**Assignment (2.5)**

**Brief**

Objective:

The objective of this assignment is to gain a deeper understanding of AWS Database as a Service (DBaaS) offerings and how they can be used in different use cases.

Instructions:

* Research and investigate the various AWS DBaaS offerings, including RDS, DynamoDB, and Aurora.
* Choose a specific use case (e.g. web application development, big data processing, machine learning) and determine which AWS DBaaS offering would be the most appropriate for that use case, providing a detailed explanation of your reasoning.
* Write a sample project that demonstrates the use of one of the AWS DBaaS offerings and compare it with the other offerings.
* Write a report summarizing your research, including a comparison and contrast of the different AWS DBaaS offerings, an analysis of the chosen use case, and any recommendations for improvement.

If you are creating Database on AWS:

* Submit your Database type (DynamoDB, RDS, Aurora?)
* Submit your AWS Database name

**Submission**

* Submit the URL of the GitHub Repository that contains your work to NTU black board.
* Should you reference the work of your classmate(s) or online resources, give them credit by adding either the name of your classmate or URL.

**References**

**Comparison of Database Service Choices with Amazon**

|  |  |  |
| --- | --- | --- |
| RDS | Aurora | DynamoDB |
| Amazon Relational Database Service (RDS) is a managed SQL database service provided by Amazon Web Services (AWS).  You get support for the most popular relational database servers, including Oracle, Microsoft SQL Server, MariaDB, MySQL, and PostgreSQL. RDS makes it easy to move on-premise database workloads to the cloud and helps automate and offload all the time-consuming administrative tasks associated with managing databases. Many organizations look to adopt RDS because Amazon handles high availability, recovery, backups, and patching. With Multi-AZ deployment, Amazon RDS automatically creates synchronous master/slave pairs across availability zones. If an unplanned outage happens, RDS can automatically failover to a standby replica in a different availability zone. This provides several benefits: data redundancy, failover support and minimal latency during system backups.  AWS RDS read replicas provide the benefit of having multiple read-only copies of your database instance within the same or different AWS region. Any updates made to the source database are asynchronously copied to the read replicas. You get multiple benefits: fault-tolerant availability, load balancing for high-volumes of read traffic, scalability for read-heavy database workloads. You also have multiple DB support – read replicas are available MySQL, MariaDB, PostgreSQL, Oracle, and SQL Server as well as Aurora. | Amazon Aurora is a relational database engine that can be run on RDS or as Aurora Serverless. Aurora is MySQL and PostgreSQL compatible but  Aurora databases can be set up quickly,and applications are configured to access Aurora databases using existing code, drivers, and programs with minimal changes thanks to its compatibility with MySQL and PostgreSQL. Aurora is designed for fault tolerance, availability, and storage elasticity and can be set up with cross-region read replicas. | DynamoDB is Amazon’s NoSQL database solution. Like other NoSQL databases, DynamoDB is commonly used to handle big data – large volumes of unstructured or semi-structured data.  Amazon DynamoDB is a fully managed NoSQL database service that supports document and key-value data models. Availability and fault tolerance are built in with automatic backup and restore, security, and multiregion, multimaster distribution along with in-memory caching.  DynamoDB is an ideal fit for internet-scale mobile, web, gaming, IoT, retail, media, and entertainment applications that require single-digit millisecond low latency data access and need to support petabytes of data. DynamoDB can automatically scale up/ down, and provides ACID transactions support. Your DBAs do not need to provision, patch, or manage servers. There’s no software to install, maintain, or operate. |

**Operating Features comparison between RDS (also Aurora) and DynamoDB**

|  |  |  |
| --- | --- | --- |
| Features | RDS (also Aurora as SQL) | DynamoDB (NoSQL) |
| Database type | Relational database management system (RDBMS) | Non-relational |
| Structure | Table with Columns and Rows | Collection of JavaScript Object Notation (JSON) documents, key-value, graph, or column |
| Schema | Predefined | Dynamic |
| Scale | Vertical | Horizontal |
| Language | SQL structured | Java, .NET, Node.js, and JavaScript in the browser |
| Performance | Suitable for online analytical processing (OLAP) | Built for online transaction processing (OLTP) at scale |
| Optimization | Optimized for storage | Optimized for read/write |

**Specific use-case for different AWS DBaaS offerings:**

If you only need a managed solution for database instances, choose RDS to use existing applications or architecture models with minimal changes.

If you are looking for a native High Availability (HA) solution for a read-intensive workload within an HA environment – Aurora is a good choice.

If you require a low-latency response to high-traffic queries and use AWS as the primary cloud infrastructure, strongly consider DynamoDB since it makes technical and economic sense. There are two key differentiators in DynamoDB where it [scales much better than traditional RDBMS](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-relational-modeling.html):

* Schema flexibility lets DynamoDB store complex hierarchical data within a single item.
* Composite key design lets it store related items close together on the same table.

If you use AWS as the primary cloud infrastructure for your cloud-native applications, DynamoDB is a good choice for workloads that fit the low-latency/ high-traffic use case.

Which database you use depends on the needs of your applications. Many organizations use a combination of different database types to handle different application workloads.

**A sample project demonstrates the use of one of the AWS DBaaS offerings and compare it with the other offerings**

#### Why should you choose a NoSQL Database like HBase, Couchbase or Cassandra over RDBMS?

1)Applications and databases need to work with Big Data

2)Big Data needs a flexible data model with a better database architecture

3)To process  Big Data, these databases need continuous application availability with modern transaction support.

NoSQL in Big Data Applications

1. HBase for Hadoop, a popular NoSQL database is used extensively by Facebook for its messaging infrastructure.
2. HBase is used by Twitter for generating data, storing, logging, and monitoring data around people search.
3. HBase is used by the discovery engine Stumble upon for data analytics and storage.
4. MongoDB is another[NoSQL Database](https://www.projectpro.io/NoSQL-Databases/27) used by CERN, a European Nuclear Research Organization for collecting data from the huge particle collider “Hadron Collider”.
5. LinkedIn, Orbitz, and Concur use the Couchbase NoSQL Database for various data processing and monitoring tasks.

The Database Landscape is flooded with increased data velocity, growing data variety, and exploding data volumes and only NoSQL databases like HBase, Cassandra, Couchbase can keep up with these requirements of  Big Data applications.

Limitations of SQL in handing Big data applications:

* It is not possible for SQL to process unpredictable and unstructured information. However, Big Data applications, demand for an occurrence-oriented database which is highly flexible and operates on a schema less data model.
* SQL Databases are vertically scalable – this means that they can only be scaled by enhancing the horse power of the implementation hardware, thereby making it a costly deal for processing large batches of data.
* IT enterprises need to increase the RAM, SSD, CPU, etc., on a single server in order to manage the increasing load on the RDBMS.
* With increasing size of the database or increasing number of users, Relational Database Management Systems using SQL suffer from serious performance bottlenecks -making real time unstructured data processing a hard row to hoe.
* With Relational Database Management Systems, built-in clustering is difficult due to the ACID properties of transactions.

1)Database Type created on AWS: RDS

AWS Database name : group3

Graphical user interface, text, application, email

Description automatically generated