



DynamoDB

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

About the Tutorial

DynamoDB is a fully-managed NoSQL database service designed to deliver fast and predictable performance. It uses the Dynamo model in the essence of its design, and improves those features. It began as a way to manage website scalability challenges presented by the holiday season load.

This tutorial introduces you to key DynamoDB concepts necessary for creating and deploying a highly-scalable and performance-focused database.

Audience

This tutorial targets IT professionals, students, and management professionals who want a solid grasp of essential DynamoDB concepts.

After completing this tutorial, you will achieve intermediate expertise in DynamoDB, and easily build on your knowledge to solve more challenging problems.

Prerequisites

This tutorial assumes general knowledge of database technology, programming, Java or Java-like programming languages, and querying languages. It also assumes familiarity with typical database operations in an application.

Copyright and Disclaimer

© Copyright 2016 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, do notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial.....	i
Audience	i
Prerequisites	i
Copyright and Disclaimer	i
Table of Contents	ii
 1. DYNAMODB – OVERVIEW.....	 1
DynamoDB vs. RDBMS	1
 2. DYNAMODB – BASIC CONCEPTS	 3
Primary Key.....	3
Secondary Indexes	3
Provisioned Throughput.....	4
Read Consistency	4
Partitions	5
 3. DYNAMODB – ENVIRONMENT.....	 6
Local Install	6
Working Environment	7
 4. DYNAMODB – OPERATIONS TOOLS	 8
GUI Console	8
The JavaScript Shell.....	9
 5. DYNAMODB – DATA TYPES.....	 10
Attribute Data Types	10
 6. DYNAMODB – CREATE TABLE	 12
Create Table using.....	12
the GUI Console	12

Create Table using	13
Java	13
7. DYNAMODB – LOAD TABLE	16
Load Table using.....	16
GUI Console	16
Load Table using.....	16
Java	16
8. DYNAMODB – QUERY TABLE	20
Query Table using	20
the GUI Console	20
Query Table using	21
Java	21
9. DYNAMODB – DELETE TABLE.....	25
Delete Table using.....	25
the GUI Console	25
Delete Table using.....	26
Java	26
10. DYNAMODB – API INTERFACE	27
Manipulate Tables	27
Read Data	27
Modify Data	28
11. DYNAMODB – CREATING ITEMS	29
How to Create an Item Using the GUI Console?	29
How to Use Java in Item Creation?	32

12. DYNAMODB – GETTING ITEMS	36
Retrieve an Item.....	36
Item Retrieval Using Java	36
13. DYNAMODB – UPDATE ITEMS	40
How to Update Items Using GUI Tools?.....	40
Update Items Using Java	41
Update Items Using Counters.....	42
14. DYNAMODB – DELETE ITEMS.....	46
Delete Items Using the GUI Console	46
How to Delete Items Using Java?	47
15. DYNAMODB – BATCH WRITING	52
What is Batch Writing?.....	52
Batch Writes with Java	52
16. DYNAMODB – BATCH RETRIEVE	56
Batch Retrievals with Java	56
17. DYNAMODB – QUERYING	60
Querying with Java.....	61
18. DYNAMODB – SCAN	64
Types of Scan Operations	64
Parallel Scan.....	65
19. DYNAMODB – INDEXES.....	67

20. DYNAMODB – GLOBAL SECONDARY INDEXES	70
Attribute Projections	70
Global Secondary Index Queries and Scans	71
Global Secondary Index Storage	72
Using Java to Work with Global Secondary Indexes	73
21. DYNAMODB – LOCAL SECONDARY INDEXES	82
Projecting an Attribute	82
Local Secondary Index Creation	83
Using Java to Work with Local Secondary Indexes	84
22. DYNAMODB – AGGREGATION	95
23. DYNAMODB – ACCESS CONTROL	96
Types of Permissions	96
Granting Privileges: Using The Shell	99
24. DYNAMODB – PERMISSIONS API	102
Permissions and API Actions	102
25. DYNAMODB – CONDITIONS	105
Detailed Control	105
26. DYNAMODB – WEB IDENTITY FEDERATION	107
27. DYNAMODB – DATA PIPELINE	108
Using Data Pipeline	108
28. DYNAMODB – DATA BACKUP	109
Exporting and Importing Data	109
Importing Data	109
Errors	110

29. DYNAMODB – MONITORING	111
Cloudwatch Console	111
API Integration	111
30. DYNAMODB – CLOUDTRAIL	113
31. DYNAMODB – MAPREDUCE	116
Hive Setup	116
Activate SSH Session	117
32. DYNAMODB – TABLE ACTIVITY	119
Managing Streams	119
33. DYNAMODB – ERROR HANDLING	126
Codes and Messages	126
34. DYNAMODB – BEST PRACTICES	129
Tables	129
Items	129
Queries and Scans	129
Local Secondary Indices	130
Global Secondary Indices	130
35. DYNAMODB – USEFUL RESOURCES	131

1. DynamoDB – Overview

DynamoDB allows users to create databases capable of storing and retrieving any amount of data, and serving any amount of traffic. It automatically distributes data and traffic over servers to dynamically manage each customer's requests, and also maintains fast performance.

DynamoDB vs. RDBMS

DynamoDB uses a NoSQL model, which means it uses a non-relational system. The following table highlights the differences between DynamoDB and RDBMS:

Common Tasks	RDBMS	DynamoDB
Connect to the Source	It uses a persistent connection and SQL commands.	It uses HTTP requests and API operations.
Create a Table	Its fundamental structures are tables, and must be defined.	It only uses primary keys, and no schema on creation. It uses various data sources.
Get Table Info	All table info remains accessible.	Only primary keys are revealed.
Load Table Data	It uses rows made of columns.	In tables, it uses items made of attributes.
Read Table Data	It uses SELECT statements and filtering statements.	It uses GetItem, Query, and Scan.
Manage Indexes	It uses standard indexes created through SQL statements. Modifications to it occur automatically on table changes.	It uses a secondary index to achieve the same function. It requires specifications (partition key and sort key).
Modify Table Data	It uses an UPDATE statement.	It uses an UpdateItem operation.
Delete Table Data	It uses a DELETE statement.	It uses a DeleteItem operation.
Delete a Table	It uses a DROP TABLE statement.	It uses a DeleteTable operation.

Advantages

The two main advantages of DynamoDB are scalability and flexibility. It does not force the use of a particular data source and structure, allowing users to work with virtually anything, but in a uniform way.

Its design also supports a wide range of use from lighter tasks and operations to demanding enterprise functionality. It also allows simple use of multiple languages: Ruby, Java, Python, C#, Erlang, PHP, and Perl.

Limitations

DynamoDB does suffer from certain limitations, however, these limitations do not necessarily create huge problems or hinder solid development.

You can review them from the following points:

- **Capacity Unit Sizes** – A read capacity unit is a single consistent read per second for items no larger than 4KB. A write capacity unit is a single write per second for items no bigger than 1KB.
- **Provisioned Throughput Min/Max** – All tables and global secondary indices have a minimum of one read and one write capacity unit. Maximums depend on region. In the US, 40K read and write remains the cap per table (80K per account), and other regions have a cap of 10K per table with a 20K account cap.
- **Provisioned Throughput Increase and Decrease** – You can increase this as often as needed, but decreases remain limited to no more than four times daily per table.
- **Table Size and Quantity Per Account** – Table sizes have no limits, but accounts have a 256 table limit unless you request a higher cap.
- **Secondary Indexes Per Table** – Five local and five global are permitted.
- **Projected Secondary Index Attributes Per Table** – DynamoDB allows 20 attributes.
- **Partition Key Length and Values** – Their minimum length sits at 1 byte, and maximum at 2048 bytes, however, DynamoDB places no limit on values.
- **Sort Key Length and Values** – Its minimum length stands at 1 byte, and maximum at 1024 bytes, with no limit for values unless its table uses a local secondary index.
- **Table and Secondary Index Names** – Names must conform to a minimum of 3 characters in length, and a maximum of 255. They use the following characters: A-Z, a-z, 0-9, "_", "-", and ".".
- **Attribute Names** – One character remains the minimum, and 64KB the maximum, with exceptions for keys and certain attributes.
- **Reserved Words** – DynamoDB does not prevent the use of reserved words as names.
- **Expression Length** – Expression strings have a 4KB limit. Attribute expressions have a 255-byte limit. Substitution variables of an expression have a 2MB limit.

2. DynamoDB – Basic Concepts

Before using DynamoDB, you must familiarize yourself with its basic components and ecosystem. In the DynamoDB ecosystem, you work with tables, attributes, and items. A table holds sets of items, and items hold sets of attributes. An attribute is a fundamental element of data requiring no further decomposition, i.e., a field.

Primary Key

The Primary Keys serve as the means of unique identification for table items, and secondary indexes provide query flexibility. DynamoDB streams record events by modifying the table data.

The Table Creation requires not only setting a name, but also the primary key; which identifies table items. No two items share a key. DynamoDB uses two types of primary keys:

- **Partition Key** – This simple primary key consists of a single attribute referred to as the “partition key.” Internally, DynamoDB uses the key value as input for a hash function to determine storage.
- **Partition Key and Sort Key** – This key, known as the “Composite Primary Key”, consists of two attributes:
 - The partition key and
 - The sort key.

DynamoDB applies the first attribute to a hash function, and stores items with the same partition key together; with their order determined by the sort key. Items can share partition keys, but not sort keys.

The Primary Key attributes only allow scalar (single) values; and string, number, or binary data types. The non-key attributes do not have these constraints.

Secondary Indexes

These indexes allow you to query table data with an alternate key. Though DynamoDB does not force their use, they optimize querying.

DynamoDB uses two types of secondary indexes:

- **Global Secondary Index** – This index possesses partition and sort keys, which can differ from table keys.
- **Local Secondary Index** – This index possesses a partition key identical to the table, however, its sort key differs.

API

The API operations offered by DynamoDB include those of the control plane, data plane (e.g., creation, reading, updating, and deleting), and streams. In control plane operations, you create and manage tables with the following tools:

- CreateTable
- DescribeTable
- ListTables
- UpdateTable
- DeleteTable

In the data plane, you perform CRUD operations with the following tools:

Create	Read	Update	Delete
PutItem BatchWriteItem	GetItem BatchGetItem Query Scan	UpdateItem	DeleteItem BatchWriteItem

The stream operations control table streams. You can review the following stream tools:

- ListStreams
- DescribeStream
- GetShardIterator
- GetRecords

Provisioned Throughput

In table creation, you specify provisioned throughput, which reserves resources for reads and writes. You use capacity units to measure and set throughput.

When applications exceed the set throughput, requests fail. The DynamoDB GUI console allows monitoring of set and used throughput for better and dynamic provisioning.

Read Consistency

DynamoDB uses **eventually consistent** and **strongly consistent** reads to support dynamic application needs. Eventually consistent reads do not always deliver current data.

The strongly consistent reads always deliver current data (with the exception of equipment failure or network problems). Eventually consistent reads serve as the default setting, requiring a setting of true in the **ConsistentRead** parameter to change it.

Partitions

DynamoDB uses partitions for data storage. These storage allocations for tables have SSD backing and automatically replicate across zones. DynamoDB manages all the partition tasks, requiring no user involvement.

In table creation, the table enters the CREATING state, which allocates partitions. When it reaches ACTIVE state, you can perform operations. The system alters partitions when its capacity reaches maximum or when you change throughput.

3. DynamoDB – Environment

The DynamoDB Environment only consists of using your Amazon Web Services account to access the DynamoDB GUI console, however, you can also perform a local install.

Navigate to the following website: <https://aws.amazon.com/dynamodb/>

Click the “Get Started with Amazon DynamoDB” button, or the “Create an AWS Account” button if you do not have an Amazon Web Services account. The simple, guided process will inform you of all the related fees and requirements.

After performing all the necessary steps of the process, you will have the access. Simply sign in to the AWS console, and then navigate to the DynamoDB console.

Be sure to delete unused or unnecessary material to avoid associated fees.

Local Install

The AWS (Amazon Web Service) provides a version of DynamoDB for local installations. It supports creating applications without the web service or a connection. It also reduces provisioned throughput, data storage, and transfer fees by allowing a local database. This guide assumes a local install.

When ready for deployment, you can make a few small adjustments to your application to convert it to AWS use.

The install file is a **.jar executable**. It runs in Linux, Unix, Windows, and any other OS with Java support. Download the file by using one of the following links:

- **Tarball** – http://dynamodb-local.s3-website-us-west-2.amazonaws.com/dynamodb_local_latest.tar.gz
- **Zip archive** – http://dynamodb-local.s3-website-us-west-2.amazonaws.com/dynamodb_local_latest.zip

Note: Other repositories offer the file, but not necessarily the latest version. Use the links above for up-to-date install files. Also, ensure you have Java Runtime Engine (JRE) version 6.x or a newer version. DynamoDB cannot run with older versions.

After downloading the appropriate archive, extract its directory (DynamoDBLocal.jar) and place it in the desired location.

You can then start DynamoDB by opening a command prompt, navigating to the directory containing DynamoDBLocal.jar, and entering the following command:

```
java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb
```

You can also stop the DynamoDB by closing the command prompt used to start it.

Working Environment

You can use a JavaScript shell, a GUI console, and multiple languages to work with DynamoDB. The languages available include Ruby, Java, Python, C#, Erlang, PHP, and Perl.

In this tutorial, we use Java and GUI console examples for conceptual and code clarity. Install a Java IDE, the AWS SDK for Java, and setup AWS security credentials for the Java SDK in order to utilize Java.

Conversion from Local to Web Service Code

When ready for deployment, you will need to alter your code. The adjustments depend on code language and other factors. The main change merely consists of changing the **endpoint** from a local point to an AWS region. Other changes require deeper analysis of your application.

A local install differs from the web service in many ways including, but not limited to the following key differences:

- The local install creates tables immediately, but the service takes much longer.
- The local install ignores throughput.
- The deletion occurs immediately in a local install.
- The reads/writes occur quickly in local installs due to the absence of network overhead.

4. DynamoDB – Operations Tools

DynamoDB provides three options for performing operations: a web-based GUI console, a JavaScript shell, and a programming language of your choice.

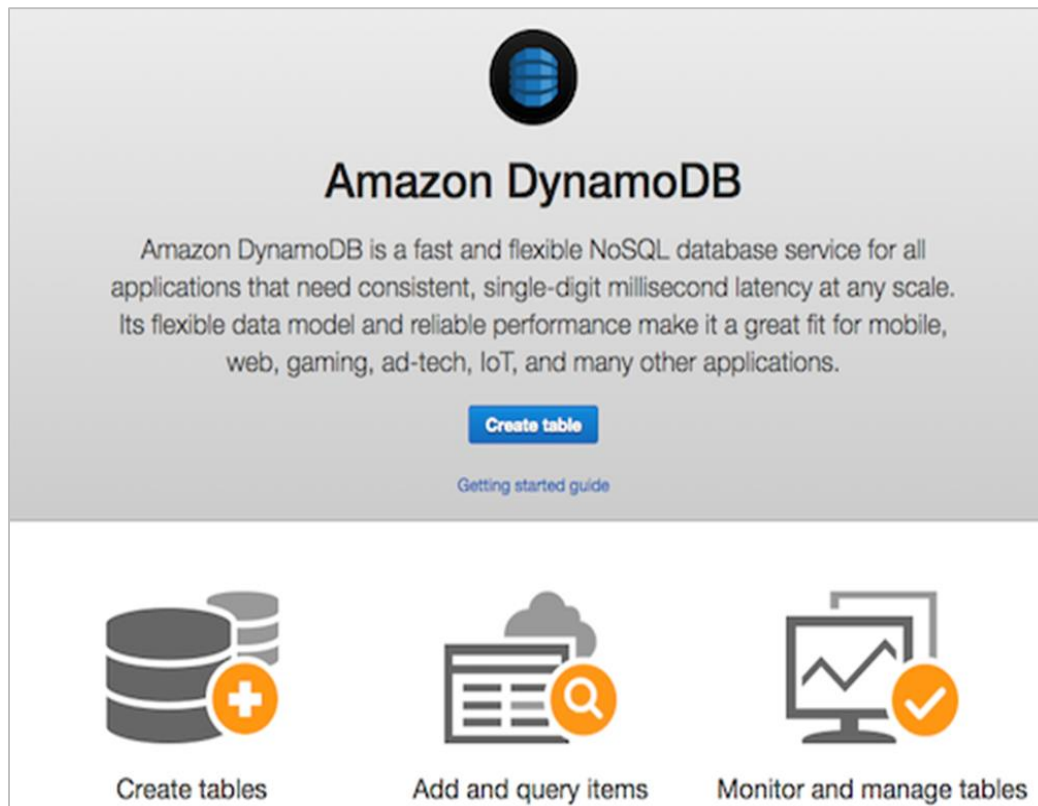
In this tutorial, we will focus on using the GUI console and Java language for clarity and conceptual understanding.

GUI Console

The GUI console or the AWS Management Console for Amazon DynamoDB can be found at the following address: <https://console.aws.amazon.com/dynamodb/home>

It allows you to perform the following tasks:

- CRUD
- View Table Items
- Perform Table Queries
- Set Alarms for Table Capacity Monitoring
- View Table Metrics in Real-Time
- View Table Alarms

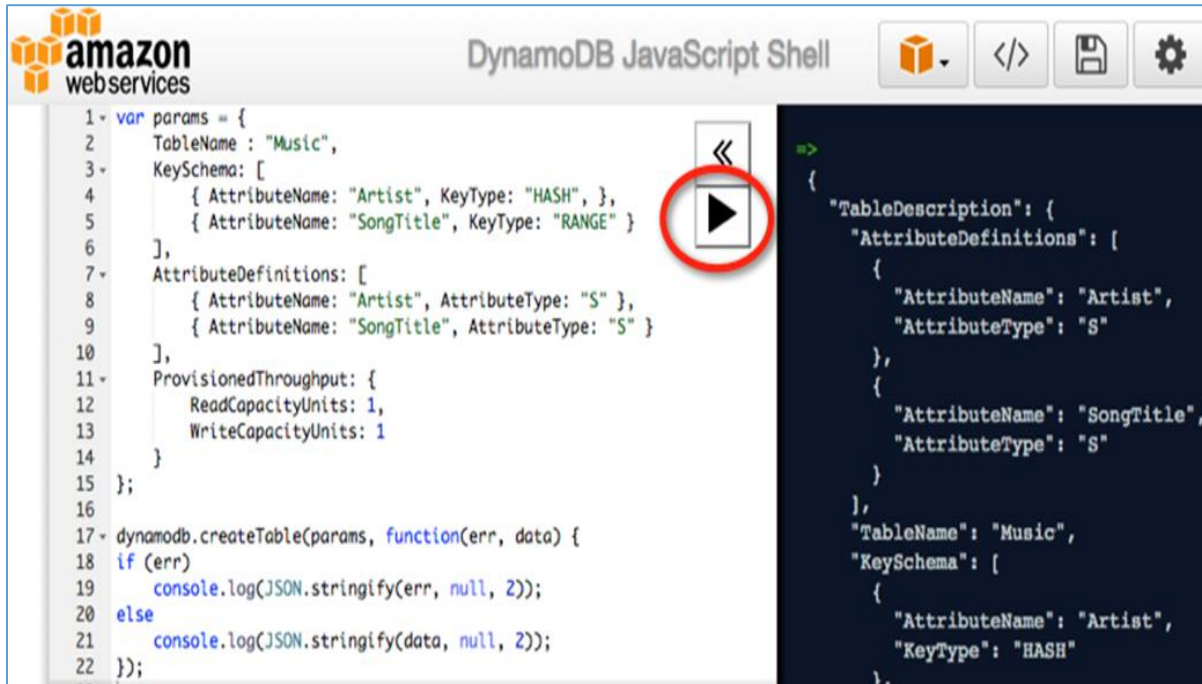


If your DynamoDB account has no tables, on access, it guides you through creating a table. Its main screen offers three shortcuts for performing common operations:

- Create Tables
- Add and Query Tables
- Monitor and Manage Tables

The JavaScript Shell

DynamoDB includes an interactive JavaScript shell. The shell runs inside a web browser, and the recommended browsers include Firefox and Chrome.



The screenshot shows the Amazon DynamoDB JavaScript Shell interface. The left pane contains the following JavaScript code:

```

1 var params = {
2   TableName : "Music",
3   KeySchema: [
4     { AttributeName: "Artist", KeyType: "HASH", },
5     { AttributeName: "SongTitle", KeyType: "RANGE" }
6   ],
7   AttributeDefinitions: [
8     { AttributeName: "Artist", AttributeType: "S" },
9     { AttributeName: "SongTitle", AttributeType: "S" }
10  ],
11  ProvisionedThroughput: {
12    ReadCapacityUnits: 1,
13    WriteCapacityUnits: 1
14  }
15 };
16
17 dynamodb.createTable(params, function(err, data) {
18   if (err)
19     console.log(JSON.stringify(err, null, 2));
20   else
21     console.log(JSON.stringify(data, null, 2));
22 });

```

The right pane displays the resulting JSON output:

```

{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "Artist",
        "AttributeType": "S"
      },
      {
        "AttributeName": "SongTitle",
        "AttributeType": "S"
      }
    ],
    "TableName": "Music",
    "KeySchema": [
      {
        "AttributeName": "Artist",
        "KeyType": "HASH"
      },
      {
        "AttributeName": "SongTitle",
        "KeyType": "RANGE"
      }
    ]
  }
}

```

Note: Using other browsers may result in errors.

Access the shell by opening a web browser and entering the following address:
<http://localhost:8000/shell>

Use the shell by entering JavaScript in the left pane, and clicking the "Play" icon button in the top right corner of the left pane, which runs the code. The code results display in the right pane.

DynamoDB and Java

Use Java with DynamoDB by utilizing your Java development environment. Operations confirm to normal Java syntax and structure.

5. DynamoDB – Data Types

Data types supported by DynamoDB include those specific to attributes, actions, and your coding language of choice.

Attribute Data Types

DynamoDB supports a large set of data types for table attributes. Each data type falls into one of the three following categories:

- **Scalar** – These types represent a single value, and include number, string, binary, Boolean, and null.
- **Document** – These types represent a complex structure possessing nested attributes, and include lists and maps.
- **Set** – These types represent multiple scalars, and include string sets, number sets, and binary sets.

Remember DynamoDB as a schemaless, NoSQL database that does not need attribute or data type definitions when creating a table. It only requires a primary key attribute data types in contrast to RDBMS, which require column data types on table creation.

Scalars

- **Numbers** – They are limited to 38 digits, and are either positive, negative, or zero.
- **String** – They are Unicode using UTF-8, with a minimum length of >0 and maximum of 400KB.
- **Binary** – They store any binary data, e.g., encrypted data, images, and compressed text. DynamoDB views its bytes as unsigned.
- **Boolean** – They store true or false.
- **Null** – They represent an unknown or undefined state.

Document

- **List** – It stores ordered value collections, and uses square ([...]) brackets.
- **Map** – It stores unordered name-value pair collections, and uses curly ({...}) braces.

Set

Sets must contain elements of the same type whether number, string, or binary. The only limits placed on sets consist of the 400KB item size limit, and each element being unique.

Action Data Types

DynamoDB API holds various data types used by actions. You can review a selection of the following key types:

- **AttributeDefinition** – It represents key table and index schema.
- **Capacity** – It represents the quantity of throughput consumed by a table or index.
- **CreateGlobalSecondaryIndexAction** – It represents a new global secondary index added to a table.
- **LocalSecondaryIndex** – It represents local secondary index properties.
- **ProvisionedThroughput** – It represents the provisioned throughput for an index or table.
- **PutRequest** – It represents PutItem requests.
- **TableDescription** – It represents table properties.

Supported Java Datatypes

DynamoDB provides support for primitive data types, Set collections, and arbitrary types for Java.

End of ebook preview
If you liked what you saw...
Buy it from our store @ **<https://store.tutorialspoint.com>**