to store data as items.

Each item stored in a DynamoDB table does not need to have the same number or type of attributes.

This property is a difference between NoSQL tables and relational database tables. With relational tables, the number and type of data in each row must match the table's structure.

Key concepts: Primary keys

Simple primary key

- The primary key consists of one attribute.
- The attribute is called the partition key or hash key.



One attribute

Composite primary key

- The primary key consists of two attributes.
- The first attribute is the partition key or hash key.
- The second attribute is the sort key or range attribute.




Two attributes

REMEMBER: Primary keys uniquely identify each item in the table, so no two items can have the same primary key.

A primary key that is based on a single table attribute is called a simple primary key. The attribute that the primary key is based on is called the partition key or hash key.

A primary key that is based on two table attributes is called a composite primary key. The first attribute in a composite primary key is called the partition key or hash key. The second attribute in the composite primary key is called the sort key or range attribute.

Whether simple or composite, each item must have a unique value for its primary key attributes.



The concept of partitioning

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Next, you'll look at how partitioning works.

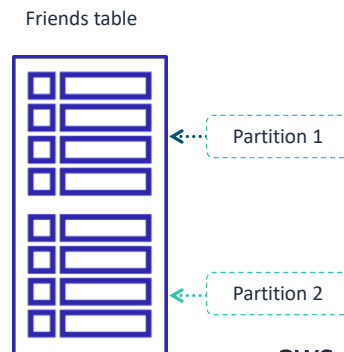
Key concepts: Partitions

Core concepts behind DynamoDB

- DynamoDB tables store item data in partitions.
 - Table data is partitioned and indexed by the primary key.
 - Each table has one or more partitions.
 - New partitions are added as data is added to the table.
 - Tables whose data is distributed evenly across multiple partitions generally deliver the best performance.
- The partition in which an item's data is stored is determined by its primary key attribute.

Example

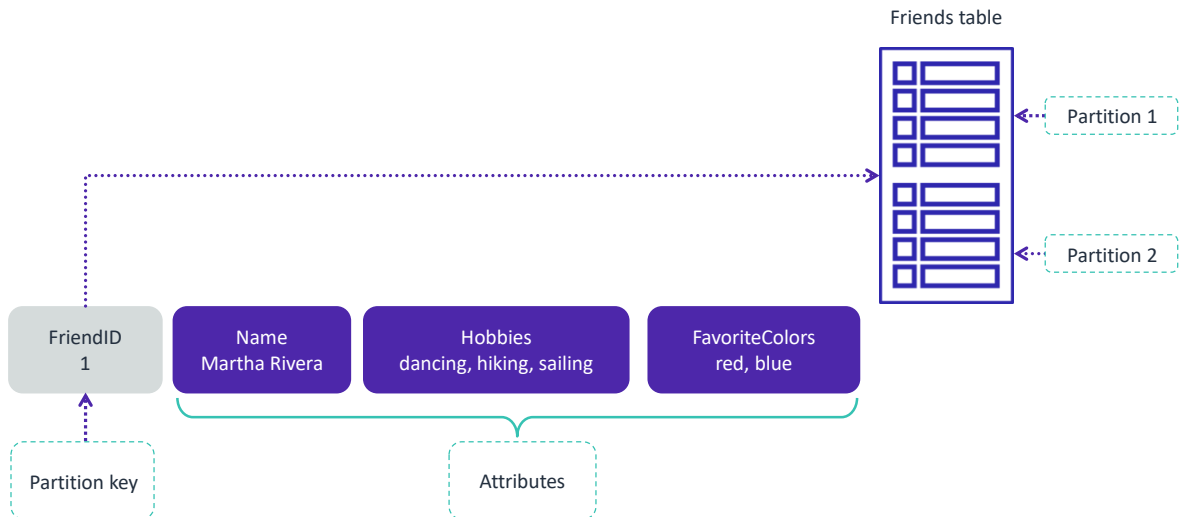
- The value for the FriendID attribute will determine which partition in the Friends table will store the data for that person.



DynamoDB tables store data in partitions. Each table comprises one or more partitions.

DynamoDB automatically adds new partitions to a table when the existing partitions fill with data.

Key concepts: Storing items in partitions



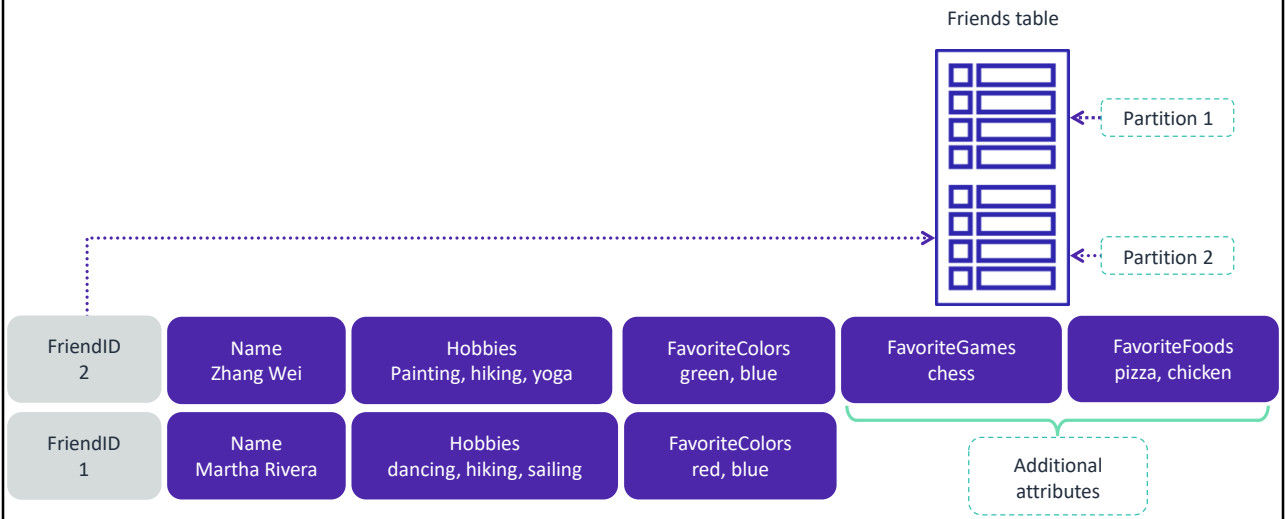
This example shows the item data for Martha Rivera being stored in the Friends table.

When choosing the partition to store the data, DynamoDB applies a function to the partition key attribute of the primary key. In the case of the Friends table, the primary key is a simple primary key. Therefore, the partition key attribute is the same as the primary key, which is **FriendID**.

The value that the function returns will be in either **Partition 1** or **Partition 2** depending on the value for the **FriendID** attribute.

In this case, the function chose **Partition 1** to store Martha's data.

Key concepts: Storing attributes in partitions



This example shows the item data for another friend, Zhang Wei, which is being stored in the Friends table.

When choosing the partition to store the data, DynamoDB applies a function to the partition key attribute, **FriendID**. In this case, the function chooses **Partition 2** to store Zhang's data.

Notice that two additional attributes are being stored for Zhang, more than were stored for Martha. These extra attributes do not present a problem. If an item has a primary key value, DynamoDB permits each item to have different numbers and types of attributes.

DynamoDB global tables

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Next, you'll look at the way that DynamoDB global tables can be used to improve performance and data availability.

Using global tables

Core concepts of global tables

- Using the global table option creates a DynamoDB table that is automatically replicated across your choice of AWS Regions worldwide.
 - Deliver fast, local read and write performance for global applications.
 - Your applications can stay highly available in the unlikely event of isolation or degradation of an entire AWS Region.
- Global tables eliminate the difficult work of replicating data between Regions and resolving update conflicts between copies.

Example

- What if you have millions of friends all over the world who will use your Friends table?
 - Global tables will put the table data closer to your friends.
 - Being closer to the data means that your friends will experience faster query performance.



The DynamoDB global tables feature provides high availability and scalability across Regions.

A **global table** is a collection of one or more DynamoDB tables, which must all be owned by a single AWS account.

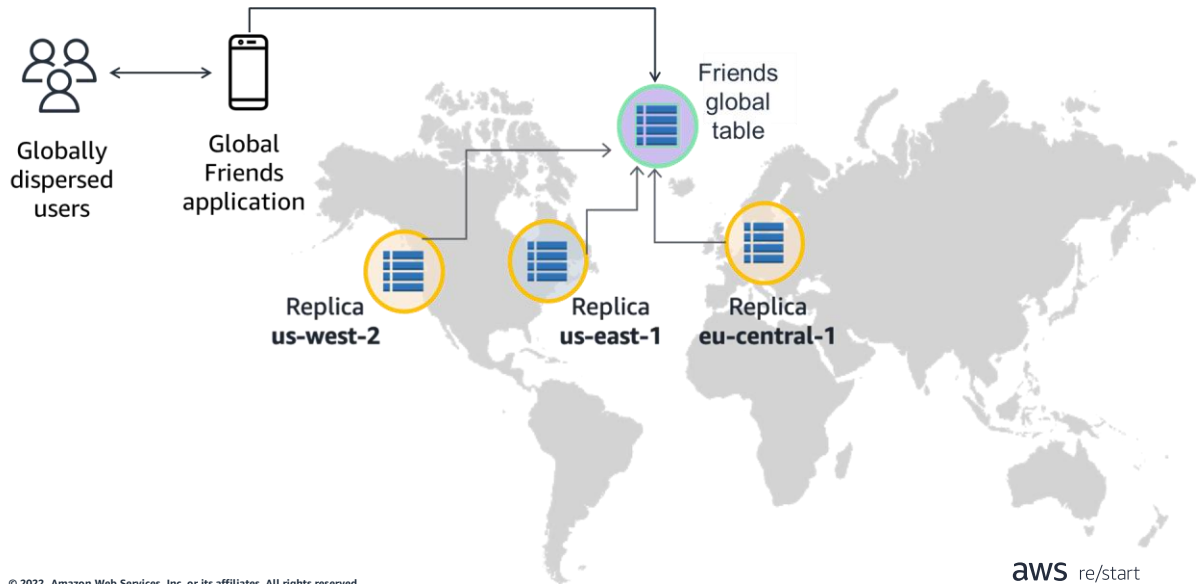
The tables in the collection are also known as replica tables. Each replica stores the same set of data items.

When you create a global table, you specify the AWS Regions where you want the table to be available.

DynamoDB performs all of the necessary tasks to create identical tables in these Regions. It automatically performs ongoing copying of data to all of the tables to keep their contents in sync.

DynamoDB global tables work well for large-scale applications with globally dispersed users by accessing the replica that is closest to them.

Global tables example



This example shows how globally dispersed friends benefit from DynamoDB global tables by using a mobile application.

Users can access the table replica that is geographically closest to them. This access enhances your friends' experience by providing better response times when reading and writing to the DynamoDB Friends table.

Checkpoint questions



Why are nonrelational (NoSQL) databases a good choice when you must get started quickly on a project?



Which table attribute is responsible for determining where item data is partitioned?



When you create items in a DynamoDB table, does each entry need to have the same attributes?

1. Why are nonrelational (NoSQL) databases a good choice when you must get started quickly on a project?

NoSQL databases have flexible schemas, so it is not necessary to do any data engineering. Developers can quickly and easily work with the data without needing to understand relationships between data.

2. What table attribute is responsible for determining where item data is partitioned?

The partition key is responsible for where the data is partitioned.

3. When you create items in a DynamoDB table, does each entry need to have the same attributes?

No. Because DynamoDB is a NoSQL database, individual items can have different attributes from other entries.

Key takeaways



- DynamoDB is a fully managed NoSQL database service.
- DynamoDB offers fast access to table data.
- DynamoDB has essentially no table size or throughput limits.
- Global tables reduce the difficulty of replicating data between Regions and resolving update conflicts.

This module includes the following key takeaways:

- DynamoDB is a fully managed NoSQL database service.
- DynamoDB offers consistent, single-digit millisecond latency at any scale.
- DynamoDB has essentially no table size or throughput limits.
- Global tables reduce the difficulty of replicating data between Regions and resolving update conflicts.



Thank you



© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. Corrections, feedback, or other questions? Contact us at <https://support.aws.amazon.com/#/contacts/aws-training>. All trademarks are the property of their owners.

Thank you for completing this module.