DynamoDB Features

Amazon DynamoDB is a fully managed NoSQL database service provided by Amazon Web Services (AWS). It offers high performance, scalability, and reliability for applications requiring low-latency access to data. Here are some of its key features:

1. **Fully Managed**: DynamoDB is a fully managed service, meaning AWS handles administrative tasks such as hardware provisioning, setup, configuration, patching, backups, and scaling. This allows developers to focus on building applications rather than managing infrastructure.

2. **Scalability**: DynamoDB can handle any amount of traffic, from a few requests per second to millions of requests per second, and can scale both up and down based on demand. It automatically partitions and distributes data to ensure consistent performance as the workload grows.

3. **High Performance**: DynamoDB delivers single-digit millisecond latency for read and write operations, making it suitable for applications requiring real-time responses.

4. **Flexible Data Model**: DynamoDB supports both document and key-value data models. It allows you to store and retrieve JSON-like documents as well as simple key-value pairs. This flexibility enables developers to model their data in a way that best fits their application's requirements.

5. **Schema-less**: Unlike traditional relational databases, DynamoDB does not require a predefined schema. You can add or remove attributes from items dynamically, which simplifies the development process and accommodates evolving data structures.

6. **Consistent, Single-digit Millisecond Latency**: DynamoDB provides predictable and low-latency performance at any scale. This ensures that applications can deliver responsive user experiences even under heavy load.

7. **ACID Transactions**: DynamoDB supports ACID (Atomicity, Consistency, Isolation, Durability) transactions across multiple items and tables. This allows developers to maintain data integrity and enforce business logic with transactional guarantees.

8. **Global Tables**: DynamoDB Global Tables replicate data across multiple AWS Regions, enabling you to build globally distributed applications with low-latency access to data for users around the world.

9. **Backup and Restore**: DynamoDB provides automated and on-demand backups, enabling point-in-time recovery of data. You can also restore tables or individual items from backups with a few clicks.

10. **Security and Compliance**: DynamoDB integrates with AWS Identity and Access Management (IAM) for fine-grained access control. It also supports encryption at rest and in transit to ensure data security. Additionally, DynamoDB is compliant with various industry standards and regulations, such as PCI DSS, HIPAA, and GDPR.

11. **Integration with AWS Ecosystem**: DynamoDB seamlessly integrates with other AWS services, such as AWS Lambda, Amazon S3, Amazon EMR, Amazon Redshift, and Amazon Kinesis, allowing you to build end-to-end solutions for various use cases.

12. **Pay-Per-Use Pricing Model**: DynamoDB offers a flexible pricing model where you only pay for the resources you consume, making it cost-effective for a wide range of applications.

Core components of Amazon DynamoDB

Tables – Similar to other database systems, DynamoDB stores data in tables. A table is a collection of data. For example, see the example table called People that you could use to store personal contact information about friends, family, or anyone else of interest. You could also have a Cars table to store information about vehicles that people drive.

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. In a People table, each item represents a person. For a Cars table, each item represents one vehicle. Items in DynamoDB are similar in many ways to rows, records, or tuples in other 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, an item in a People table contains attributes called PersonID, LastName, FirstName, and so on. For a Department table, an item might have attributes such as DepartmentID, Name, Manager, and so on. Attributes in DynamoDB are similar in many ways to fields or columns in other database systems.

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.

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 (physical storage internal to DynamoDB) in which the item will be stored.

In a table that has only a partition key, no two items can have the same partition key value.

The People table described in Tables, items, and attributes is an example of a table with a simple primary key (PersonID). You can access any item in the People table directly by providing the PersonId value for that item.

Partition key and sort key – Referred to as a composite primary key, this type of key is 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 (physical storage internal to DynamoDB) in which the item will be stored. All items with the same partition key value are stored together, in sorted order by sort key value.

Secondary indexes

In Amazon DynamoDB, indexes play a crucial role in optimizing query performance by allowing efficient access to data. DynamoDB supports two types of indexes: Global Secondary Indexes (GSI) and Local Secondary Indexes (LSI). Here are the details of each:

1. **Global Secondary Indexes (GSI)**:

- **Definition**: A global secondary index is an index with a partition key and an optional sort key that can be different from those on the base table. It provides a different view of the data, enabling efficient querying based on different access patterns.

- **Features**:

  - Supports both partition and sort keys (composite primary key).

  - Can be created at the time of table creation or added later to an existing table.

  - Allows querying across all partitions in the table, providing flexibility in access patterns.

  - Provides eventual consistency for read operations by default, with an option to request strongly consistent reads.

  - Automatically maintained by DynamoDB, which means DynamoDB manages index updates during write operations to the base table.

2. **Local Secondary Indexes (LSI)**:

- **Definition**: A local secondary index is an index with the same partition key as the base table but with a different sort key. Unlike global secondary indexes, LSIs have the same partition key as the table's primary key.

- **Features**:

  - Supports a composite primary key with the same partition key as the base table and a different sort key.

  - Must be defined at the time of table creation and cannot be added later to an existing table.

  - Query operations are limited to a single partition (the same partition key as the base table).

  - Provides eventual consistency for read operations by default, with an option to request strongly consistent reads.

  - Automatically maintained by DynamoDB, similar to GSIs, during write operations to the base table.

3. **Key Differences**:

- **Partition and Sort Keys**: GSIs have their own partition and sort keys, whereas LSIs share the same partition key as the base table but have a different sort key.

- **Creation Time**: GSIs can be created or modified at any time, while LSIs must be defined at table creation and cannot be added later.

- **Query Scope**: GSIs can query across all partitions in the table, whereas LSIs are limited to a single partition.

- **Consistency**: Both GSIs and LSIs support eventual consistency by default, but strongly consistent reads can be requested for both types.

DynamoDB Global Tables

Amazon DynamoDB Global Tables is a fully managed, multi-region, multi-master database replication feature provided by Amazon Web Services (AWS). It enables you to replicate your DynamoDB tables automatically across multiple AWS Regions, allowing you to build globally distributed applications with low-latency access to data for users around the world. Here are the details of DynamoDB Global Tables:

1. **Multi-Region Replication**:

- DynamoDB Global Tables replicates your tables across multiple AWS Regions globally.

- Data is automatically replicated asynchronously to ensure consistency across Regions.

- Replication happens with a propagation delay, typically ranging from a few milliseconds to seconds, depending on network latency.

2. **Multi-Master Replication**:

- Each replica table in a DynamoDB Global Table operates as a fully independent table with its own read and write capacity settings.

- Changes can be made to any replica table in any Region, and those changes are propagated to all other Regions.

- Conflict resolution for concurrent writes is handled automatically by DynamoDB.

3. **Automatic Failover**:

- DynamoDB Global Tables provides built-in automatic failover capabilities to ensure high availability and fault tolerance.

- In the event of a Region failure or downtime, DynamoDB automatically redirects traffic to healthy Regions without requiring any manual intervention.

- Applications can continue to read and write data without interruption during failover events.

4. **Global Data Access**:

- With DynamoDB Global Tables, you can build applications that provide low-latency access to data for users located in different geographic regions.

- Users are directed to the nearest AWS Region, reducing latency and improving the overall user experience.

- Global Tables support both eventually consistent reads and strongly consistent reads, depending on the consistency requirements of your application.

5. **Cross-Region Backup and Restore**:

- DynamoDB Global Tables can be used in conjunction with on-demand backups and point-in-time recovery to create cross-region backups for disaster recovery purposes.

- Backups taken in one Region can be restored in another Region, providing additional data protection and resilience.

6. **Managed Service**:

- DynamoDB Global Tables is a fully managed service provided by AWS.

- AWS handles all aspects of deployment, configuration, monitoring, and maintenance, allowing you to focus on building your applications rather than managing infrastructure.

DynamoDB RCU and WCU

In Amazon DynamoDB, RCU (Read Capacity Units) and WCU (Write Capacity Units) are units of provisioned throughput that determine the capacity and performance of your DynamoDB tables. Here are the details of RCU and WCU:

1. **Read Capacity Units (RCU)**:

- RCU represents the throughput capacity for read operations (reads per second) on your DynamoDB table.

- One RCU is equal to one strongly consistent read per second, or two eventually consistent reads per second, for items up to 4 KB in size.

- For items larger than 4 KB, additional RCUs are required. Each 4 KB of data or fraction thereof requires one additional RCU for strongly consistent reads, or two additional RCUs for eventually consistent reads.

- RCUs can be allocated to individual tables based on the expected read traffic.

- You can adjust the provisioned RCU capacity of your table dynamically based on your application's needs, but there may be limits on the frequency of these adjustments.

2. **Write Capacity Units (WCU)**:

- WCU represents the throughput capacity for write operations (writes per second) on your DynamoDB table.

- One WCU is equal to one write operation per second for items up to 1 KB in size.

- For larger items, additional WCUs are required. Each 1 KB of data or fraction thereof requires one additional WCU.

- Similar to RCUs, WCUs can be allocated to individual tables based on the expected write traffic.

- You can adjust the provisioned WCU capacity of your table dynamically based on your application's needs, subject to certain limits.

3. **Provisioned Throughput Capacity**:

- Provisioned throughput capacity (RCUs and WCUs) is specified when creating or updating a DynamoDB table.

- DynamoDB allocates and manages the underlying resources required to meet the provisioned throughput capacity.

- You are billed for the provisioned throughput capacity that you specify, regardless of whether you use it fully or not.

4. **Dynamic Scaling**:

- DynamoDB offers dynamic scaling, which automatically adjusts the provisioned throughput capacity of your tables in response to changes in traffic patterns.

- With auto-scaling enabled, DynamoDB adjusts the provisioned throughput capacity within defined upper and lower bounds to accommodate varying workloads.

- Auto-scaling helps optimize costs by ensuring that you have sufficient capacity to handle workload spikes without over-provisioning resources.