RELATIONAL DATABASE SERVICES (RDS)

❖ What is DATA?

In simple words, data can be facts related to any object. For e.g.: your age, job, house no, contactno., name, places are some data related to you.

❖ What is DATABASE?

Database is a systematic collection of data. Databases supports storage and manipulation of data: Facebook, telecom companies, amazon.com

❖ What is DBMS?

DBMS is a collection of programs which enable its users to access database, manipulate data, reporting/ representation of data.

❖ Types of DBMS

1. Hierarchical
2. Network
3. Relational
4. None Relational
5. Object oriented
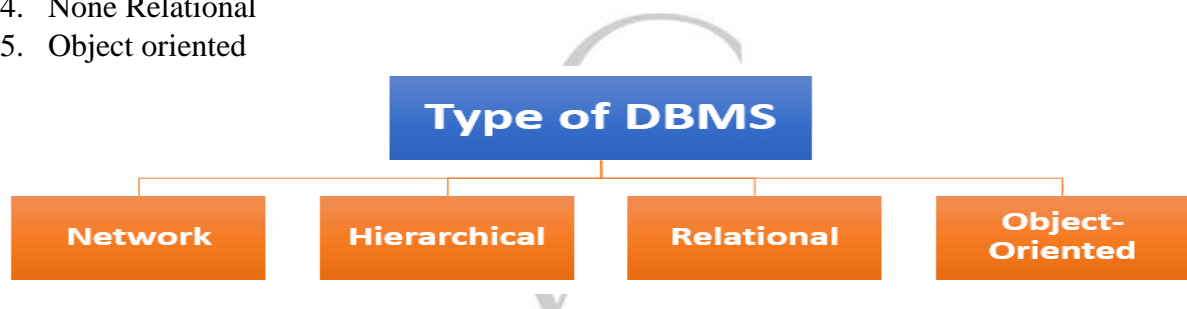
❖ Relational Database:

➢ A relational database is a data structure that allows you to link information from different tables of different types of data bucket.
➢ Tables are related to each other.
➢ All fields must be filled.
➢ Best suited for OLTP (online transaction processing)
➢ Relational DB: MySQL, Oracle, DBMS, IBM DB2
➢ A row of a table is also called records. It contains the specific information of each individual entry in the table.
➢ Each table has its own primary key.
➢ It normalizes data into structures.
➢ It means it required a schema (design of database) is used to strictly define tables, columns, indexes andrelation between tables.
➢ Relational DB are usually used in enterprises application/scenario. Exception inMySQL which is used for web application.
➢ Common application for MySQL includes php and java-based web applications that requires a database storage backend. E.g.: JOOMLA
➢ Cannot scale out horizontally.

➢ Virtually all relational DB uses SQL.

➢ Requires High end H/W as its performance is dependent on that (complex Querying)

❖ Non-Relational DB/NO-SQL DB:

➢ Non-relational databases store data without a structured mechanism to link data from different tables to one another.

➢ It is simplest from DB, non -relational DB store data without a structured mechanism to link data from diff. tables to one another.

➢ They are high performance that are non-schema based unlike relational DB.

➢ Required low-cost hardware.

➢ Much faster performance (read/write) compared to relational Dbs.

➢ Easier to develop

➢ Horizontal scaling is possible.

➢ Never provide tables with flat fixed column records. It means schema-free.

➢ Best suited for online analytical processing (OLAP).

➢ E.g.: of NoSQL databases: MongoDB, Cassandra, DynamoDB, Postage, Raven, Redis.

	SQL	**NoSQL**
Database Type	Relational Databases	Non-relational Databases / Distributed Databases
Structure	Table-based	• Key-value pairs • Document-based • Graph databases • Wide-column stores
Scalability	Designed for scaling up vertically by upgrading one expensive custom-built hardware	Designed for scaling out horizontally by using shards to distribute load across multiple commodity (inexpensive) hardware
Strength	• Great for highly structured data and don't anticipate changes to the database structure • Working with complex queries and reports	• Pairs well with fast paced, agile development teams • Data consistency and integrity is not top priority • Expecting high transaction load

❖ Types of No-SQL Databases:
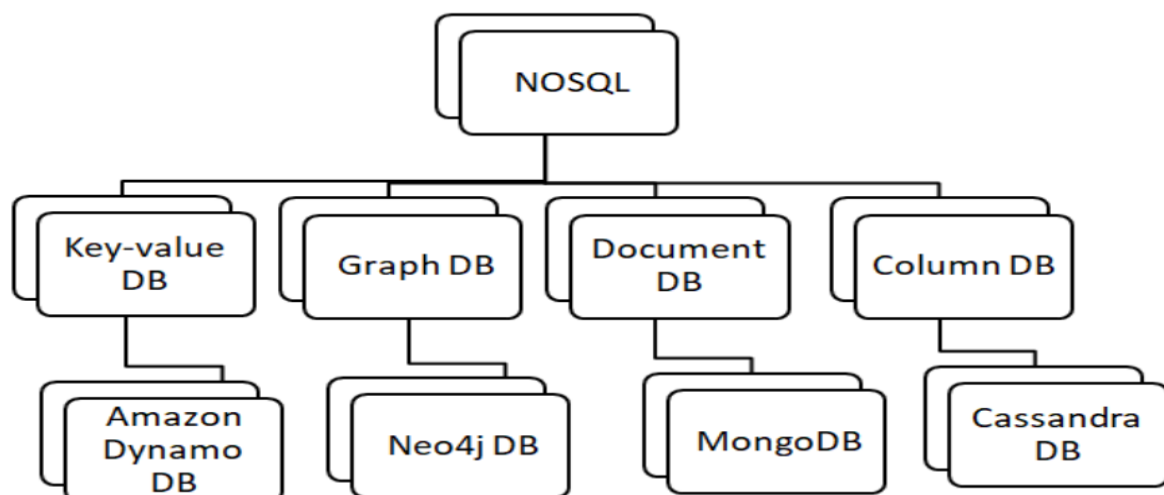
1. **Columnar Database (Cassandra, HBase)**
2. **Document Database (MongoDB, CouchDB, Raven DB)**
3. **Key Value Database (Redis, Riak, DynamoDB, Tokyo Cabinet)**
4. **Graph Based Database (Neo4J, Flock DB)**

1. <u>**Columnar Database:**</u>

➢ A columnar database is a DBMS that stores data in columns instead of Rows.
➢ In a columnar DB all the column-1 values are physically together followed by all the column-2 values.
➢ In a row-oriented DBMS, the data would be stored like this: (1, bob, 30, 8000: 2, Arun, 26, 4000: 3, vain, 39, 2000 ;)
➢ In a column-based DBMS the database would be stored like this: (1, 2, 3: bob, Arun, vain; 30, 26, 39; 8000, 4000, 2000 ;)
➢ Benefit is that because a column-based DBMS is self-indexing, it uses less disk space that a RDBMS containing the same data. It easily performs operation like min, max, average.

2. <u>**Document Database:**</u>

➢ Document DB make it easier for developer to store and querying data in a DB by using the same document model format they use in their application code.
➢ Document DBs are efficient for storing catalogue.
➢ Store semi-structure data as document typically in JSON or XML format. (example)
➢ A document database is a great choice for contain management application such as blogs and video platform.

3. <u>**Key-Value Database:**</u>

➢ A key-value DB is a simple DB that uses an associative array (like dictionary) as a fundamental model where each key is associated with one and only one value in a collection.
➢ It allows horizontal scaling.
➢ Used cases: shopping cart, and session store in app like fb and twitter.
➢ They improve application performance by storing critical pieces of data in memory for low latency access.
➢ Amazon elastic ache as an in-memory key-value stores.

4. <u>**Graph Based Database:**</u>

➢ A graph DB is basically a collection of nodes and edges.
➢ Each node represents an entity and each edges represent a connection or relationship between two nodes.
➢ In an AWS fully managed relational DB engines service where AWS is responsible for:
• Security and patching.
• Automated backup.
• Software updates for DB engine.
• If selected multi-AZ with synchronous replication between the activeand stand by DB instances.
• Automatic failover if multi-AZ option was selected.
• By default, every DB has weekly maintenance window. (Max 35 days.)

Settings managed by the users:
- Managing DB settings.
- Creating relational database schema.
- Database performance tuning.

Types of Non-Relational Databases

upwork

Types	Performance	Scalability	Flexibility	Complexity
Key-Value Store	High	High	High	None
Column Store	High	High	Moderate	Low
Document	High	Variable (High)	High	Low
Graph Database	Variable	Variable	Hight	High

Categories of NoSQL Databases

Category	Key-Value Store	Tabular (Column-Family / BigTable)	Document Database	Graph Database
Based on	Amazon's Dynamo Paper	Google's BigTable Paper	Lotus Notes	Euler & Graph theory
Data Modal	Global Collection of K-V pair	BigTable, Column Families	K-V collections	Nodes, Rels, K-V on both
Data Size Handling (scalability)	Highest (stores only K-V so, distributed to multiple nodes)	Next Higher	Next Higher	Smallest (Limited to single node)
Data Complexity (connectivity /relationship)	-	Little	Document link	Highest
Example	Riak, Voldemort, (Tokyo	Hbase, Hypertable, Cassendra (column-family), MonetDB	CouchDB, MongoDB	Neo4j, Allergo , Sones

❖ RDS

➢ It is a fully **managed** relational **DB** engine **service** where AWS is responsible for:
- Security and patching of the DB instances.
- Automated Backup for your DB instances (default setting)
- Software updated for the Database DB Engine.
- If multi-AZ is selected then **Synchronous Replication** between the active and standby DB instances in same Region.
➢ Automatic Failover if multi-AZ option was selected at launch.
➢ Providing the ability to create DB read replicas for DB scaling (intensive read deployments)
➢ Every DB instance has a weekly maintenance window – if you did not specify one at the time you create the DB instance, AWS will choose one randomly for you (30 min long)

- AWS is not responsible for
  - Managing DB Settings.
  - Building a relational DB Schema.
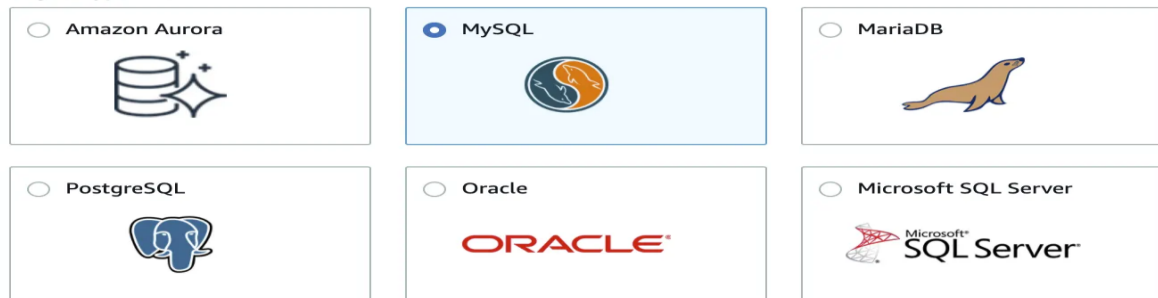 - DB performance tuning

❖ Relational Database Engine Options:

1. MS SQL Sever
2. My SQL: supports 64TB of DB
3. Oracle
4. AWS Aurora: high throughput
5. Postgres SQL: highly reliable and stable
6. Maria DB: MySQL compatible, 64TB DB

There are two Licensing Options:
1. BYOL (Bring Your Own License)
2. License from AWS on hourly basis

RDS Limits:
- Up to 40DB instances per account.
- 10 of this 40 can be Oracle or MS-SQL server under license included model. Or
- Under BYOL model, all 40 can be any DB engine you need.

❖ RDS Instance Storage:

➢ Amazon RDS use EBS volumes (not instance store) for DB and logs storage.

1. **General Purpose**: use for DB workloads with moderate I/O requirement.
   Limits: min: 20 GB Max: 16384 GB

2. **Provisional IOPS RDS Storage:** use for high performance OLTP workloads.
   Limits: min: 100GB Max: 16384GB

❖ Templates available in RDS:

a. Production
b. Dev/Test
c. Free-Tier

❖ DB Instance Size:

a. Standard class
b. Memory-Optimized class
c. Burstable class

❖ What is Multi-AZ in RDS:

➢ You can select multi-AZ option during RDS DB instance launch.
➢ RDS service creates a standby instance in a different AZ in the same region and configure "synchronous replication" between the primary and standby.
➢ You cannot read/write to the standby RDS DB instances.
➢ You cannot select which AZ in the region will be chosen to create the standby DB instance.
➢ You can however view, which AZ is selected after the standby is created.
➢ Depending on the instance class it may take 1 to few minutes to failover to the standby instance.
➢ Recommended to implement DB connection retries into your application, so failover doesn't create an issue.
➢ Running a DB instance as multi-AZ deployment can further reduce the impact of a maintenance event because amazon RDS applies OS updates by following steps:
  • Perform maintenance on the standby
  • Promote the standby to primary
  • Perform maintenance on the old primary which becomes the new standby
➢ AWS recommends the use of provisioned IOPS instances for multi-AZ RDS instance.
➢ When you modify the DB engine for your DB instance in a multi-AZ deployment.
➢ Amazon RDS upgrades both the primary and secondary DB instances at the same time.
➢ In this case the database engine for the entire multi-AZ deployment is shutdown during the upgrade.

What are Multi-AZ deployments?

📦 Automatic, synchronous replication of your database to a standby instance in a different Availability Zone
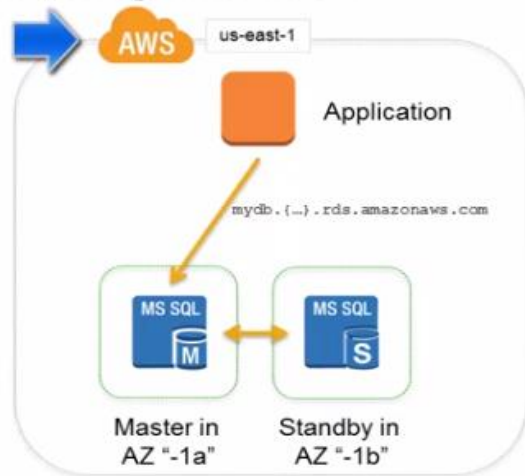
❖ When Multi-AZ RDS Failover Triggers:

➢ In case of failure of primary DB instance failure.
➢ In case of AZ failure.
➢ Loss of network connectivity to primary DB.
➢ Loss of primary EC2 instance failure.
➢ EBS failure of primary DB instance.
➢ The primary DB instance is changed.
➢ Patching the O.S of the DB instance.
➢ Manual failover. (In case of rebooting.)

❖ Multi-AZ RDS Failover Consequences:

➢ During failover the CNAME of the RDS DB instance is updated to map to the standby IP address.
➢ It is recommended to use the end point to reference your DB instances and not its IP address.
➢ The CNAME doesn't change because the RDS endpoint doesn't change.
➢ RDS end point doesn't change by selecting multi-AZ option, however the primary and standby instances will have different IP addresses, as they are in different AZ.
➢ It is always recommended that you do not use the IP address to point RDS instances, always use endpoint. By using endpoint there will be no change whenever a failover happens.

❖ When we do manual failover?

➢ In case of rebooting.
➢ This is by selecting the "reboot with failover" reboot options on the primary RDS DB instances.
➢ A DB instance reboot is required for changes to take effect when you change the DB parameter group on when you change a static DB parameter.
➢ Whenever failover occurs AWS RDS sends SNS notification.

➢ You can use API calls to find out the RDS events occurred in the last 14 days.
➢ Even you can use CLI to view last 14 days events.
➢ Using AWS console, you can only last one day events
➢ In case of OS patching, system upgrades and DB scaling, these things happen on standby first then on primary to avoid outage.
➢ In multi-AZ, snapshots and automated backups are done on standby instance to avoid I/O suspension on primary.
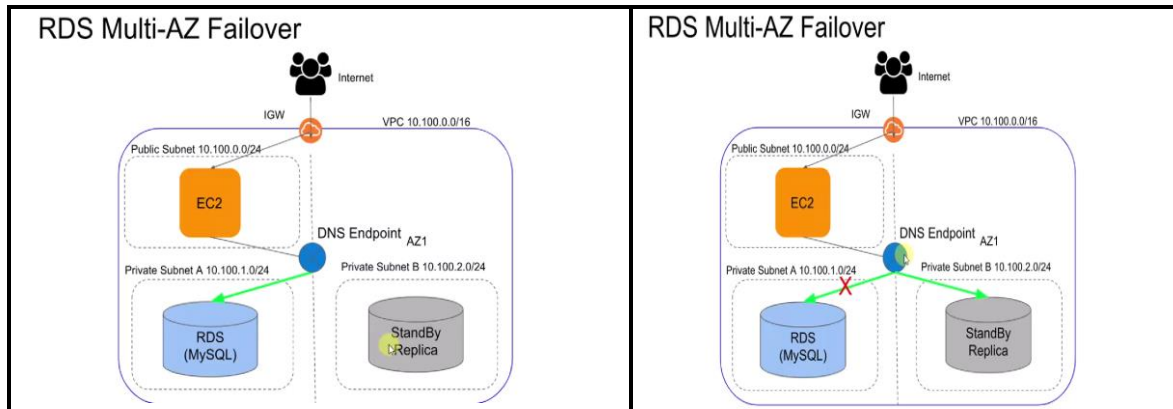
❖ RDS Multi-AZ Deployment Maintenance:

➢ Firstly, perform maintenance of standby.
➢ Now convert standby into primary so that maintenance can be done on primary. (currently)
➢ You can manually upgrade a DB instance to a supported DB engine version from AWS console as follows: - RDS->DB instance->modify DB->set DB engine version.
➢ By default, change will take effect during the next maintenance window.
➢ Or you can force an immediate upgrade if you want.
➢ In multi-AZ version upgrade will be conducted by both primary and standby at the same time which will cause an outage.
➢ Do it during maintenance window.

❖ There are two methods to backup and restore your RDS DB instances:

1. **AWS RDS automated backup**
2. **User initiated manual backup**

➢ Either you can take backup of entire DB instance or just the DB.
➢ You can create a restore volume snapshots of your entire DB instances.
➢ Automated backups by AWS, backup your DB data to multiple AZ to provide for data durability.
➢ Select-automated backup in AWS console.
➢ Stored in Amazon S3.
➢ Multi-AZ automated backups will be taken from the standby instance.
➢ The DB instance must be in "ACTIVE" state for automated backup.
➢ RDS automatically backups the DB instances daily by creating a storage volume snapshot

of your DB instance (fully daily snapshots) including the DB transaction logs.
➢ You can decide when you would like to take backup (window)
➢ No additional charge for RDS backing up your DB instance.
➢ For multi-AZ deployment, backups are taken from the standby DB instance (true for Maria DB, MySQL, Oracle, Postgre SQL).
➢ Automated backups are deleted when you delete your RDS DB instance.
➢ An outage occurs if you change the backup retention period from zero to non-zero value or the other way around.
➢ It allows you to recover your database to any point in time within a retention period.
➢ Retention period of automate backup is 7 days (by default) via AWS console.
➢ AWS Aurora is an exception. Its default is 1 day.
➢ Via CLI or API 1 day by default.
➢ You can increase it up to 35 days.
➢ If you don't want backup, put zero "0" in the retention period.

❖ Snapshot Backups

➢ DB snapshots are done manually (they are user initiated)
➢ They are stored even after you delete the original RDS instance, unlike automated.
➢ Automated snapshots cannot be shared directly, you need to create a copy to share.
➢ During your daily backup window, your I/O may be suspended (for standalone RDS deployments)
➢ For multi-AZ deployment backup are taken from the standby DB instance.
➢ No additional charges for RDS backing up your DB instances.
➢ In case of manual snapshot, point-in-time recovery is not possible.
➢ Manual snapshot is also stored in S3.
➢ They are not deleted automatically, if you delete RDS instance.
➢ Take a final snapshot before deleting your RDS DB instance.
➢ You can share manual snapshot directly with other AWS Account.
➢ When you restore a DB instance only the default DB parameters and security groups are associated with the restored instance.
➢ You cannot restore a DB snapshot into an existing DB instance rather it has to create a new DB instance it has new endpoint.
➢ Restoring from the backup or a DB snapshot changes the RDS instance endpoint.
➢ At the time of restoring, you can change the storage type (general purpose or provisioned.)
➢ You cannot encrypt an existing unencrypted DB instance.
➢ To do that you need to: create a new encrypted instance and migrate your data to it (from unencrypted to encrypted) or you can restore from a backup/snapshot into a new encrypted RDS instance.
➢ RDS supports encryption-at-rest for all DB engines using KMS.
➢ What actually encrypted when data-at-rest:

         a. All its snapshots.
         b. Backups of DB (S3 storage.)
         c. Data on EBS volume.
         d. Read replica created from the snapshots.

❖ Read Replicas

➢ It allows you to have a read-only copy of your production database.

➢ This is achieved by using asynchronous replication from the primary RDS instance to the read replica.

➢ You can create read replicas primarily for very read heavy database workloads not for DR

➢ **Read Replicas**
- Can be multi-AZ
- Used to increase performance
- Must have backups turned on
- Can be aurora or MYSQL
- Can be promoted to master, this will break the read replica
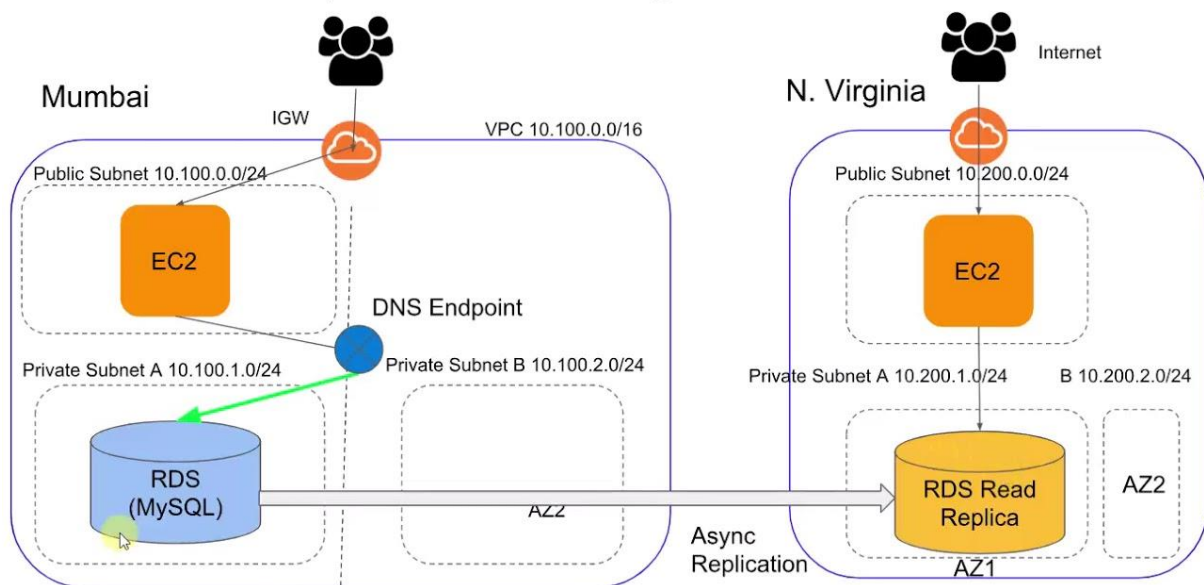
❖ Some points related to RDS Billings:

➢ No upfront cost.

➢ You have to pay only for:

a. **DB instance hours (partial hour charged as full hours)**
b. **Storage GB/month.**
c. **Internet data transfer.**
d. **Backup storage (i.e., S3)**

➢ This is increases by increasing DB backup's retention period. Also charged for:

a. **Multi-AZ DB hours.**
b. **Provisioned stage (multi-AZ)**
c. **Double write I/O**
d. **You are nor charged for DB data transfer during replication from primary to standby.**