ytoY technologies®
Engineer yocto to Yotta

Amazon Dynamodb

Amazon DynamoDB

- Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale.
- It is a fully managed cloud database and supports both document and key-value store models.
- Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad tech, IoT, and many other applications.
- DynamoDB automatically spreads the data and traffic for the table over a sufficient number of servers to handle the request capacity specified by the customer and the amount of data stored, while maintaining consistent and fast performance

DynamoDB Data Model

- **Tables, Items, and Attributes**
  - In Amazon DynamoDB, a *table* is a collection of *items* and each item is a collection of *attributes*.

Example items

```
{
        Id = 101
        ProductName = "Book 101 Title"
        ISBN = "111-1111111111"
        Authors = [ "Author 1", "Author 2" ]
        Price = -2
        Dimensions = "8.5 x 11.0 x 0.5"
        PageCount = 500
        InPublication = 1
        ProductCategory = "Book"
}
```

DynamoDB Data Model

- **Tables, Items, and Attributes**
  - In Amazon DynamoDB, a *table* is a collection of *items* and each item is a collection of *attributes*.
  - DynamoDB only requires that a table has a primary key, but does not require you to define all of the attribute names and data types in advance.
  - Individual items in a DynamoDB table can have any number of attributes, although there is a limit of 400 KB on the item size.
  - An item size is the sum of lengths of its attribute names and values (binary and UTF-8 lengths).

DynamoDB Data Model

- **Primary Key**
  - When you create a table, in addition to the table name, you must specify the primary key of the table.
  - The primary key uniquely identifies each item in the table, so that no two items can have the same key.
- **Secondary Indexes**
  - When you create a table with a composite primary key (partition key and sort key), you can optionally define one or more secondary indexes on that table.
  - A secondary index lets you query the data in the table using an alternate key, in addition to queries against the primary key.

DynamoDB Data Model

- **DynamoDB Data Types**
- Amazon DynamoDB supports the following data types:
  - **Scalar types** – Number, String, Binary, Boolean, and Null.
  - **Document types** – List and Map.
  - **Set types** – String Set, Number Set, and Binary Set.

Supported Operations in DynamoDB

- Table Operations
  - DynamoDB provides operations to create, update and delete tables.
  - After the table is created, you can use the UpdateTable operation to increase or decrease a table's provisioned throughput.
  - DynamoDB also supports an operation to retrieve table information (the DescribeTable operation) including the current status of the table, the primary key, and when the table was created.
  - The ListTables operation enables you to get a list of tables in your account in the region of the endpoint you are using to communicate with DynamoDB

Supported Operations in DynamoDB

- Item Operations
  - Item operations enable you to add, update and delete items from a table.
  - The UpdateItem operation allows you to update existing attribute values, add new attributes, and delete existing attributes from an item.
  - You can also perform conditional updates. For example, if you are updating a price value, you can set a condition so the update happens only if the current price is $20.
  - DynamoDB provides an operation to retrieve a single item (GetItem) or multiple items (BatchGetItem). You can use the BatchGetItem operation to retrieve items from multiple tables

Supported Operations in DynamoDB

- Query and Scan
  - The Query operation enables you to query a table using the hash attribute and an optional range filter.
  - If the table has a secondary index, you can also Query the index using its key.You can query only tables whose primary key is of hash-and-range type; you can also query any secondary index on such tables.
  - Query is the most efficient way to retrieve items from a table or a secondary index.
  - DynamoDB also supports a Scan operation, which you can use on a table or a secondary index.
  - The Scan operation reads every item in the table or secondary index. For large tables and secondary indexes, a Scan can consume a large amount of resources; for this reason, we recommend that you design your applications so that you can use the Query operation mostly, and use Scan only where appropriate.

Supported Operations in DynamoDB

- Data Read and Consistency Considerations
  - DynamoDB maintains multiple copies of each item to ensure durability.
  - When you receive an "operation successful" response to your write request, DynamoDB ensures that the write is durable on multiple servers.
  - However, it takes time for the update to propagate to all copies.
  - The data is eventually consistent, meaning that a read request immediately after a write operation might not show the latest change.
  - However, DynamoDB offers you the option to request the most up-to-date version of the data. To support varied application requirements, DynamoDB supports both eventually consistent and strongly consistent read options

Supported Operations in DynamoDB

- Conditional Updates and Concurrency Control
  - In a multiuser environment, it is important to ensure data updates made by one client don't overwrite updates made by another client.This "lost update" problem is a classic database concurrency issue.
  - Suppose two clients read the same item. Both clients get a copy of that item from DynamoDB. Client 1 then sends a request to update the item.
  - Client 2 is not aware of any update. Later, Client 2 sends its own request to update the item, overwriting the update made by Client 1.
  - Thus, the update made by Client 1 is lost. DynamoDB supports a "conditional write" feature that lets you specify a condition when updating an item. DynamoDB writes the item only if the specified condition is met; otherwise it returns an error.
  - In the "lost update" example, client 2 can add a condition to verify item values on the server-side are same as the item copy on the client-side. If the item on the server is updated, client 2 can choose to get an updated copy before applying its own updates

Features

- Fast, Consistent Performance
  - Amazon DynamoDB is designed to deliver consistent, fast performance at any scale for all applications. Average service-side latencies are typically single-digit milliseconds.
  - As your data volumes grow and application performance demands increase, Amazon DynamoDB uses automatic partitioning and SSD technologies to meet your throughput requirements and deliver low latencies at any scale.
- Highly Scalable
  - When creating a table, simply specify how much request capacity you require. If your throughput requirements change, simply update your table's request capacity using the AWS Management Console or the Amazon DynamoDB APIs.
  - Amazon DynamoDB manages all the scaling behind the scenes, and you are still able to achieve your prior throughput levels while scaling is underway.

Features

- Flexible
  - Amazon DynamoDB supports both document and key-value data structures, giving you the flexibility to design the best architecture that is optimal for your application.
  - DynamoDB supports storing, querying, and updating documents. Using the AWS SDK you can write applications that store JSON documents directly into Amazon DynamoDB tables. This capability reduces the amount of new code to be written to insert, update, and retrieve JSON documents and perform powerful database operations like nested JSON queries using just a few lines of code.
  - Amazon DynamoDB supports key-value data structures. Each item (row) is a key-value pair where the primary key is the only required attribute for items in a table and uniquely identifies each item. DynamoDB is schema-less. Each item can have any number of attributes (columns). In addition to querying the primary key, you can query non-primary key attributes using Global Secondary Indexes and Local Secondary Indexes.

Features

- Flexible
  - Amazon DynamoDB supports both document and key-value data structures, giving you the flexibility to design the best architecture that is optimal for your application.
  - DynamoDB supports storing, querying, and updating documents. Using the AWS SDK you can write applications that store JSON documents directly into Amazon DynamoDB tables. This capability reduces the amount of new code to be written to insert, update, and retrieve JSON documents and perform powerful database operations like nested JSON queries using just a few lines of code.
- Event Driven Programming
  - Amazon DynamoDB integrates with AWS Lambda to provide Triggers which enables you to architect applications that automatically react to data changes.
- Fine-grained Access Control
  - Amazon DynamoDB integrates with AWS Identity and Access Management (IAM) for fine-grained access control for users within your organization.
  - You can assign unique security credentials to each user and control each user's access to services and resources.

Features

- Fully Managed
  - Amazon DynamoDB is a fully managed cloud NoSQL database service – you simply create a database table, set your throughput, and let the service handle the rest.
  - You no longer need to worry about database management tasks such as hardware or software provisioning, setup and configuration, software patching, operating a reliable, distributed database cluster, or partitioning data over multiple instances as you scale.

Features

- Provisioned Throughput in Amazon DynamoDB
  - When you create or update a table, you specify how much provisioned throughput capacity you want to reserve for reads and writes.
  - DynamoDB will reserve the necessary machine resources to meet your throughput needs while ensuring consistent, low-latency performance.
  - A unit of read capacity represents one strongly consistent read per second (or two eventually consistent reads per second) for items as large as 4 KB.
  - A unit of write capacity represents one write per second for items as large as 1 KB. Items larger than 4 KB will require more than one read operation.
  - The total number of read operations necessary is the item size, rounded up to the next multiple of 4 KB, divided by 4 KB. For example, to calculate the number of read operations for an item of 10 KB, you would round up to the next multiple of 4 KB (12 KB) and then divide by 4 KB, for 3 read operations.

Accessing DynamoDB

- Amazon DynamoDB is a web service that uses HTTP and HTTPS as a transport and JavaScript Object Notation (JSON) as a message serialization format.
- Your application code can make requests directly to the DynamoDB web service API.
- Instead of making the requests to the DynamoDB API directly from your API Version 2012-08-10 11 Amazon DynamoDB Developer Guide Read Capacity Units application, we recommend that you use the AWS Software Development Kits (SDKs).
- The easy-to-use libraries in the AWS SDKs make it unnecessary to call the DynamoDB API directly from your application. The libraries take care of request authentication, serialization, and connection management.