Basic operations on DynamoDB tables

PDF (dynamodb-dg.pdf#WorkingWithTables.Basics) | RSS (dynamodbupdates.rss)

Similar to other database systems, Amazon DynamoDB stores data in tables. You can manage your tables using a few basic operations.

Topics

Creating a table

Use the `CreateTable` operation to create a table in Amazon DynamoDB. To create the table, you must provide the following information:

- **Table name.** The name must conform to the DynamoDB naming rules, and must be unique for the current AWS account and Region. For example, you could create a `People` table in US East (N. Virginia) and another `People` table in Europe (Ireland). However, these two tables would be entirely different from each other. For more information, see Supported data types and naming rules in Amazon DynamoDB (./HowItWorks.NamingRulesDataTypes.html) .
- **Primary key.** The primary key can consist of one attribute (partition key) or two attributes (partition key and sort key). You need to provide the attribute names, data types, and the role of each attribute: `HASH` (for a partition key) and `RANGE` (for a sort key). For more information, see Primary key (./HowItWorks.CoreComponents.html#HowItWorks.CoreComponents.PrimaryKey) .
- **Throughput settings (for provisioned tables).** If using provisioned mode, you must specify the initial read and write throughput settings for the table. You can modify these settings later, or enable DynamoDB auto scaling to manage the settings for you. For more information, see Managing settings on DynamoDB provisioned capacity tables

(./ProvisionedThroughput.html) and Managing throughput capacity automatically with DynamoDB auto scaling (./AutoScaling.html) .

Example 1: Create a provisioned table

The following AWS CLI example shows how to create a table (`Music`). The primary key consists of `Artist` (partition key) and `SongTitle` (sort key), each of which has a data type of `String` . The maximum throughput for this table is 10 read capacity units and 5 write capacity units.

```
aws dynamodb create-table \
    --table-name Music \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput \
        ReadCapacityUnits=10,WriteCapacityUnits=5
```

The `CreateTable` operation returns metadata for the table, as shown following.

```
{
    "TableDescription": {
        "TableArn": "arn:aws:dynamodb:us-east-
1:123456789012:table/Music",
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 5,
```

```
            "ReadCapacityUnits": 10
        },
        "TableSizeBytes": 0,
        "TableName": "Music",
        "TableStatus": "CREATING",
        "TableId": "12345678-0123-4567-a123-abcdefghijkl",
        "KeySchema": [
            {
                "KeyType": "HASH",
                "AttributeName": "Artist"
            },
            {
                "KeyType": "RANGE",
                "AttributeName": "SongTitle"
            }
        ],
        "ItemCount": 0,
        "CreationDateTime": 1542397215.37
    }
}
```

The `TableStatus` element indicates the current state of the table (`CREATING`). It might take a while to create the table, depending on the values you specify for `ReadCapacityUnits` and `WriteCapacityUnits`. Larger values for these require DynamoDB to allocate more resources for the table.

Example 2: Create an on-demand table

To create the same table `Music` using on-demand mode.

```
aws dynamodb create-table \
    --table-name Music \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode=PAY_PER_REQUEST
```

The `CreateTable` operation returns metadata for the table, as shown following.


```
{
    "TableDescription": {
        "TableArn": "arn:aws:dynamodb:us-east-
1:123456789012:table/Music",
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 0,
            "ReadCapacityUnits": 0
        },
        "TableSizeBytes": 0,
        "TableName": "Music",
        "BillingModeSummary": {
            "BillingMode": "PAY_PER_REQUEST"
        },
        "TableStatus": "CREATING",
        "TableId": "12345678-0123-4567-a123-abcdefghijkl",
        "KeySchema": [
            {
                "KeyType": "HASH",
                "AttributeName": "Artist"
            },
            {
                "KeyType": "RANGE",
                "AttributeName": "SongTitle"
            }
        ],
        "ItemCount": 0,
        "CreationDateTime": 1542397468.348
```

```
    }
}
```

> **Important**
>
> When calling `DescribeTable` on an on-demand table, read capacity units and write capacity units are set to 0.

Example 3: Create a table using the DynamoDB standard-infrequent access table class

To create the same `Music` table using the DynamoDB Standard-Infrequent Access table class.

```
aws dynamodb create-table \
    --table-name Music \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput \
        ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --table-class STANDARD_INFREQUENT_ACCESS
```

The `CreateTable` operation returns metadata for the table, as shown following.

```
{
    "TableDescription": {
        "TableArn": "arn:aws:dynamodb:us-east-
1:123456789012:table/Music",
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
```

```
                "AttributeType": "S"
            }
        ],
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 5,
            "ReadCapacityUnits": 10
        },
        "TableClassSummary": {
            "LastUpdateDateTime": 1542397215.37,
            "TableClass": "STANDARD_INFREQUENT_ACCESS"
        },
        "TableSizeBytes": 0,
        "TableName": "Music",
        "TableStatus": "CREATING",
        "TableId": "12345678-0123-4567-a123-abcdefghijkl",
        "KeySchema": [
            {
                "KeyType": "HASH",
                "AttributeName": "Artist"
            },
            {
                "KeyType": "RANGE",
                "AttributeName": "SongTitle"
            }
        ],
        "ItemCount": 0,
        "CreationDateTime": 1542397215.37
    }
}
```

Describing a table

To view details about a table, use the `DescribeTable` operation. You must provide the table name. The output from `DescribeTable` is in the same format as that from `CreateTable`. It includes the timestamp when the table was created, its key schema, its provisioned throughput settings, its estimated size, and any secondary indexes that are present.

> **Important**
>
> When calling `DescribeTable` on an on-demand table, read capacity units and write capacity units are set to 0.

Example

```
aws dynamodb describe-table --table-name Music
```

The table is ready for use when the `TableStatus` has changed from `CREATING` to `ACTIVE`.

> **Note**
>
> If you issue a `DescribeTable` request immediately after a `CreateTable` request, DynamoDB might return an error (`ResourceNotFoundException`). This is because `DescribeTable` uses an eventually consistent query, and the metadata for your table might not be available at that moment. Wait for a few seconds, and then try the `DescribeTable` request again.
>
> For billing purposes, your DynamoDB storage costs include a per-item overhead of 100 bytes. (For more information, go to DynamoDB Pricing (https://aws.amazon.com/dynamodb/pricing/) .) This extra 100 bytes per item is not used in capacity unit calculations or by the `DescribeTable` operation.

Updating a table

The `UpdateTable` operation allows you to do one of the following:

- Modify a table's provisioned throughput settings (for provisioned mode tables).
- Change the table's read/write capacity mode.
- Manipulate global secondary indexes on the table (see Using Global Secondary Indexes in DynamoDB (./GSI.html)).
- Enable or disable DynamoDB Streams on the table (see Change data capture for DynamoDB Streams (./Streams.html)).

Example

The following AWS CLI example shows how to modify a table's provisioned throughput settings.

```
aws dynamodb update-table --table-name Music \
    --provisioned-throughput ReadCapacityUnits=20,WriteCapacityUnits=10
```

> **Note**
>
> When you issue an `UpdateTable` request, the status of the table changes from `AVAILABLE` to `UPDATING`. The table remains fully available for use while it is `UPDATING`. When this process is completed, the table status changes from `UPDATING` to `AVAILABLE`.

Example

The following AWS CLI example shows how to modify a table's read/write capacity mode to on-demand mode.

```
aws dynamodb update-table --table-name Music \
    --billing-mode PAY_PER_REQUEST
```

Deleting a table

You can remove an unused table with the `DeleteTable` operation. Deleting a table is an unrecoverable operation.

Example

The following AWS CLI example shows how to delete a table.

```
aws dynamodb delete-table --table-name Music
```

When you issue a `DeleteTable` request, the table's status changes from `ACTIVE` to `DELETING`. It might take a while to delete the table, depending on the resources it uses (such as the data stored in the table, and any streams or indexes on the table).

When the `DeleteTable` operation concludes, the table no longer exists in DynamoDB.

Listing table names

The `ListTables` operation returns the names of the DynamoDB tables for the current AWS account and Region.

Example

The following AWS CLI example shows how to list the DynamoDB table names.

```
aws dynamodb list-tables
```

Describing provisioned throughput quotas

The `DescribeLimits` operation returns the current read and write capacity quotas for the current AWS account and Region.

Example

The following AWS CLI example shows how to describe the current provisioned throughput quotas.

```
aws dynamodb describe-limits
```

The output shows the upper quotas of read and write capacity units for the current AWS account and Region.

For more information about these quotas, and how to request quota increases, see Throughput default quotas (./ServiceQuotas.html#default-limits-throughput) .