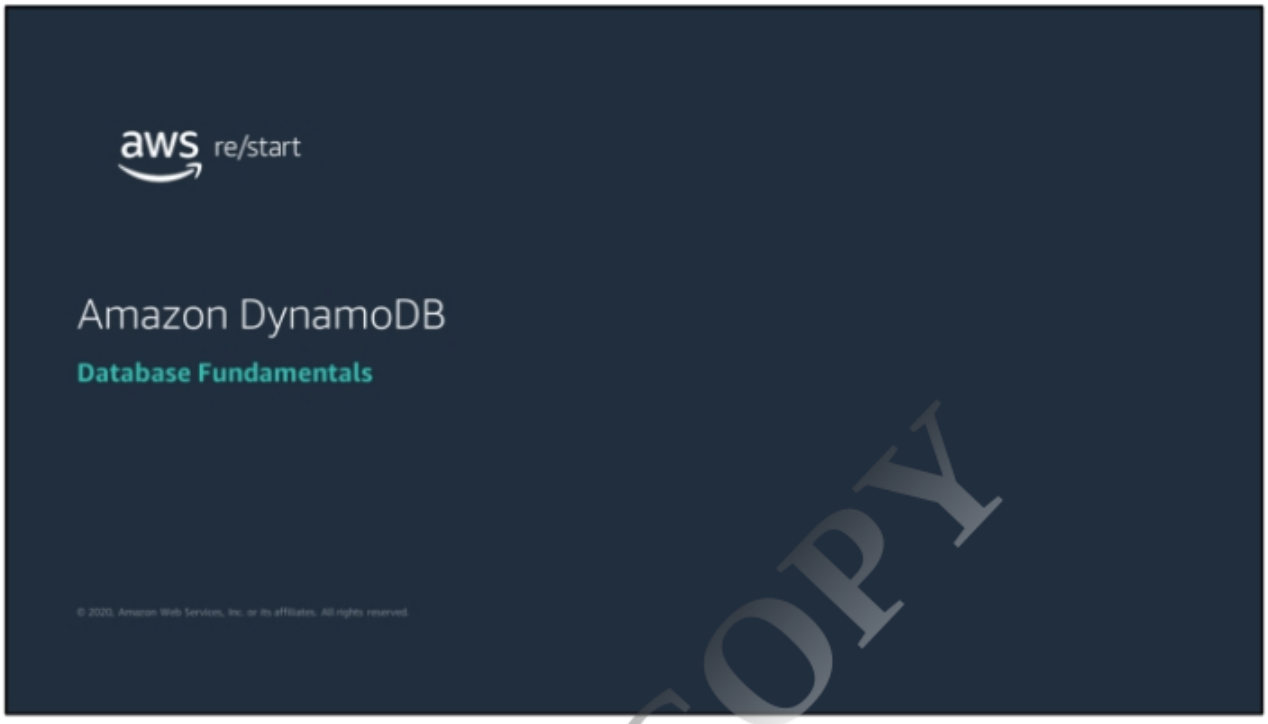


Welcome to Amazon DynamoDB.

What you will learn

At the core of the lesson

You will learn how to:

- Describe the details of the Amazon DynamoDB database service
- Highlight the difference between relational databases and nonrelational databases

Key terms:

- Amazon DynamoDB
- Partition
- Relational database
- Nonrelational database

2

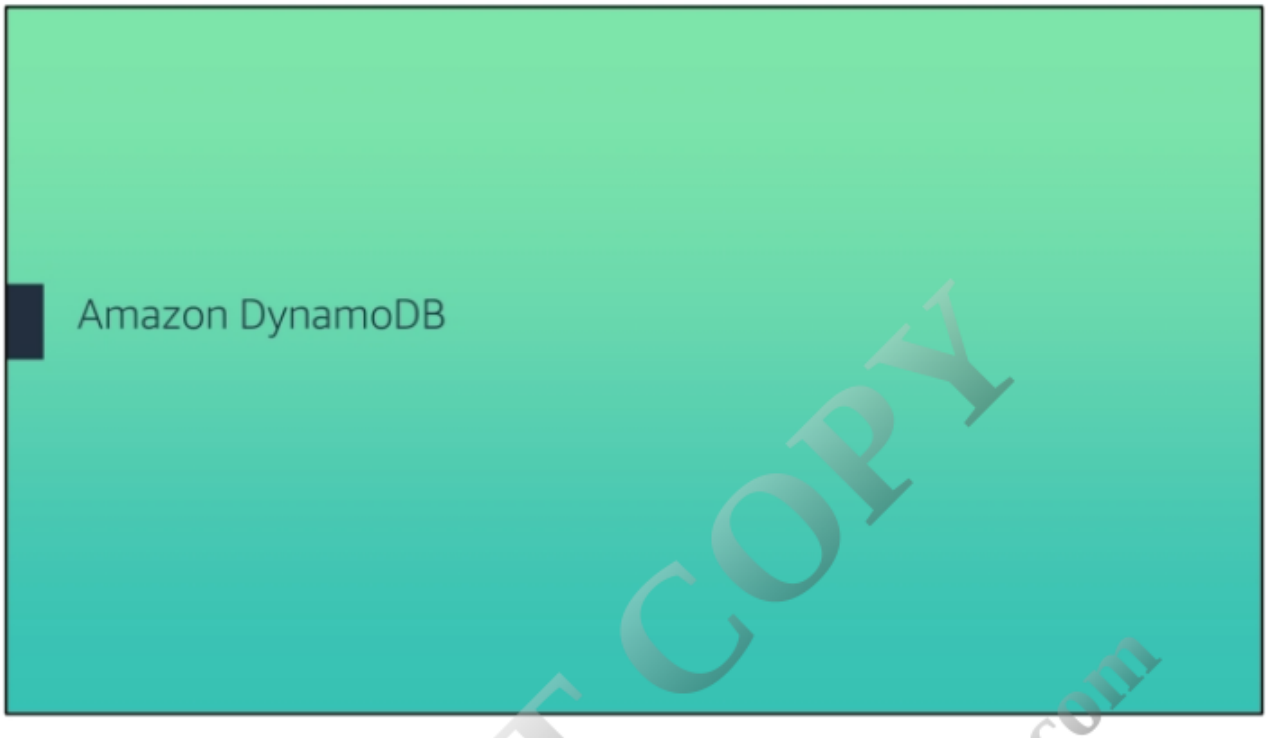
aws re/start

In this module, you will discover key concepts that are related to database solutions.

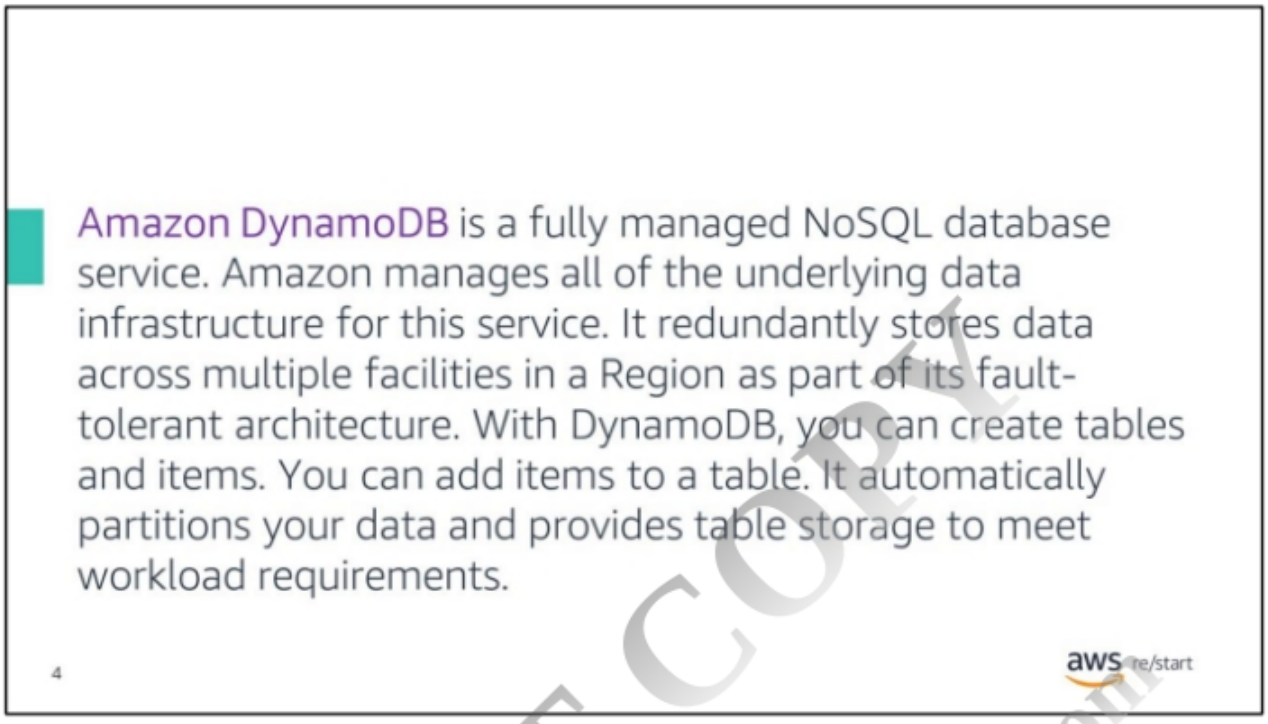
You will learn how to:

- Describe the details of the Amazon DynamoDB database service
- Highlight the difference between relational databases and non-relational databases

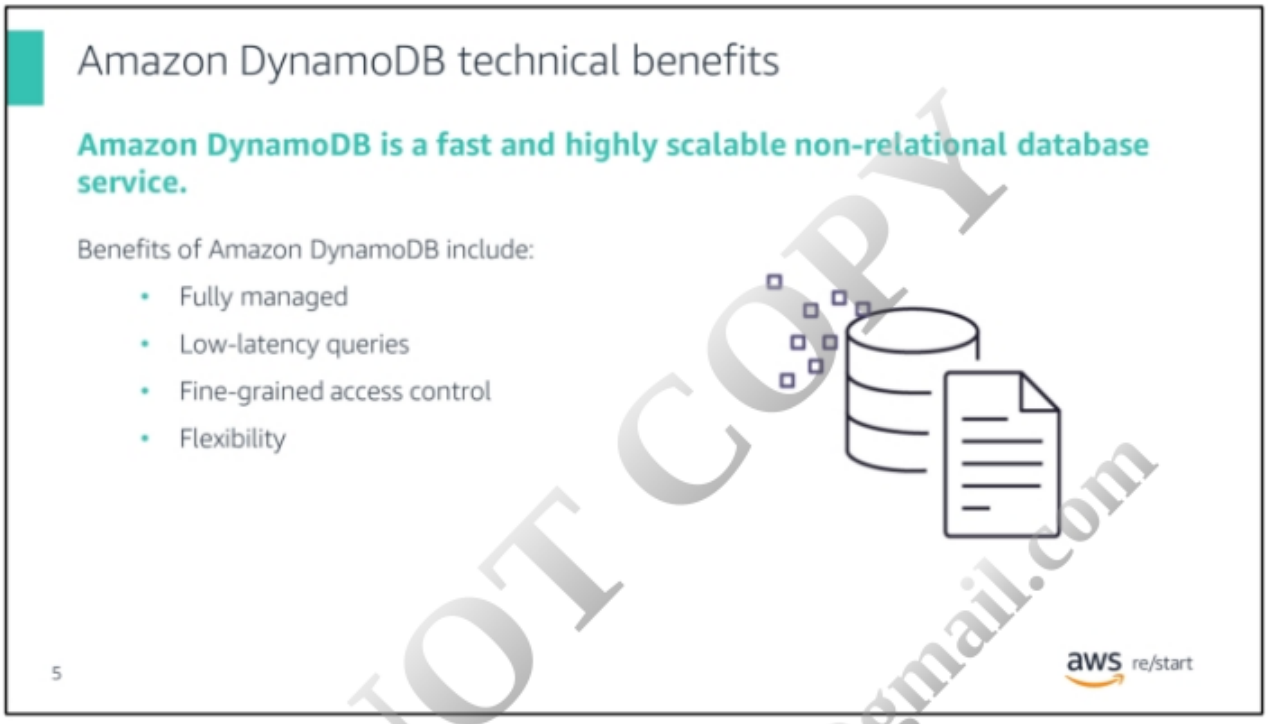
The goal of this module is to help you understand the database resources that are available to power your solution. You will also review the different service features that are available so that you can understand how different choices affect solution availability.



Welcome to an introduction to Amazon DynamoDB, which is a NoSQL AWS database service.



DynamoDB is a fully managed NoSQL database service. Amazon manages all of the underlying data infrastructure for this service. It redundantly stores data across multiple facilities in a Region as part of its fault-tolerant architecture. With DynamoDB, you can create tables and items. You can add items to a table. It automatically partitions your data and provides table storage to meet workload requirements.

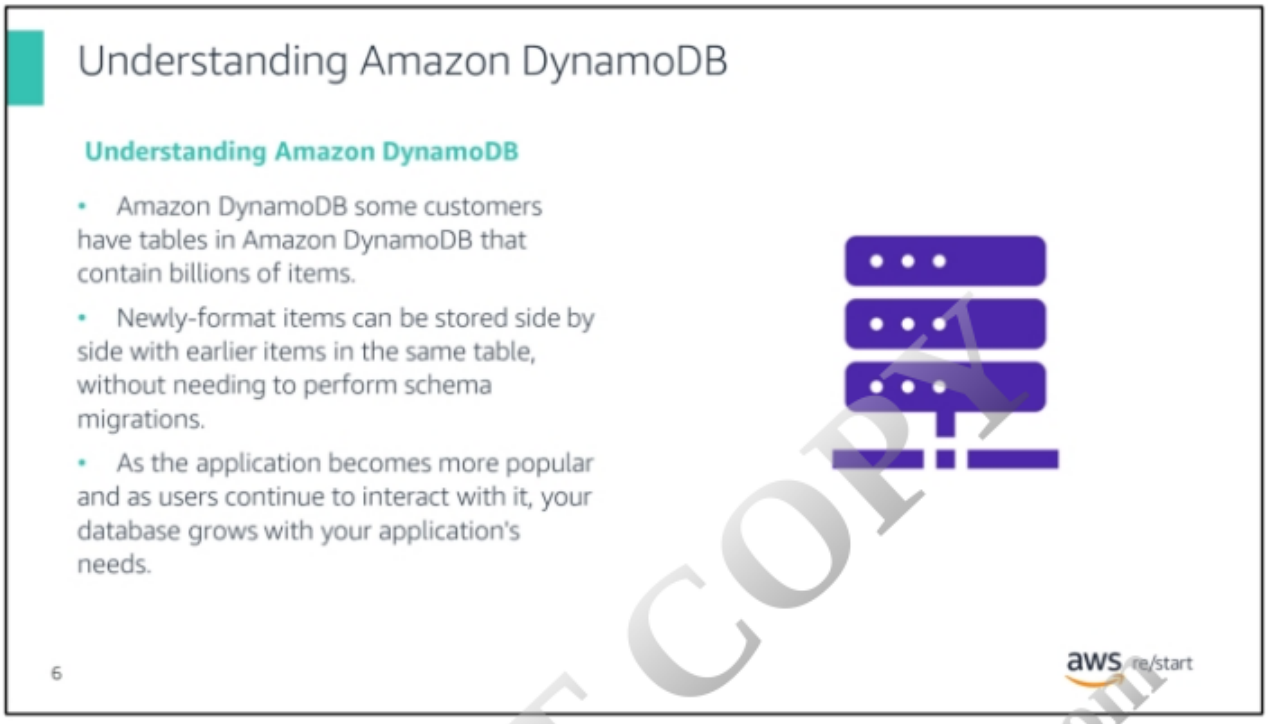


Amazon DynamoDB is a fast and flexible NoSQL database service that supports both the document and key-value store models. Its benefits include:

- *Fully managed service* – When you create a database table and set your target utilization for automatic scaling, the service automatically performs database management tasks. It handles hardware or software provisioning, setup and configuration, software patching, operating a distributed database cluster, and partitioning data over multiple instances as you scale. DynamoDB also provides point-in-time recovery, backup, and restore for all your tables.
- *Low-latency queries* – The average service-side latency for running a query is typically single-digit milliseconds. As your data volumes grow and application performance demands increase, DynamoDB adapts to meet these needs. It uses automatic partitioning and SSD technologies to achieve your throughput requirements and to deliver low latencies at any scale.
- *Fine-grained access control* – You can use DynamoDB with AWS Identity and Access Management (IAM) for fine-grained access control of users in your organization. You can assign unique security credentials to each user and control each user's access to services and resources.

- *Flexibility*: DynamoDB supports storing, querying, and updating data as JavaScript Object Notation (JSON) documents. This support makes it ideal for storing semistructured data and manipulating it by using JSON queries.

Additionally, you can also use Amazon DynamoDB with Amazon CloudWatch so that you can see throughput and latency request statistics.



With DynamoDB there are no practical limit exists for the number of items you can store in a table. For instance, some customers have production tables that contain billions of items.

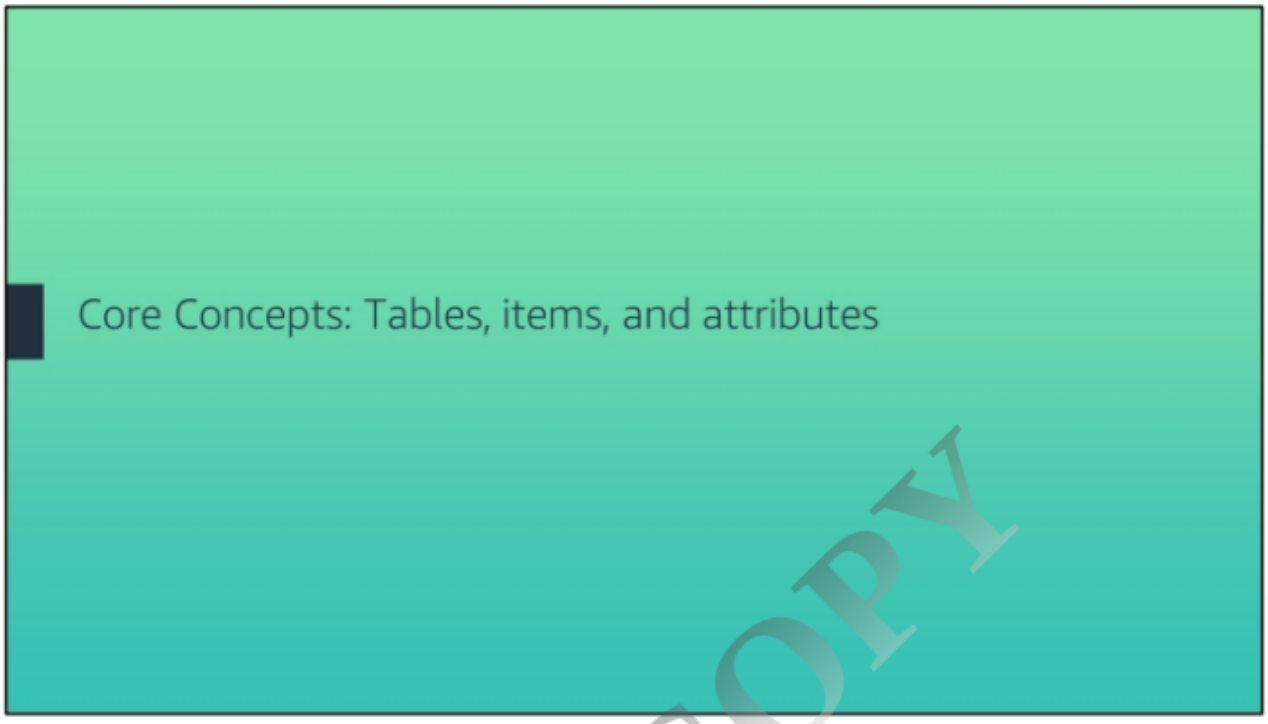
One of the benefits of a NoSQL database is that items in the same table can have different attributes. As a result, you have the flexibility to add attributes as your application evolves. Newer-format items can be stored side by side with earlier items in the same table, without needing to perform schema migrations.

As your application becomes more popular and as users continue to interact with it, your storage can grow with your application's needs. All the data in DynamoDB is stored on SSDs, and its simple query language enables consistent, low-latency query performance.

In addition to scaling storage, DynamoDB also enables you to provision the amount of read/write throughput that you need for your table. As the number of application users grows, DynamoDB tables can be scaled to handle the increased numbers of read/write requests with manual provisioning. Alternatively, you can enable automatic scaling so that DynamoDB monitors the load on the table and automatically increases or decreases the provisioned

throughput.

Some more key differentiating features include global tables that enable automatic replication across your choice of AWS Regions, encryption at rest, and item time-to-live (TTL).



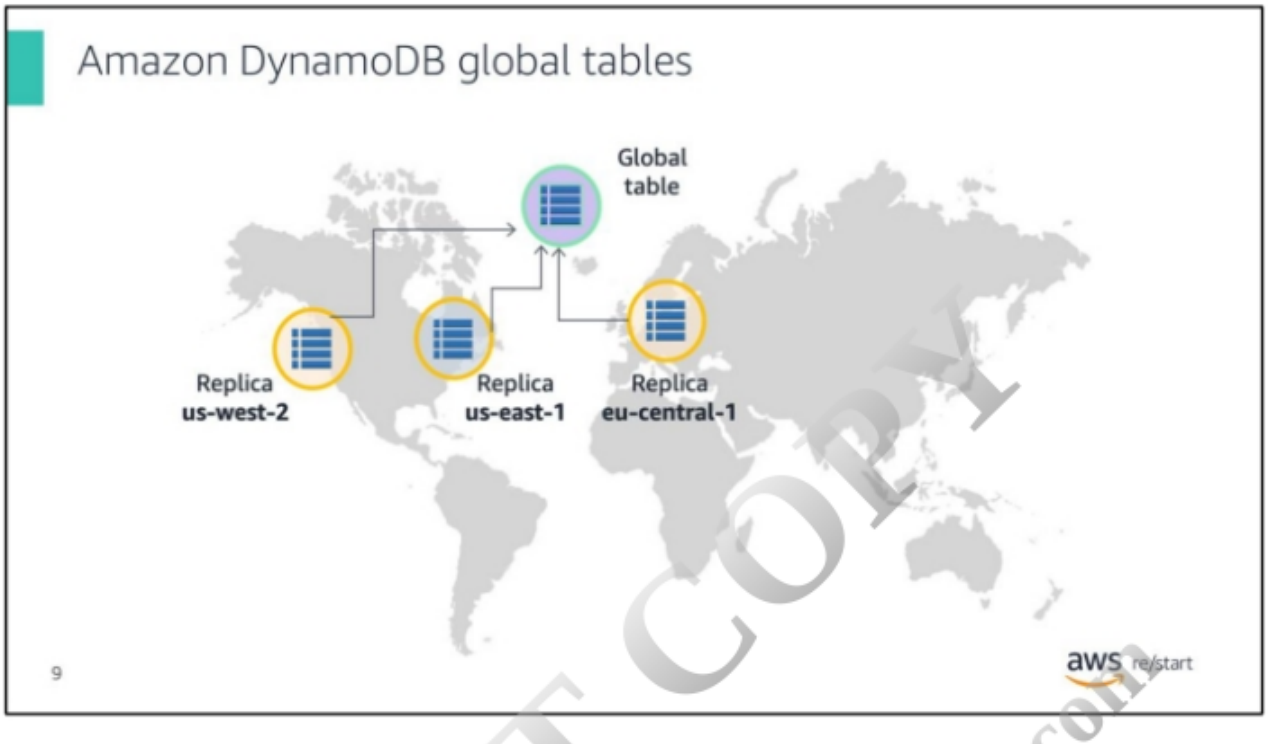
What are Amazon DynamoDB, which is a NoSQL database service.

Key Concepts

Core concepts behind DynamoDB:

- **Tables** – Similar to other database systems, DynamoDB stores data in tables. A *table* is a collection of data.
  - For example, think about an example table called *People* that you could use to store personal contact information about friends, family, or anyone else of interest.
- **Items** – Each table contains zero or more items. An *item* is a group of attributes that is uniquely identifiable among all of the other items.
  - For example, with the example *People* table, each item can represent a person.
- **Attributes** – 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.
  - For example, an item in a *People* table contains attributes called *PersonID*, *LastName*, *FirstName*, and so on.

aws re/start

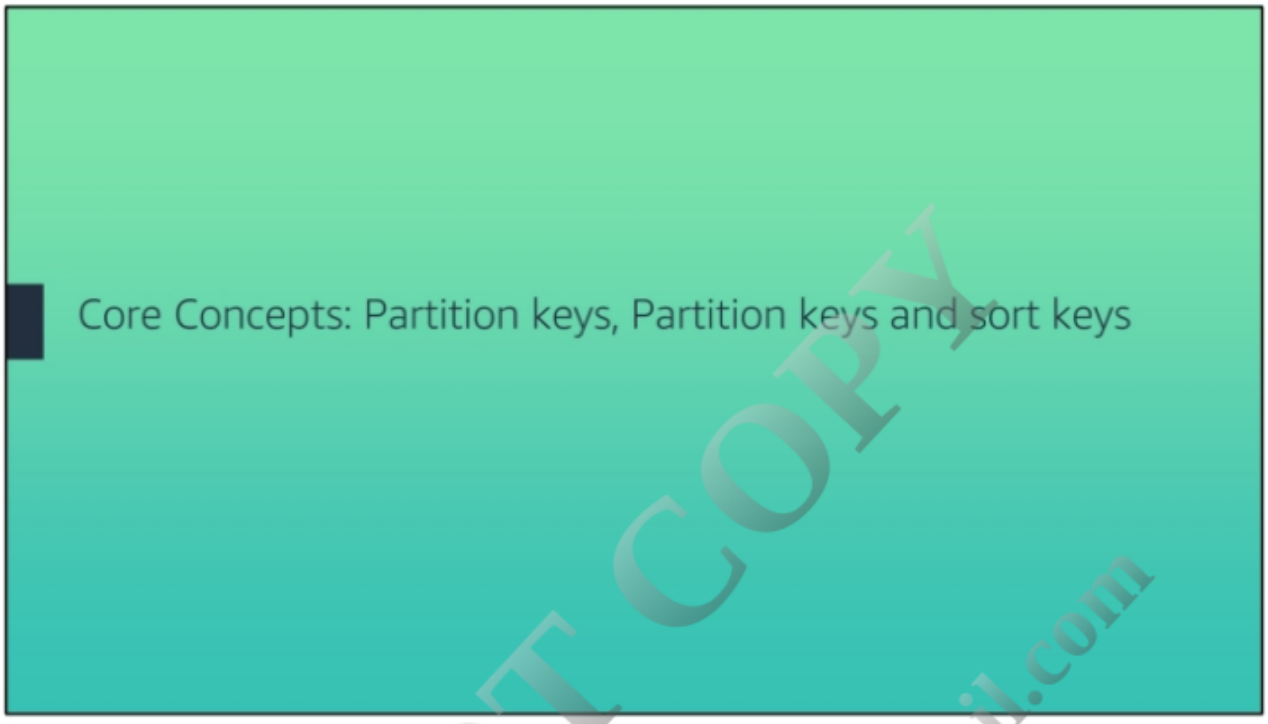


The DynamoDB global tables feature provides high availability and scalability across Regions.

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. A *replica table* (or *replica*) is a single DynamoDB table that functions as a part of a global table. Each replica stores the same set of data items. Any given global table can only have one replica table per Region. Every replica has the same table name and the same primary key definition.

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, and it propagates ongoing data changes to all of them.

DynamoDB global tables work well for large-scale applications with globally dispersed users. In such an environment, users expect fast application performance, which they obtain by accessing the replica that is closest to them. In addition, if one of the AWS Regions becomes temporarily unavailable, users can still access the same data in the other Regions.



What are Amazon DynamoDB, which is a NoSQL database service.

Tables and data

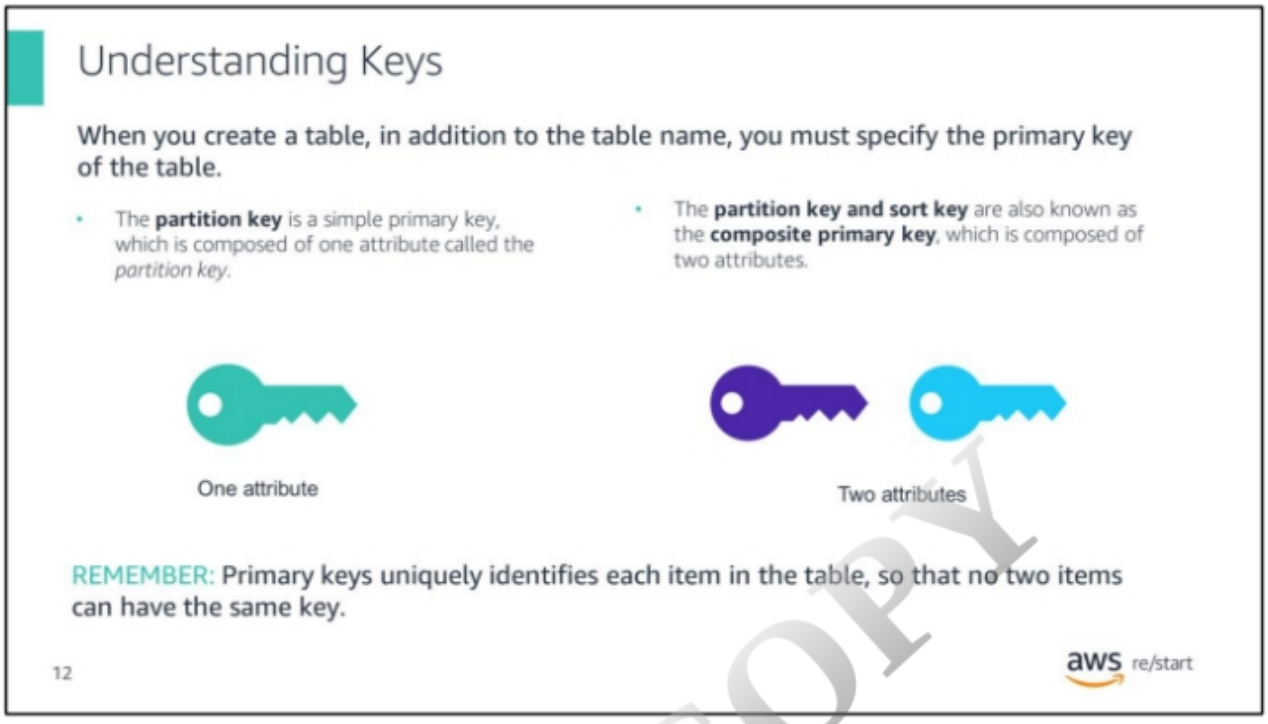
Core components

- DynamoDB supports two different kinds of primary keys:
  - Partition keys
  - Partition and sort keys

11

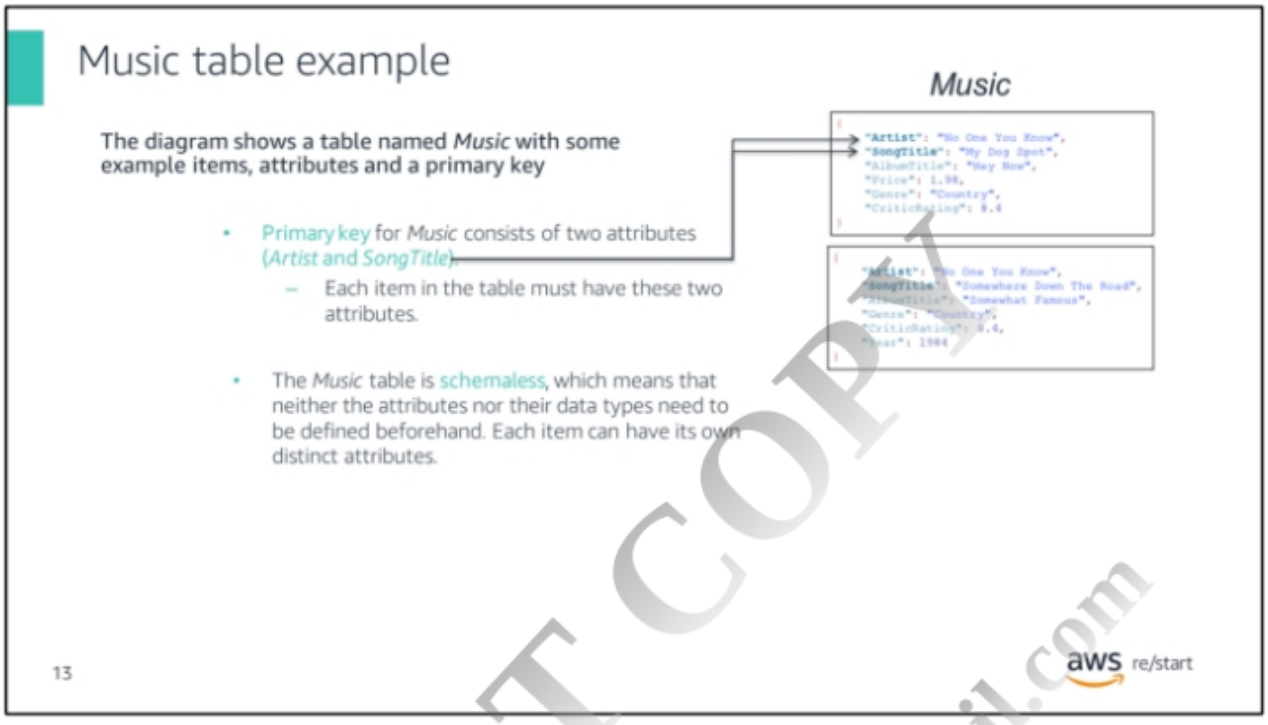
aws re/start

DynamoDB supports two different kinds of primary keys.



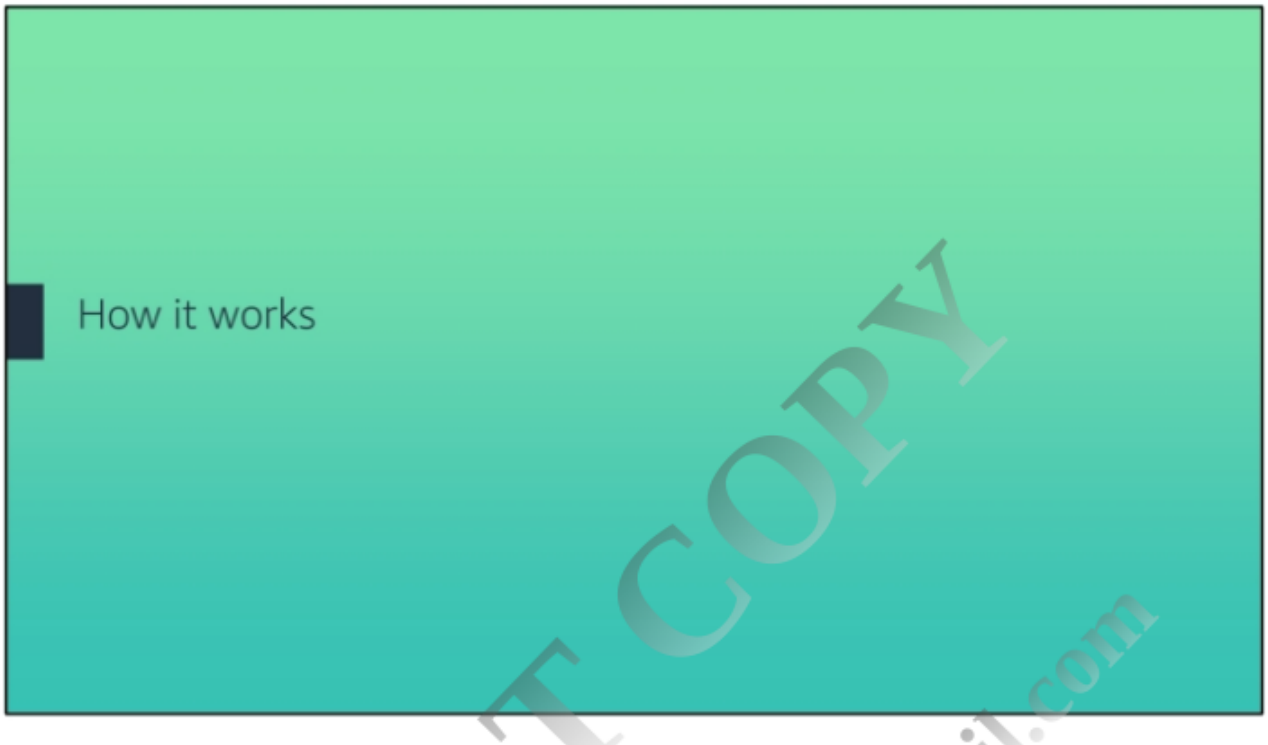
=The **partition key** is a simple primary key, which is composed of one attribute called the *partition key*. The partition key and sort key are also known as the **composite primary key**, which is composed of two attributes. To better understand what a composte key is consider the following example:
The *Music* table described in Tables, Items, and Attributes is an example of a table with a composite primary key (*Artist* and *SongTitle*).

To learn more about how DynamoDB works, refer to the Core Components of Amazon DynamoDB.

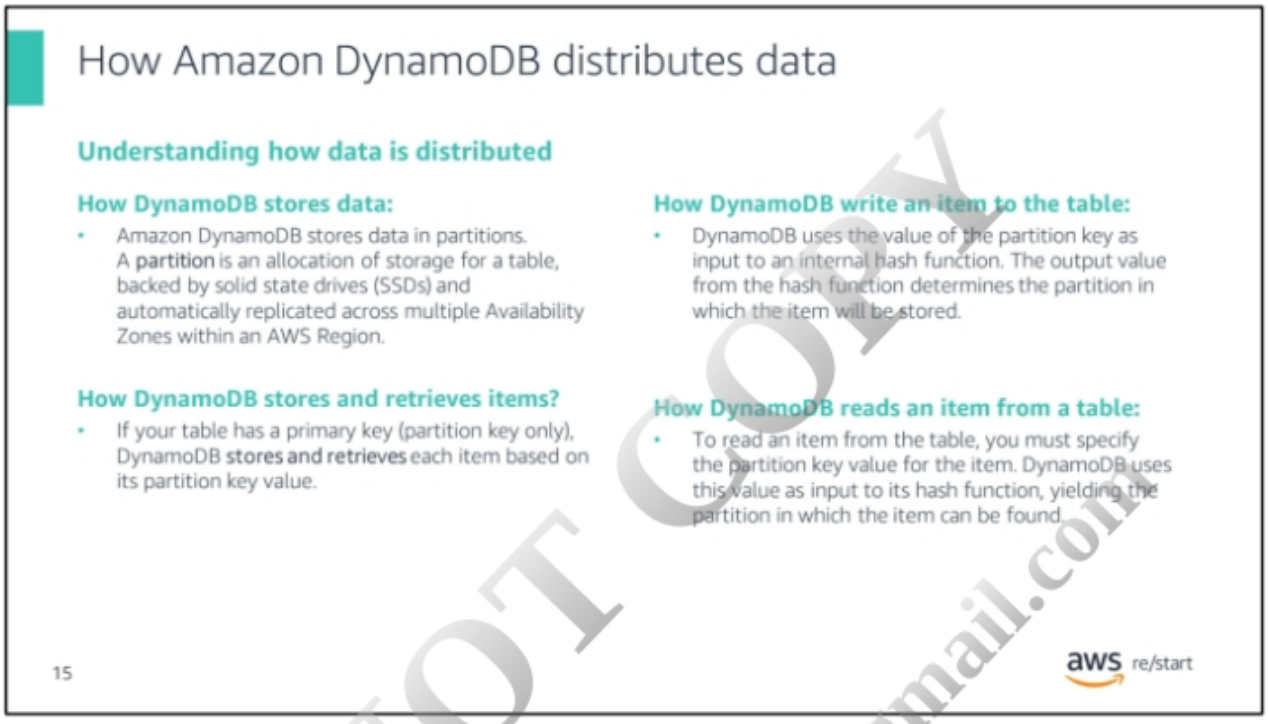


Consider a table named Music. The table has a primary key which consist of two attributes (*Artist* and *SongTitle*). Each item in the table must have these two attributes. The combination of *Artist* and *SongTitle* distinguishes each item in the table from all of the others.

With DynamoDB other than the primary key, the *Music* table is schemaless, which means that neither the attributes nor their data types need to be defined beforehand. Each item can have its own distinct attributes.



Lets look at how Amazon DynamoDB works.



To understanding how Amazon DynamoDB distributes data lets examine how data is stored, written to a table, and retrieved from a table.

How DynamoDB stores data:

Amazon DynamoDB stores data in partitions. A **partition** is an allocation of storage for a table, backed by solid state drives (SSDs) and automatically replicated across multiple Availability Zones within an AWS Region.

How DynamoDB stores and retrieves items:

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

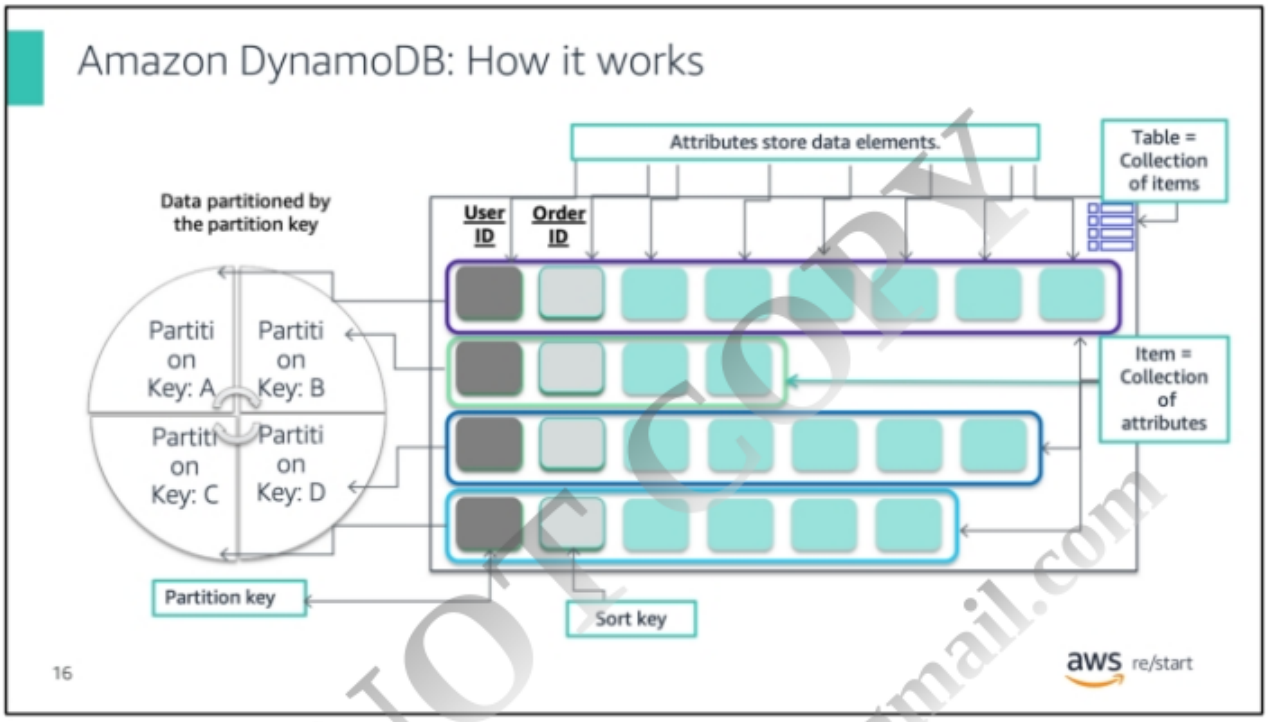
How DynamoDB write an item to the table:

DynamoDB uses the value of the partition key as input to an internal hash function. The output value from the hash function determines the partition in

which the item will be stored.

How DynamoDB reads an item from a table:

To read an item from the table, you must specify the partition key value for the item. DynamoDB uses this value as input to its hash function, yielding the partition in which the item can be found.



In DynamoDB, data is stored in *tables*. A table contains *items* with *attributes*. You can think of items as rows or tuples in a relational database, and attributes as columns.

DynamoDB stores data in partitions, and divides table items into multiple partitions based on the *partition key* value. A *partition* is an allocation of storage for a table. It is backed by SSDs and is automatically replicated across multiple Availability Zones in an AWS Region. DynamoDB handles partition management. The partition key of an item is also known as its *hash attribute*.

A *sort key* can be defined to store all the items with the same partition key value physically close together. It can order them by sort key value in the partition. It represents a one-to-many relationship based on the partition key, and enables querying on the sort key attribute.

A table has a primary key that uniquely identifies each item in the table. The two types of primary keys are:

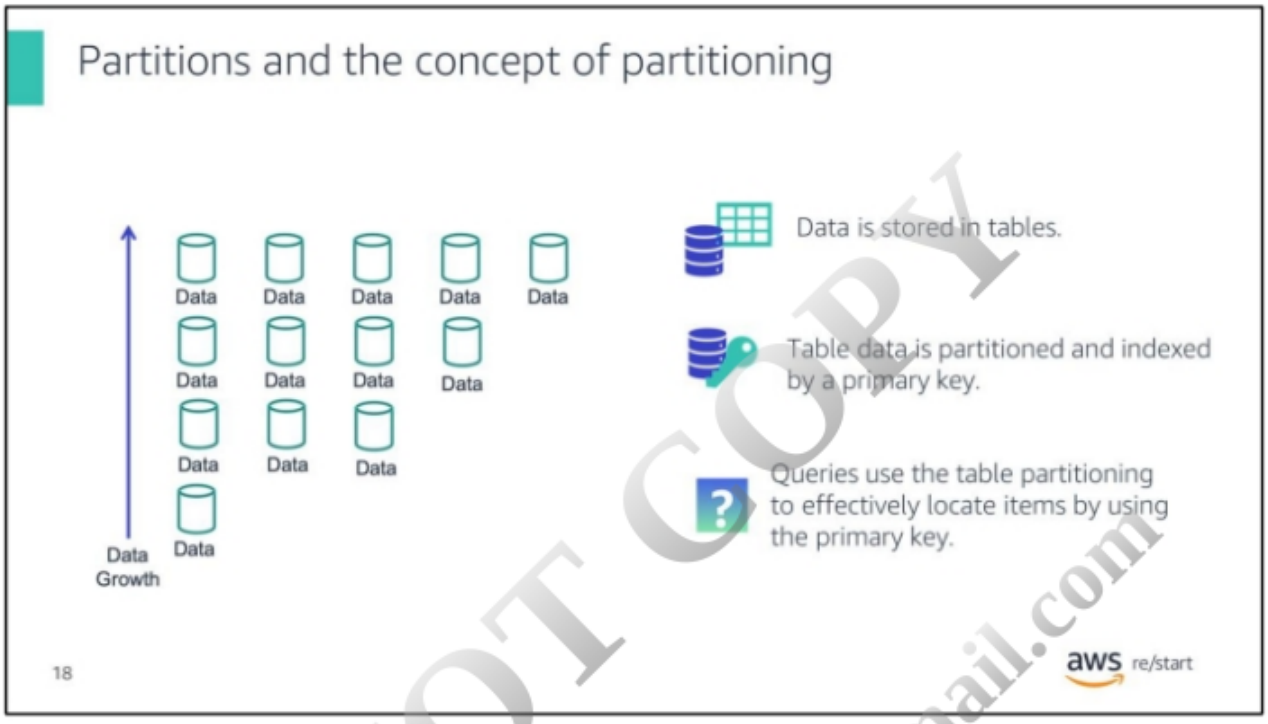
- *Partition primary key* – The primary key consists of a single attribute, which is the partition key. DynamoDB builds an unordered index on this primary key attribute. Each item in the table is uniquely identified by its partition

key value.

- *Partition and sort primary key* – The primary key is made of two attributes. The first attribute is the partition key attribute and the second attribute is the sort key attribute. DynamoDB builds an unordered index on the partition key attribute and a sorted index on the sort key attribute. Each item in the table is uniquely identified by the combination of its partition key and sort key values.

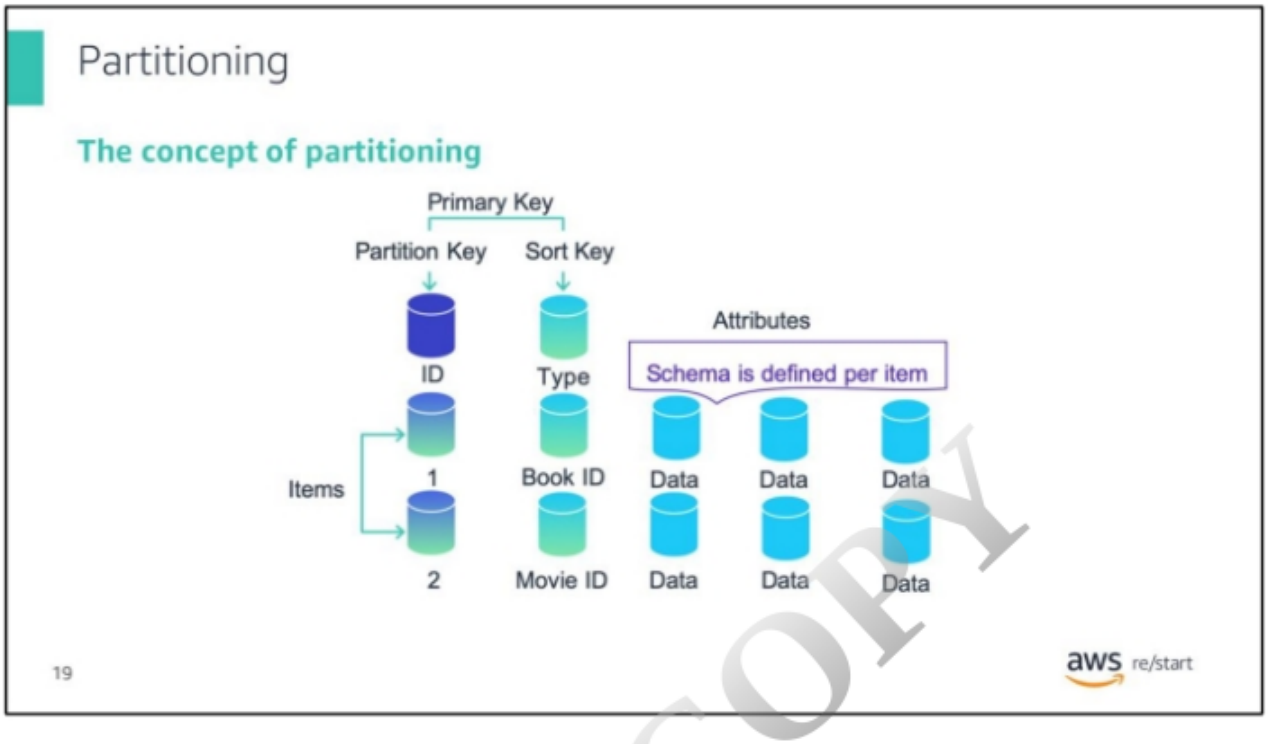
In the example, the table has a partition-and-sort primary key that consists of the attributes *User ID* and *Order ID*. For information about the DynamoDB data model and tables, refer to: What Is Amazon DynamoDB?

Lets look at how partitioning works.



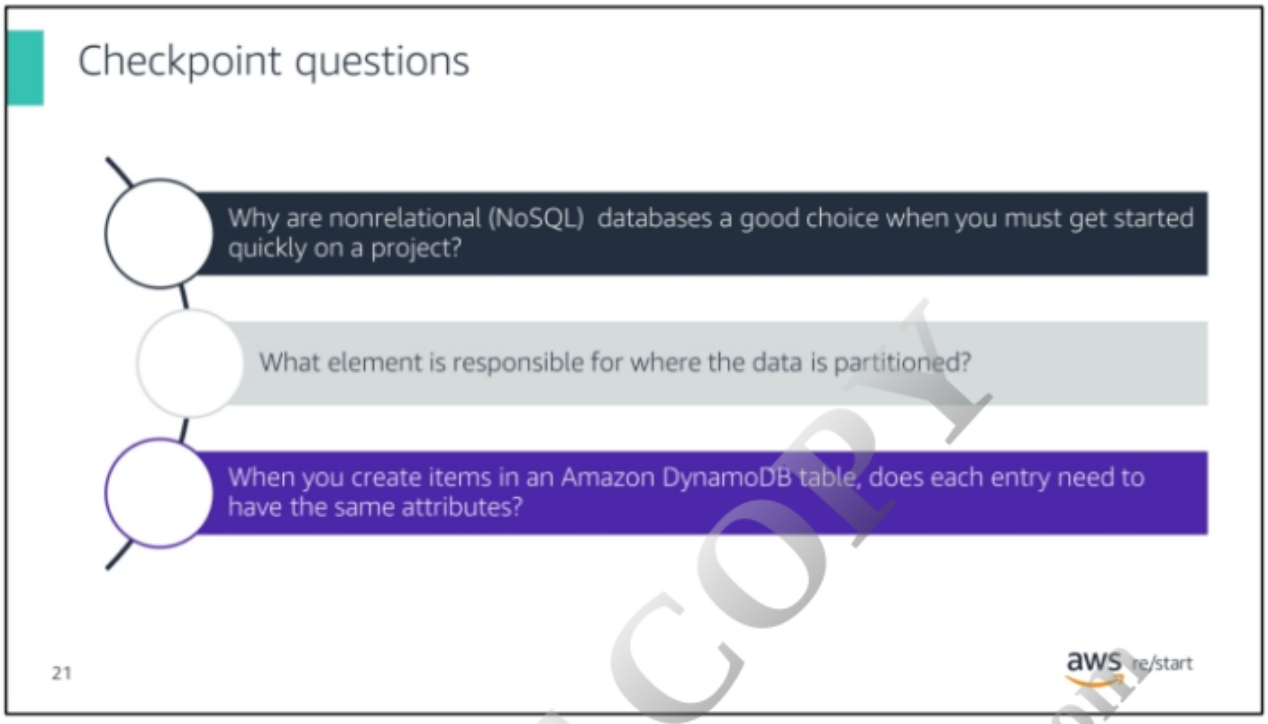
As data grows, table data is partitioned and indexed by a primary key.
Two different ways are available for retrieving data from a DynamoDB table.

- In the first method, the *query operation* uses table partitioning to locate items effectively by using the primary key.
- The second method uses a *scan*, which enables you to locate items in the table by matching conditions on non-key attributes.
  - This second method gives you the flexibility to locate items by other attributes.
  - However, the operation is less efficient because DynamoDB scans through all the items in the table to find the ones that match your criteria.



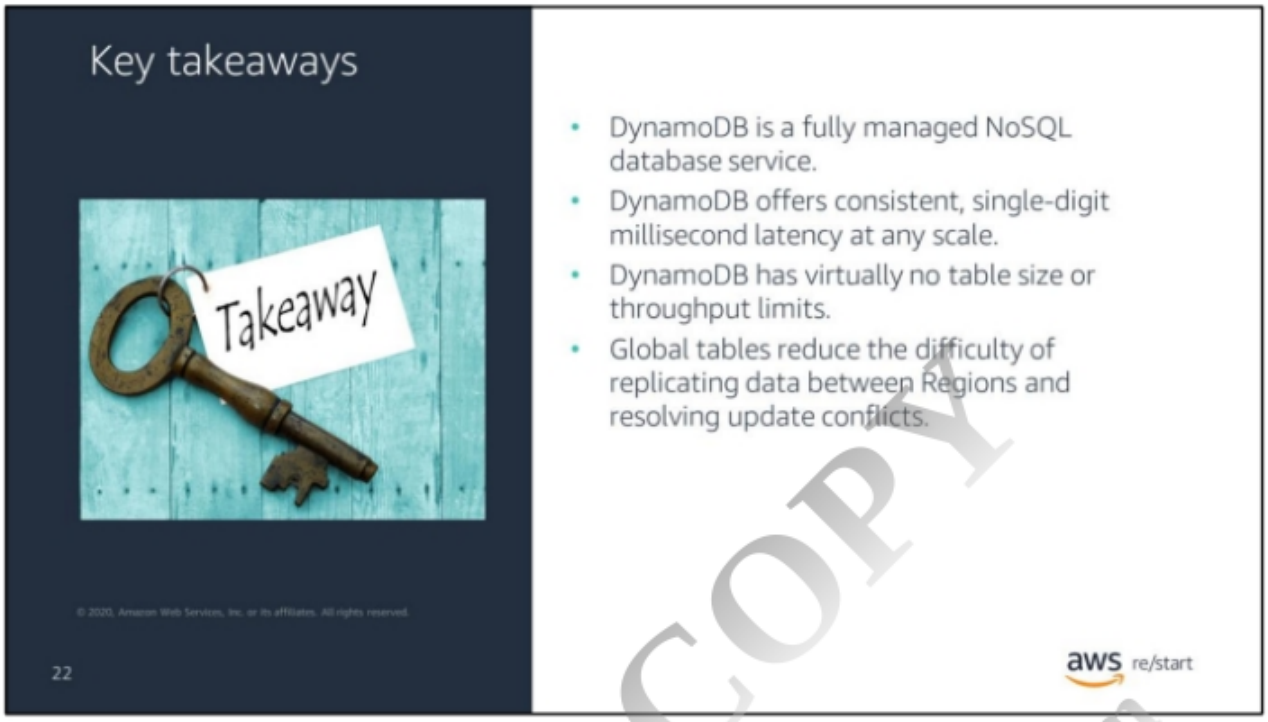
A *partition* is an allocation of storage for a table, which is backed with solid state drives (SSDs) and automatically replicated across multiple Availability Zones within an AWS Region.
For more information, refer to the following resource: Partitions and Data Distribution.

Review the DynamoDB demonstration: Amazon DynamoDB Console Demo. This video demonstration can be found in the learning management system.



Answers:

1. 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. The primary key is responsible for where the data is partitioned.
3. No. Because DynamoDB is a NoSQL database, individual items can have different attributes than other entries.



Some key takeaways from this lesson include:

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