**AWS DYNAMO DB**

**What is a Non-Relational Database?**

* Non-Relational Database is a database that does not follow the relational database model provided by the relational database management system.
* It is a NoSQL database, we have seen a steady growth in Non-Relational Database with the rise in big data applications.
* We have a Non-Relational Database. Inside the database, we have a collection. Inside a collection, we have got a document, and inside the document, we have key-value pairs. If we talk in a relational sense then Collection is a table, the document is a row, and row consists of key-value pairs.

**How does a Non-Relational database work?**

* A Non-Relational database model uses a variety of different data models such as key-value, document, Graph, in-memory, and search.

For example.

* **In a relational database**, a book record consists of separate tables, and the relationship between tables are defined by primary and foreign constraints. For example, Book table has three columns, i.e., Book id, Book title and Edition Number, Author table has three columns, i.e., Author id, Author name, and Book id. The relationship model is designed so that the database can enforce referential integrity between the tables to reduce redundancy.
* **In Non-Relational database**, records are stored in the form of json format. Each book item such as Book id, Book title, Edition Number, Author id, Author name is stored as attributes in a document.

**Why to use Non-Relational database**

Non-Relational database is used because of the following features:

**Flexibility**: It has a very flexible data model which provides faster and iterative development. The flexible model of Non-Relational database makes an ideal for structured, semi-structured and unstructured data.

**Scalability:** Non-Relational databases provide scaling out by using distributed clusters of hardware rather than scaling up by adding expensive servers.

**High-performance:** Non-Relational databases use some specific data models such as key-value, document, etc. that provides higher performance than relational databases.

**Highly functional**: Non-Relational databases provide highly functional APIs and data types for their respective data models.

**Non-Relational database types**

* Key value
* Document
* Graph
* In-memory
* Search

**Key value**

* A Key-value database is a non-relational database which stores the data in the form of key value.
* Key-value databases provide highly partition able and horizontal scaling that other types of databases cannot achieve.
* Key-value databases are used by those enterprises who want to store large volumes of data without any performance overhead.
* A key in the key-value pair must be unique, and it allows you to access the value associated with the key.
* Redis, Riak, and Oracle NoSQL are examples of key-value databases.

**Document**

* The document database is a non-relational database used to store the semi structured data as documents.
* The document database is required for developers as data in the application tier is represented as JSON format.
* In a Document database, a document can have the same or a different data structure.
* Documents are grouped into collections which behave similarly as a table in the relational database.
* The document database is very popular as it allows you to persist the data in a database by using the same model that you use in your application code.
* Apache CouchDB, MongoDB are examples of a Document database.

**Graph**

* A Graph database is a network database represented by edges and nodes to store the data.
* A Graph can be easily transformed from one model to another model by using a Graph database.
* Nodes have some relationships, which is represented by edges between the nodes.
* Some defined properties are associated with both nodes and edges
* Traversing the joins in a Graph database is very fast because the relationship between the nodes is not calculated at query time but it is persisted in a database

**In-memory**

* An in-memory database is a type of non-relational database that depends on memory for data storage rather than storing the data on disk or SSDs.
* It minimizes the response time by eliminating the need to access disks.
* Since the data is stored and managed in main memory, therefore it is at risk of data lost on server failure.
* An In-memory database is ideal for applications that require microsecond response time.
* The use cases of In-memory database are Real-time bidding, Gaming Leaderboards, and caching.

**Search**

* Search database is a non-relational database which is used to search the data content.
* It uses indexes to categorize the similar characteristics among the data, and facilitate the search capability.
* It is mainly used with data that may be long, semi structured or unstructured data.
* It offers some specialized methods such as full-text search, complex search expressions, and ranking of search results.

**What is DynamoDB?**

* Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that require consistent single-digit millisecond latency at any scale.
* It is a fully managed database that supports both document and key-value data models.
* Its flexible data model and performance makes it a great fit for mobile, web, gaming, ad-tech, IOT, and many other applications.
* It is stored in SSD storage.
* It is spread across three geographically data centers.

Because of its availability in three geographically data centers, it consists of two different types of consistency models:

* Eventual Consistent Reads
* Strongly Consistent Reads

**Eventual Consistent Reads**

* It maintains consistency across all the copies of data which is usually reached within a second. If you read a data from DynamoDB table, then the response would not reflect the most recently completed write operation, and if you repeat to read the data after a short period, then the response would be the latest update. This is the best model for Read performance.

**Strongly Consistent Reads**

A strongly consistent read returns a result that reflects all writes that received a successful response prior to the read.

**Note**: If your application wants the data from DynamoDB table immediately, then choose the Strongly Consistent Read model. If you can wait for a second, then choose the Eventual Consistent Model.

**AWS DynamoDB Throughput Capacity**

DynamoDB throughput capacity depends on the read/write capacity modes for performing read/write operation on tables.

There are two types of read/write capacity modes:

* Provisioned mode
* On-demand mode

**Provisioned mode**

It defines the maximum amount of capacity that an application can use from a specified table.

In a provisioned mode, you need to specify the number of reads and writes per second required by the application.

If the limit of Provisioned mode throughput capacity is exceeded, then this leads to the request throttling.

A provisioned mode is good for applications that have predictable and consistent traffic.

**The Provisioned mode consists of two capacity units:**

* Read Capacity unit
* Write Capacity unit

**Read Capacity Unit**

The total number of read capacity units depends on the item size, and read consistency model.

Read Capacity unit represents two types of consistency models:

Strongly Consistent model: Read Capacity Unit represents one strong consistent read per second for an item up to 4KB in size.

Eventually Consistent model: Read Capacity Unit represents two eventually consistent reads per second for an item up to 4KB in size.

DynamoDB will require additional read capacity units when an item size is greater than 4KB. For example, if the size of an item is 8KB, 2 read capacity units are required for strongly consistent read while 1 read capacity unit is required for eventually consistent read.

**Write Capacity Unit**

The total number of write capacity unit depends on the item size.

Only 1 write capacity unit is required for an item up to size 1KB.

DynamoDB will require additional write capacity units when size is greater than 1KB. For example, if an item size is 2KB, two write capacity units are required to perform 1 write per second.

For example, if you create a table with 20 write capacity units, then you can perform 20 writes per second for an item up to 1KB in size.

**On-Demand mode**

DynamoDB on-demand mode has a flexible new billing option which is capable of serving thousands of requests per second without any capacity planning.

On-Demand mode offers pay-per-request pricing for read and write requests so that you need to pay only for what you use, thus, making it easy to balance costs and performance.

In On-Demand mode, DynamoDb accommodates the customer's workload instantly as the traffic level increases or decreases.

On-Demand mode supports all the DynamoDB features such as encryption, point-in-time recovery, etc except auto-scaling

If you do not perform any read/write, then you just need to pay for data storage only.

On-Demand mode is useful for those applications that have unpredictable traffic and database is very complex to forecast.