DynamoDB Concepts

This section briefly introduces some basic DynamoDB concepts. This helps you as you follow the steps in the tutorials.

Tables

Similar to other database management systems, DynamoDB stores data in tables. A *table* is a collection of data. For example, you could create a table named `People`, where you could store information about friends, family, or anyone else of interest. You could also have a `Cars` table to store information about vehicles that people drive. Table names, like everything in DynamoDB, are case-sensitive and must be between 3 and 255 characters long and made up of a-z, A-Z, 0-9, underscore, dot or dash characters.

Items

Each table contains multiple items. An *item* is a group of attributes that is uniquely identifiable among all of the other items. In a `People` table, each item represents one person. For a `Cars` table, each item represents one vehicle. In many ways, items are similar to rows, records, or tuples in relational database systems. In DynamoDB, there is no limit to the number of items you can store in a table.

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, a `Department` item might have attributes such as `DepartmentID`, `Name`, `Manager`, and so on. An item in a `People` table could contain attributes such as `PersonID`, `LastName`, `FirstName`, and so on. Attributes in DynamoDB are similar in many ways to fields or columns in other database management systems. Attribute names, like everything in DynamoDB, are case-sensitive and must be between 1 and 255 characters long and made up of a-z, A-Z, 0-9, underscore, dot or dash characters.

Attributes can be of a certain type, such as scalar (number, string, boolean, binary or Null), document (like a complete json document), or a set (which is a grouping of scalar values).

Primary Key

When you create a table, in addition to the table name, you must specify the primary key of the table. As in other databases, a primary key in DynamoDB uniquely identifies each item in the table, so that no two items can have the same key. When you add, update, or delete an item in the table, you must specify the primary key attribute values for that item. The key values are required; you cannot omit them.

DynamoDB supports two different kinds of primary keys:

- Partition key—A simple primary key, composed of one attribute known as the *partition key*. DynamoDB uses the partition key's value as input to an internal hash function; the output from the hash function determines the partition where the item is stored. With a simple primary key, no two items in a table can have the same partition key value.
- Partition key and sort key—A composite primary key, composed of two attributes. The first attribute is the *partition key*, and the second attribute is the *sort key*. DynamoDB uses the partition key value as input to an internal hash function; the output from the hash function determines the partition where the item is stored. All items with the same partition key are stored together, in sorted order by sort key value. With a composite primary key, it is possible for two items to have the same partition key value, but those two items must have different sort key values.

Note

The partition key of an item is also known as its *hash attribute*. The term derives from the service's use of an internal hash function to evenly distribute data items across partitions, based on their partition key values.

The sort key of an item is also known as its *range attribute*. The term derives from the way DynamoDB stores items with the same partition key physically close together, in sorted order by the sort key value.

Secondary Indexes

In DynamoDB, you can read data in a table by providing primary key attribute values. If you want to read the data using non-key attributes, you can use a secondary index to do this. After you create a secondary index on a table, you can read data from the index in much the same way as you do from the table. By using secondary indexes, your applications can use many different query patterns, in addition to accessing the data by primary key values.

Additional Resources:

Amazon DynamoDB Core Components (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html)