AWS DynamoDB

- Amazon DynamoDB is a fully managed NoSQL database service that
  - makes it simple and cost-effective to store and retrieve any amount of data and serve any level of request traffic.
  - provides fast and predictable performance with seamless scalability
- DynamoDB enables customers to offload the administrative burdens of operating and scaling distributed databases to AWS, without having to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling.
- DynamoDB tables do not have fixed schemas, and the table consists of items and each item may have a different number of attributes.
- DynamoDB synchronously replicates data across three facilities in an AWS Region, giving high availability and data durability.
- **Durability, performance, reliability, and security are built in, with SSD (solid state drive) storage and automatic 3-way replication.**
- DynamoDB supports two different kinds of primary keys:
  - **Partition Key** (previously called the **Hash key**)
    - A simple primary key, composed of one attribute
    - The partition key value is used as input to an internal hash function; the output from the hash function determines the partition where the item will be stored.
    - No two items in a table can have the same partition key value.
  - **Partition Key and Sort Key** (previously called the **Hash and Range key**)
    - A composite primary key is composed of two attributes. The first attribute is the partition key, and the second attribute is the sort key.
    - The partition key value is used as input to an internal hash function; the output from the hash function determines the partition where the item will be stored.
    - All items with the same partition key are stored together, in sorted order by sort key value.
    - The combination of the partition key and sort key must be unique.
    - It is possible for two items to have the same partition key value, but those two items must have different sort key values.
- DynamoDB Table classes currently support
  - DynamoDB Standard table class is the default and is recommended for the vast majority of workloads.
  - DynamoDB Standard-Infrequent Access (DynamoDB Standard-IA) table class which is optimized for tables where storage is the dominant cost.
- DynamoDB Throughput Capacity determines the read/write capacity for processing reads and writes on the tables and it currently supports

    - Provisioned – maximum amount of capacity in terms of reads/writes per second that an application can consume from a table or index
    - On-demand – serves thousands of requests per second without capacity planning.
- DynamoDB Secondary indexes
    - add flexibility to the queries, without impacting performance.
    - are automatically maintained as sparse objects, items will only appear in an index if they exist in the table on which the index is defined making queries against an index very efficient
- DynamoDB throughput and single-digit millisecond latency make it a great fit for gaming, ad tech, mobile, and many other applications

Partitions and data distribution

Amazon DynamoDB stores data in partitions. A *partition* is an allocation of storage for a table, backed by solid state drives (SSDs) and automatically replicated across multiple Availability Zones within an AWS Region. Partition management is handled entirely by DynamoDB—you never have to manage partitions yourself.

When you create a table, the initial status of the table is CREATING. During this phase, DynamoDB allocates sufficient partitions to the table so that it can handle your provisioned throughput requirements. You can begin writing and reading table data after the table status changes to ACTIVE.

More about partitions:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.Partitions.html

DynamoDB Consistency

- Each DynamoDB table is automatically stored in the three geographically distributed locations for durability.
- Read consistency represents the manner and timing in which the successful write or update of a data item is reflected in a subsequent read operation of that same item.
- DynamoDB allows the user to specify whether the read should be eventually consistent or strongly consistent at the time of the request
    - **Eventually Consistent Reads** (Default)
        - Eventual consistency option maximizes the read throughput.
        - Consistency across all copies is usually reached within a second
        - However, an eventually consistent read might not reflect the results of a recently completed write.

    - Repeating a read after a short time should return the updated data.
    - DynamoDB uses eventually consistent reads, by default.
  - **Strongly Consistent Reads**
    - Strongly consistent read returns a result that reflects all writes that received a successful response prior to the read
    - Strongly consistent reads are 2x the cost of Eventually consistent reads
    - Strongly Consistent Reads come with disadvantages
      - A strongly consistent read might not be available if there is a network delay or outage. In this case, DynamoDB may return a server error (HTTP 500).
      - Strongly consistent reads may have higher latency than eventually consistent reads.
      - Strongly consistent reads are not supported on global secondary indexes.
      - Strongly consistent reads use more throughput capacity than eventually consistent reads.

DynamoDB Throughput Capacity

- DynamoDB throughput capacity depends on the read/write capacity modes for processing reads and writes on the tables.
- DynamoDB supports two types of read/write capacity modes:
  - Provisioned – maximum amount of capacity in terms of reads/writes per second that an application can consume from a table or index
  - On-demand – serves thousands of requests per second without capacity planning.
- DynamoDB Auto Scaling helps dynamically adjust provisioned throughput capacity on your behalf, in response to actual traffic patterns.

DynamoDB Secondary Indexes

- DynamoDB Secondary indexes on a table allow efficient access to data with attributes other than the primary key.
- DynamoDB Secondary indexes support two types
  - **Global secondary index –** an index with a partition key and a sort key that can be different from those on the base table.
  - **Local secondary index –** an index that has the same partition key as the base table, but a different sort key.

DynamoDB pricing

DynamoDB is charging for three things:

- **Read consumption**: Measured in 4KB increments (rounded up!) for each read operation. This is the amount of data that is read from the table or index. Each increment is referred as an 'RCU' (read capacity unit). Thus, if you read a single 10KB item in DynamoDB, you will consume 3 RCUs (10 / 4 == 2.5, rounded up to 3).
- **Write consumption**: Measured in 1KB increments (also rounded up!) for each write operation. This is the size of the item you're writing / updating / deleting during a write operation. Each write increment is referred as a 'WCU' (write capacity unit). Thus, if you write a single 7.5KB item in DynamoDB, you will consume 8 WCUs (7.5 / 1 == 7.5, rounded up to 8).
    - **Storage**: Measured in GB-months. This is the amount of data stored in the table or index, multiplied by the number of hours in the month.