**Relational (SQL) vs Non-Relational (NoSQL)**

In short, relational databases use predefined schema and store data in rows and columns like a spreadsheet, whereas non-relational databases such as DynamoDB have dynamic schema, are document-oriented, and scale horizontally.   
‍

**Relational (SQL)**

Relational Databases make use of Structured Query Language (SQL) to interact and have a predefined schema. SQL is a very well-known query language that most Database Administrators (DBAs) are familiar with.

The SQL schemata always have relational and tabular data, containing rules to ensure consistency and integrity. They contain tables with columns (attributes) and rows (records), and     keys have constrained logical relationships. SQL databases abstract data as a set of tuples organized into relations, which allows for abstraction over the physical representation of data and access paths.

SQL databases can be scaled vertically, by increasing the processing hardware power (for example, increasing a single server workload by increasing RAM, CPU, SSD, etc).

**Advantages of Relational (SQL)**

* The use of SQL language.
* The atomicity of operations on the database. The database's entire operation is done or it’s not done at all, using the rollback technique.
* The use of flexible queries.

**Disadvantages of Relational (SQL)**

* Requires mindful, up-front design to ensure adequate performance and resistance to evolution.
* Changes to the schema may result in downtime.
* Limited horizontal scalability.  
  ‍

**Non-Relational (NoSQL)**

NoSQL is a class of Database Management Systems that are non-relational and do not use SQL, but have dynamic schemas for unstructured data. They can have different types based on their data model:

* Document
* Key-Value
* Wide-Column
* Graph

NoSQL is the best database class for large amounts of data or data sets that are changed frequently. If you are working with large amounts of unstructured data, document databases are a great option.

Whereas, as mentioned before, SQL databases can be scaled vertically, NoSQL can be scaled horizontally, which makes the scalability way easier by increasing the number of servers or nodes.

**Advantages of Non-Relational (NoSQL)**

* Easy scalability and high availability
* Flexibility on database models
* High-performance

**Disadvantages of Non-Relational (NoSQL)**

* Some databases do not contemplate the atomicity of instructions and the integrity of the data.
* Lack of standardization

Now that we understand the core concepts of relational and non-relational databases, we can see what database options AWS provides and how they make data storage easier.   
‍

**RDS (SQL) vs DynamoDB (NoSQL)**

The primary advantage of RDS and DynamoDB is that they are fully managed by AWS.

But, what does being a fully managed service mean? It means that the underlying operating system and some core components of the service are completely managed by AWS. AWS automatically performs routine tasks such as provisioning, patching, backup, recovery, failure detection, and repair.

This is in contrast with running a database on-premises or on an EC2 instance (AWS service used to create virtual machines), and having to take care of the operating system, patching, backups, and availability. Instead, by using RDS or DynamoDB, you’re reducing administrative overhead and the best part of using one of these two services is that all you need to do is take care of the data.

Now, let's talk about the particulars of each service.  
‍

**RDS**

The Amazon RDS service allows you to set up, operate, and scale relational (SQL) databases on AWS.

Amazon RDS makes it easy to use replication to enhance availability and reliability for production workloads. AWS provides six SQL-based database engine options:

* Amazon Aurora
* MySQL
* MariaDB
* PostgreSQL
* Oracle
* Microsoft SQL Server

AWS provides several instance types with different combinations, such as CPU, memory, storage options, and networking capacity. Each type comes in a variety of sizes to suit the needs for your workload.

By choosing one of the engines above along with its version and specifying some configurations like instance type, storage size, and network specs, you can spin up a new fully-managed relational database on AWS.

**Features of RDS**

* **Multi-AZ:** RDS uses two replicated databases running at the same time in different zones. This ensures high availability during an availability zone failure. The primary database is synchronously replicated to a secondary database in another zone, and in case of a failure, it automatically performs a fail-over from the primary to the second.
* **Read Replicas:** Read replicas allow different use cases. For example, you can scale in for read-heavy database workloads.
* **Automatic Backups:** You can restore to any point in time within your backup retention period. AWS takes automated snapshots, so you don't need to do anything.
* **Patching:** As mentioned above, patching is an AWS responsibility. You can select a maintenance window for it and AWS will take care of it without any downtime if Multi-AZ is selected.
* **Monitoring:** You can monitor metrics such as CPU, memory, and disk usage/throughput using AWS CloudWatch.
* **Storage:** RDS provides three types of storage:  
  - General-purpose SSD  
  - Provisioned IOPS SSD  
  - MagnetiC
* ‍**Encryption:** You can opt for managing your encryption keys using the AWS Key Management Service (KMS). All snapshots, backups, read replicas, and standby instances are also encrypted. RDS uses SSL to secure data in transit.

**The Advantages of Amazon Aurora as an RDS Option**  
**‍**Amazon Aurora is a fully managed relational database engine created by AWS and it is compatible with MySQL and PostgreSQL. It is up to five times faster than standard MySQL databases and three times faster than standard PostgreSQL databases.

AWS also offers the **Aurora Serverless**, which is an on-demand, auto-scaling configuration for Amazon Aurora. It automatically starts up, shuts down, and scales capacity up or down based on your application's needs. It enables you to run your database in AWS without managing any database capacity.

**Application Use Cases**

* Enterprise resource planning (ERP)
* Customer relationship management (CRM)
* Finance data
* Transactions

AWS RDS options like Amazon Aurora are great options if you want to make use of a relational database without administrative overhead and high availability for your application.  
‍

**DynamoDB**

DynamoDB is the option that AWS offers for non-relational (NoSQL) databases. As we learned above, there are multiple types of non-relational databases, and DynamoDB supports the following data structures:

* Document
* Key-Value

Like RDS, DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with easy scalability.

The key point is that AWS provides DynamoDB as a **serverless** solution. By using DynamoDB, there are no servers to provision, patch, or manage, and no software to install, maintain, or operate. DynamoDB automatically scales tables to adjust for capacity and maintains performance with zero administration.

In DynamoDB, tables, items, and attributes are the core components that you work with. A table is a collection of items, and each item is a collection of attributes. DynamoDB uses primary keys to uniquely identify each item in a table and secondary indexes to provide more querying flexibility.

The only mandatory attribute is the partition key, which is similar to a primary key in a relational database. You also have the option to create a second attribute called sort key, which lets you get a bigger and faster return of the data.

For provisioning and charging, AWS utilizes CUs (Capacity Units) that are informed at the time the table is created.

* **Read Capacity Unit (RCU)**: Each unit represents one consistent or two eventually consistent reads per second, for a 4KB item.
* **Write Capacity Unit (WCU):** Each unit represents one write per second, for a 1KB item.

**Features of DynamoDB**

* **High Availability:** It automatically replicates your data across multiple availability zones to meet availability and durability requirements.
* **Scalability:** For tables using provisioned capacity, DynamoDB automatically scales the throughput and storage based on your previously set capacity by monitoring the performance usage of your application.
* **Global Tables:** If you select this option, DynamoDB replicates your data automatically across your choice of AWS regions, to ensure high availability.
* **Backups:** On-demand backups allow you to create full backups of your DynamoDB table data for archiving. You also have point-in-time recovery, which allows you to protect your tables from accidental write or delete operations.
* **Encryption:** DynamoDB encrypts all your data at rest by default. It uses encryption keys stored in AWS Key Management Service (KMS).
* **Monitoring:** You can monitor metrics using AWS CloudWatch.

**DynamoDB Accelerator (DAX)**  
**‍**Amazon DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache that delivers up to 10 times performance improvement.

DAX does all the heavy lifting without requiring developers to manage cache invalidation, data population, or cluster management.

**Application Use Cases**  
**‍**This AWS solution is ideal for applications that require a high speed in data reading and writing.

* Real-time bidding
* Shopping carts
* Mobile applications
* High I/O needs