DBaaS

Cloud Infrastructure Engineering

**Nanyang Technological University
& Skills Union - 2022/2023**

Course Content

- Quick Check-In

- Dive into the basics of DBaaS

- Dive into the basics of SQL and NoSQL Databases

- Explore the differences between the 2 Database types

- Explore creating database resources on AWS

Time	What	How or Why
7:15pm - 7:45pm	Part 1 - Presentation	Overview of DBaaS Compare NoSQL and SQL databases
7:45pm - 7:55pm	Break	
7:55pm - 8:15pm	Part 2 - Hands-on Activity	
8:15pm - 8:35pm	Practical Activity	
8:35pm - 8:45pm	Summary	Summary of lessons
8:45pm - 10:00pm	Assignment & Wrap Up	

Recap

-

Self Study Check-In

Q1) Give some examples of NoSQL Databases that you know of

Q2) Give some examples of SQL Databases that you know of

What is DBaaS?

Brief Intro to DBaaS

Database-as-a-service are **fully managed database services** provided by cloud providers like AWS and GCP.

Users need not worry about the underlying infrastructure, they can fully focus on utilizing the database service.
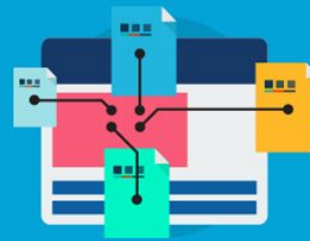
Overview

SQL

No SQL

Relational Data Model

Pros
> Easy to use and setup.
> Universal, compatible with many tools.
> Good at high-performance workloads.
> Good at structure data.

Cons
> Time consuming to understand and design the structure of the database.
> Can be difficult to scale.

Document Data Model

Pros
> No investment to design model.
> Rapid development cycles.
> In general faster than SQL.
> Runs well on the cloud.

Cons
> Unsuited for interconnected data.
> Technology still maturing.
> Can have slower response time.

Database Intro

SQL is a **decades-old method for accessing relational databases**, and most who work with databases are familiar with it.

As **unstructured data**, amounts of storage and processing power and types of analytics have changed over the years, however, we've seen different database technologies become available that are a better fit for newer types of use cases.
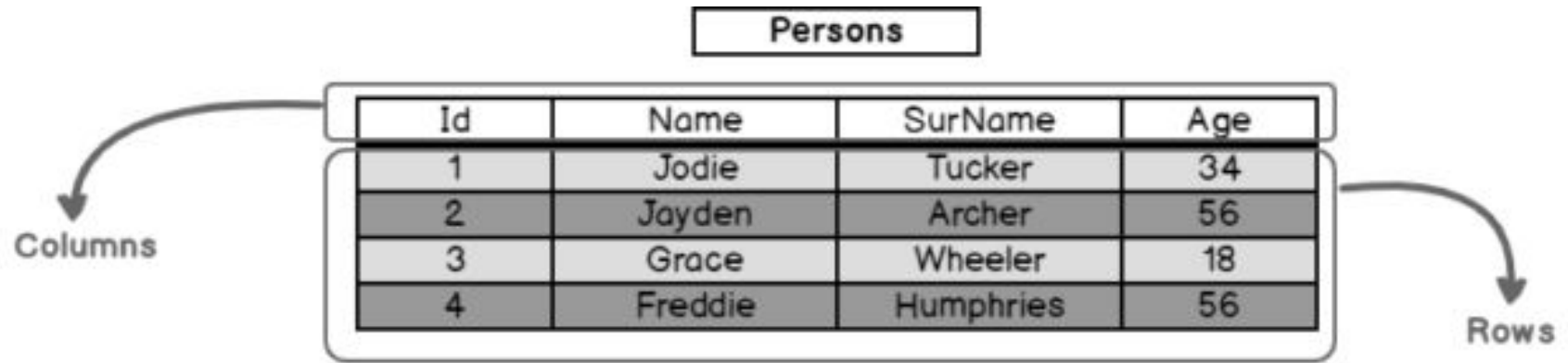
These databases are commonly called **NoSQL**.

SQL

Today, SQL is still widely used for querying relational databases, where data is **stored in rows and tables that are linked in various ways.**

One table record may link to one other or to many others, or many table records may be related to many records in another table.

These relational databases, which offer **fast data storage and recovery**, can **handle great amounts of data** and **complex SQL queries**.
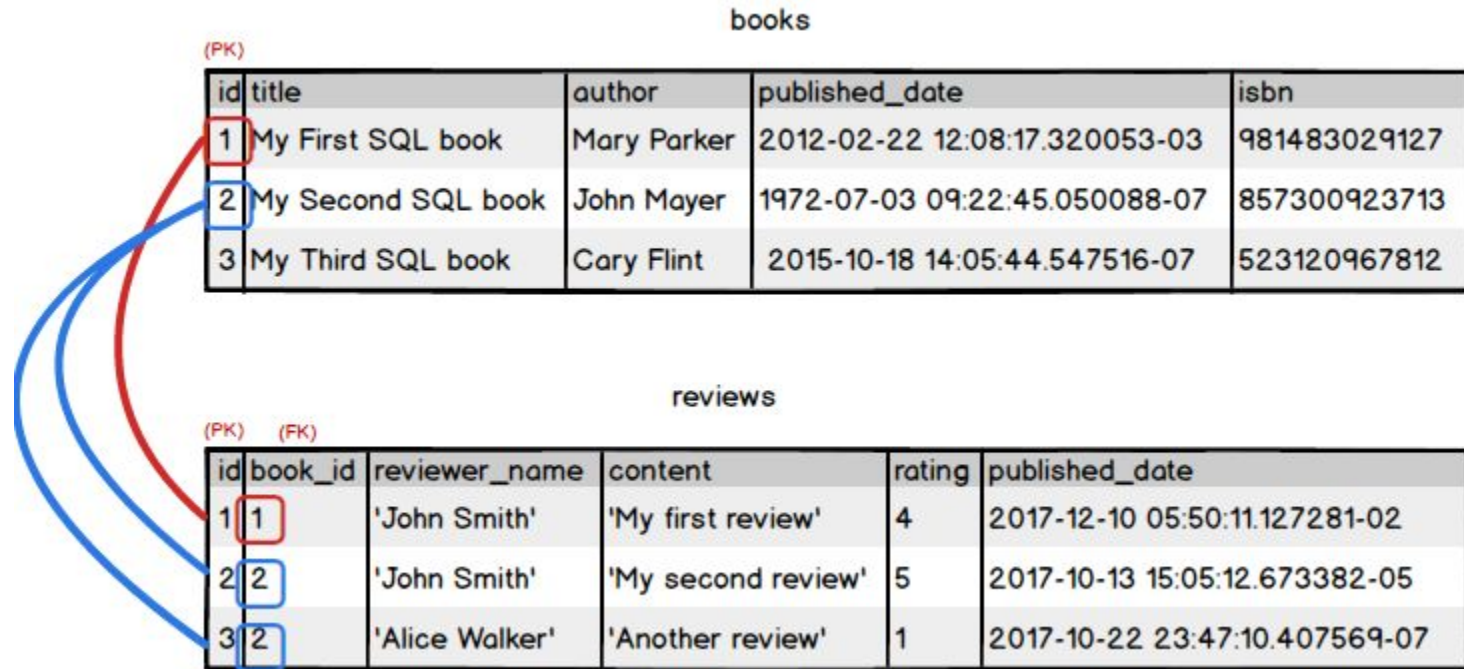
How SQL Works

How SQL Works - Multiple Tables

SQL - Structure

SQL database schema organizes data in **relational, tabular ways,** using tables with columns or attributes and rows of records.

Because SQL works with such a strictly predefined schema, it requires organizing and structuring data before starting with the SQL database.

SQL - Scalability

In general, SQL databases can scale vertically, meaning you can **increase the load on a server** by migrating to a larger server that a**dds more CPU, RAM or SSD** capability.

In AWS, you need not worry about this most of the time, depending on your DB choice.

SQL - ACID

Atomicity: All transactions must **succeed or fail completely** and cannot be left partially complete, even in the case of system failure.

Consistency: The database must **follow rules that validate** and prevent corruption at every step.

Isolation: Concurrent transactions **cannot affect each other**.

Durability: Transactions are final, and even system failure **cannot "roll back"** a complete transaction.

SQL - Huge Support

Because SQL databases have a long history now, they have huge communities, and many examples of their stable codebases online.

There are many experts available to support SQL and programming relational data.

- StackOverflow
- Individual Support Pages

SQL Options

SQL Activity

- What are some SQL databases that are available on AWS?
- What are some SQL databases that are available on GCP?

AWS SQL Options

Amazon Relational Database Service

Amazon Aurora

Amazon Redshift

AWS RDS Overview

RDS - Relational Database Service - is a **managed DB service for DB using SQL** as a query language.

It allows you to create databases in the cloud that are managed by AWS

- Postgres
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- Aurora (AWS Proprietary database)

AWS RDS Overview

AWS RDS Architecture
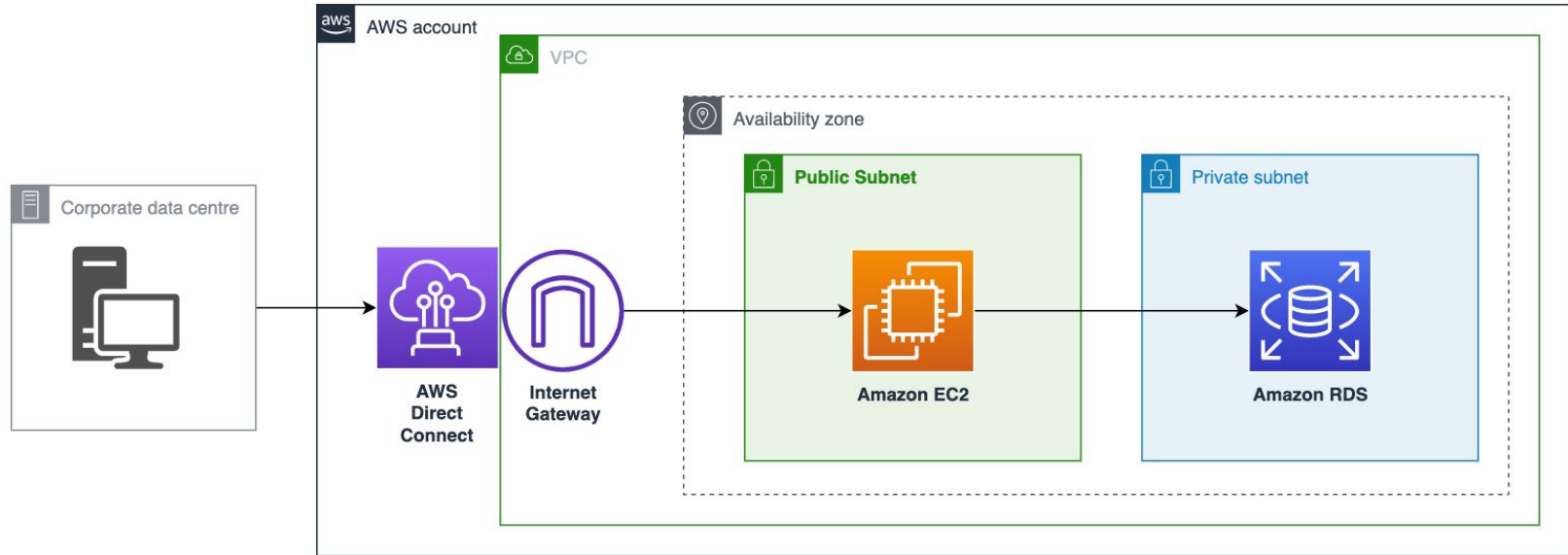
AWS RDS Multi-AZ

- For high availability and fault tolerance
- Offers synchronous replication to standby node
- Performs automatic failover in case of unplanned outage
- Failover time typically 60 - 120 seconds (RTO)
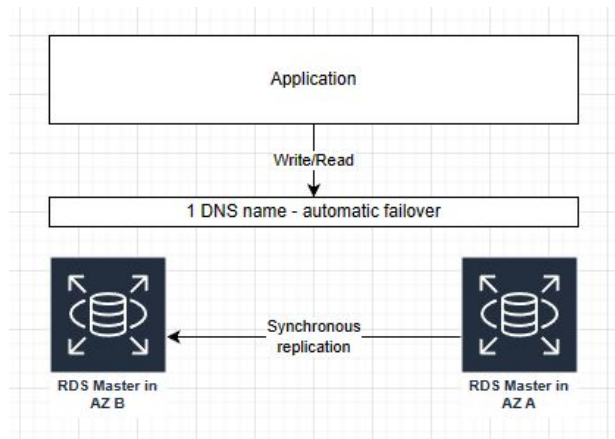
AWS RDS Read Replicas

- Used for scaling read heavy applications
- Cross - AZ and Cross-Region support
- Async replication -> Eventually consistent
- Created based on a snapshot of the master instance
- Can be promoted to primary in case of outage
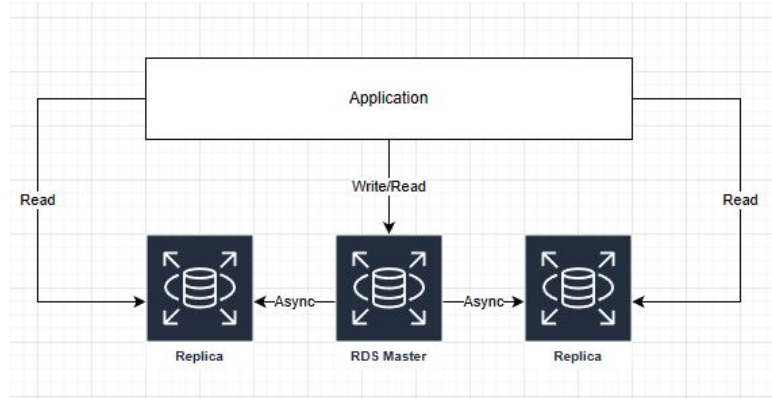
RDS Backup strategy

Backup	Snapshots
Automatically performed in backup window	Manual
Incremental	Full backup
Retention period 35 days	Retained as long as you want
Point-in-time Recovery (PITR)	Does not support PITR

RDS DR Planning

- RTO - How much downtime can there be?
- RPO - How much data loss can there be?
- Replication lag can affect recovery - How far behind your replica is to the master instance?

Feature	RTO	RPO	Cost	Scope
Automated backups	Good	Better	Low	Single Region
Manual snapshots	Better	Good	Medium	Cross-Region
Read replicas	Best	Best	High	Cross-Region

RDS Encryption

- Encryption in Transit - Use AWS Root CA and provide SSL certificate when connecting to DB in the connection string
- Encryption at Rest - AWS RDS supports AES-256 encryption and you can manage your keys using KMS to encryption both master and replicas

AWS Aurora DB

Aurora is a proprietary database from AWS

PostgreSQL and MySQL are both supported as Aurora DB

Aurora is "AWS cloud optimized" and claims **5x performance improvement** over MySQL on RDS, over **3x the performance** of Postgres on RDS

Aurora storage automatically grows in increments of 10GB, up to 64 TB.

Aurora costs more than RDS (20% more) – but it is more efficient

$ Not in the free tier $

AWS Aurora Architecture

NANYANG TECHNOLOGICAL UNIVERSITY | SINGAPORE
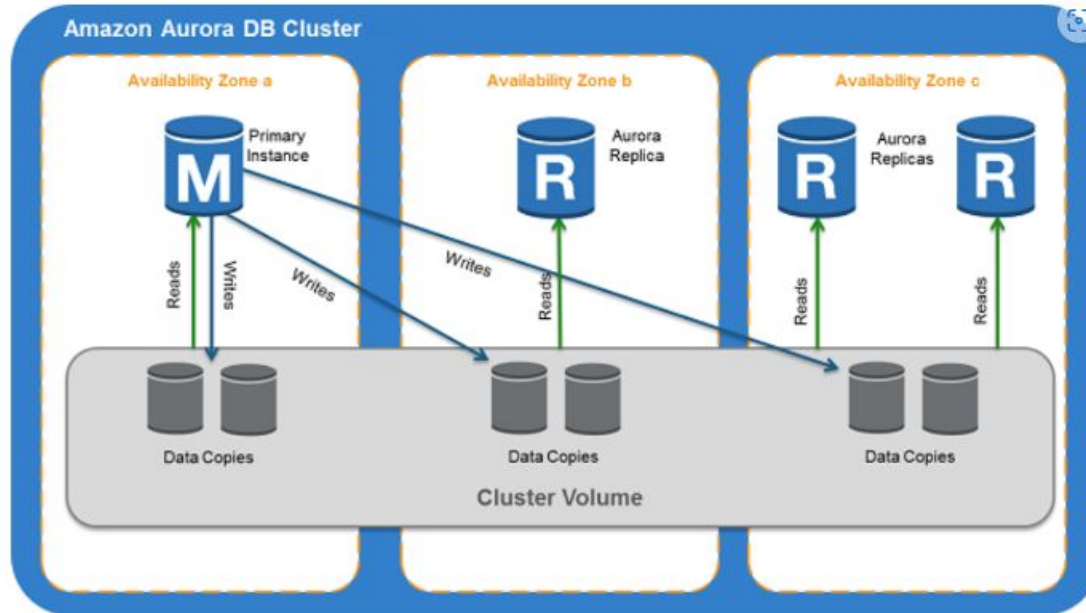
AWS Aurora Multi-Master

- 0 downtime failover
- Can write to both masters
- Synchronous replication
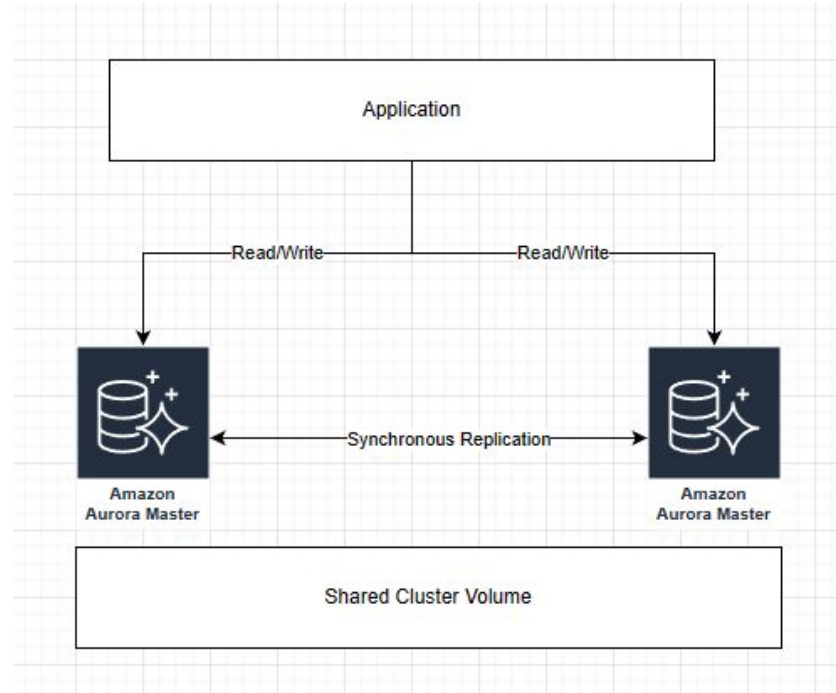
AWS Aurora Serverless

- Fully managed on-demand aurora
- Supports both MySQL and PostgreSQL
- Scales up/ down based on application use case
- Great for unpredictable or infrequent workloads
- No capacity planning needed

AWS Aurora DB - Read Replicas

You can scale the read workload of your database to reduce utilization on the main database

Create up to 15 Read Replicas

Data is only written to the main database and replicated asynchronously to Read Replicas

AWS Aurora DB - Read Replicas
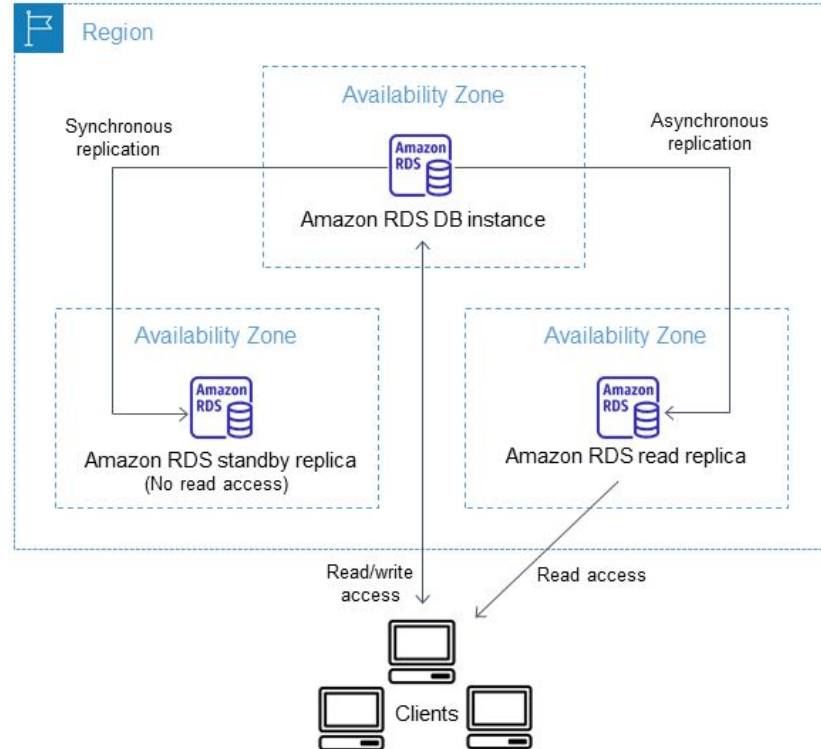
AWS Aurora DB - Multi-AZ

Ability to failover in case of an AZ outage (High Availability set up)

Data is only written to and read from the main database

Only 1 other AZ as failover

AWS Aurora DB - Multi-AZ
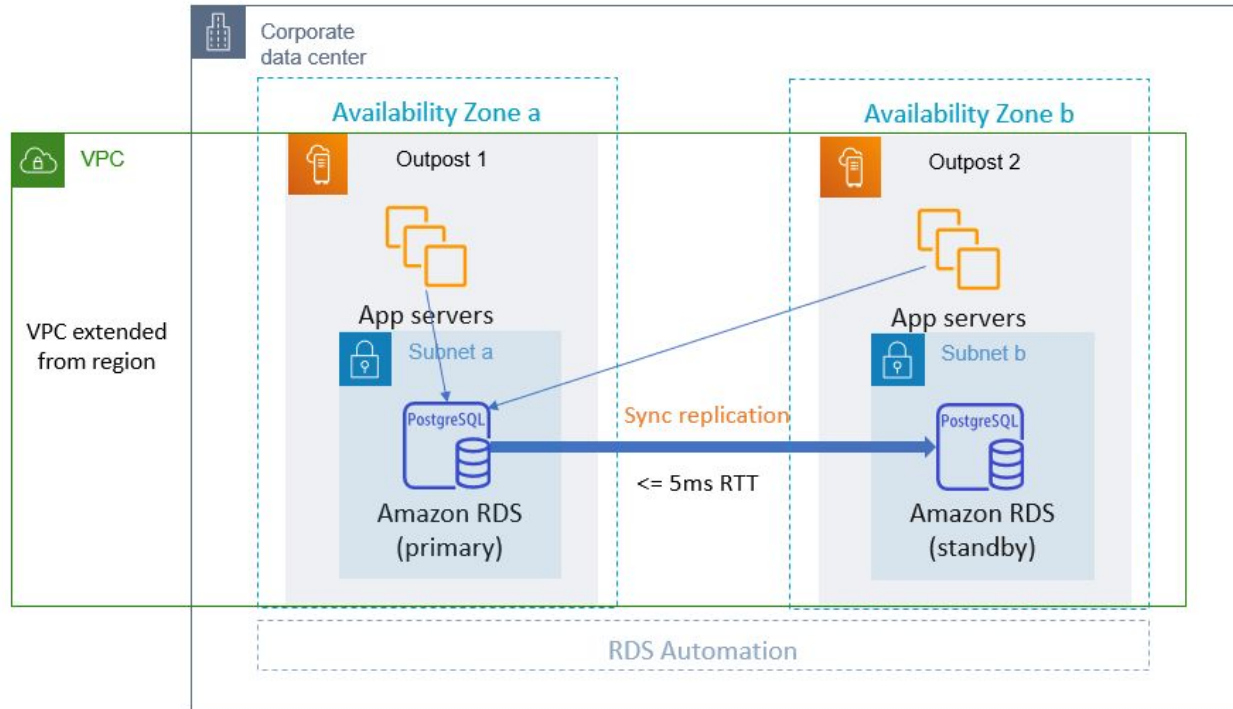
AWS Aurora DB - Multi-Region

Ability to provide **Disaster Recovery** in case of regional issues

Main database in one region will be the **main read/write database** whereas the backup databases in other region(s) will be a **read replica**

Need to consider the cost of this set-up as it can get very **expensive**

AWS Aurora DB - Multi-Region
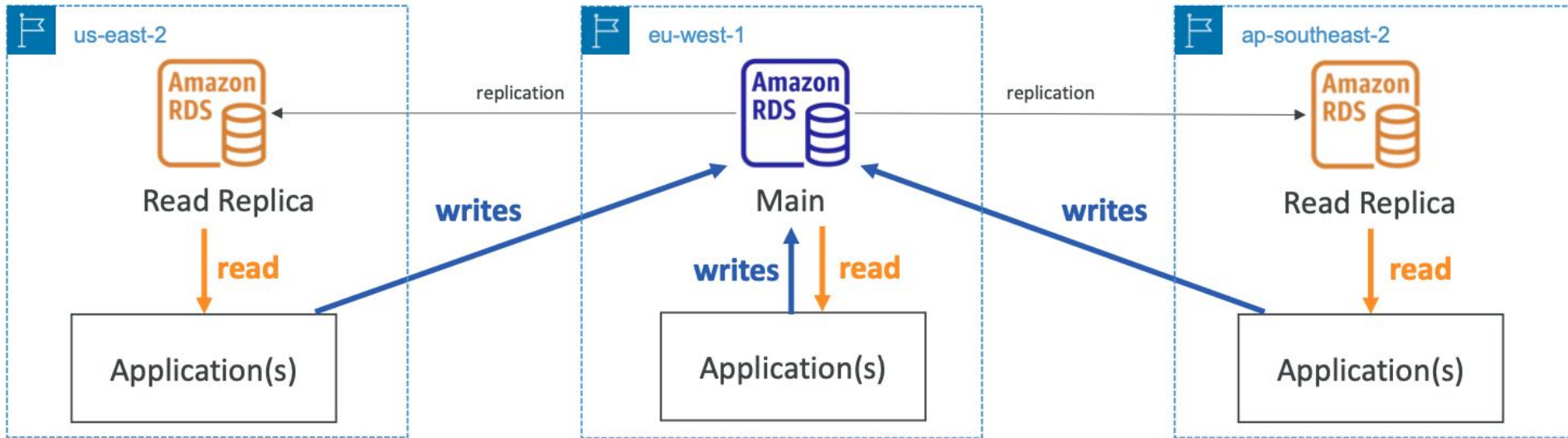
AWS SQL Activity
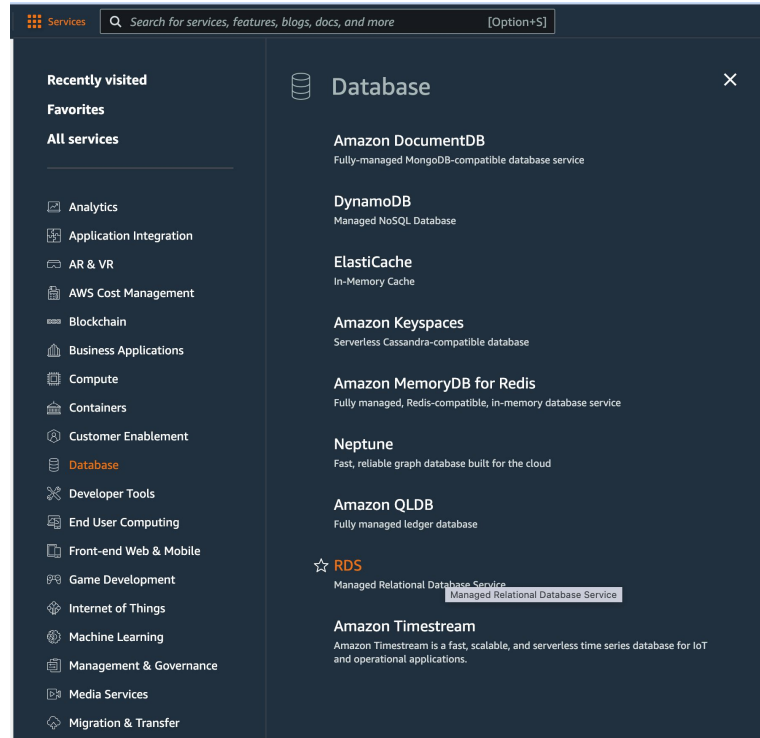
AWS SQL Activity

- Create an SQL database via console under Free Tier (RDS MYSQL)
- Explore your created database
- Drop the table and delete the database

NoSQL

NoSQL

Unlike SQL, NoSQL systems allow you to work with **different data structures** within a database.

Because they allow a dynamic schema for **unstructured data**, there's less need to pre-plan and pre-organize data, and it's easier to make modifications.

NoSQL databases allow you to **add new attributes and fields**, as well as use varied syntax across databases.

How NoSQL Works

```
{
    first_name: 'Paul',
    surname: 'Miller',
    cell: 447557505611,
    city: 'London',
    location: [45.123,47.232],
    Profession: ['banking', 'finance', 'trader'],
    cars: [
        { model: 'Bentley',
          year: 1973,
          value: 100000, ... },
        { model: 'Rolls Royce',
          year: 1965,
          value: 330000, ... }
    ]
}
```

Fields

String Number Geo-Coordinates → Typed field values

Fields can contain arrays

Fields can contain an array of sub-documents

How NoSQL Works

NANYANG TECHNOLOGICAL UNIVERSITY | SINGAPORE

NoSQL - Scalability

NoSQL databases **scale better horizontally**, which means one can add additional servers or nodes as needed to increase load.

NoSQL - Structure

- **Column-oriented,** where data is stored in cells grouped in a virtually unlimited number of columns rather than rows.
- **Key-value stores**, which use an associative array (also known as a dictionary or map) as their data model - **key-value pairs.**
- **Document stores**, which use documents to hold and encode data in standard formats, including XML, YAML, JSON and BSON.
- **Graph databases,** which represent data on a graph that shows how different sets of data relate to each other. Neo4j, RedisGraph (a graph module built into Redis) and OrientDB are examples of graph databases.

NoSQL - Structure

Key-Value
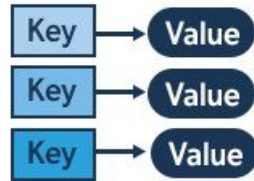
Column-Family
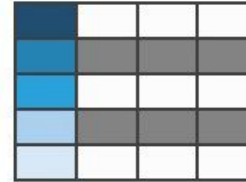
Graph
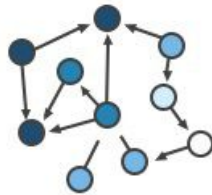
Document

NoSQL - CAP

Consistency: Every request receives either the most recent result or an error. MongoDB is an example of a strongly consistent system, whereas others such as Cassandra offer eventual consistency.

Availability: Every request has a non-error result.

Partition tolerance: Any delays or losses between nodes do not interrupt the system operation.

NoSQL Examples

NoSQL Activity

- What are some NoSQL databases that are available on AWS?
- What are some NoSQL databases that are available on GCP?

AWS NoSQL Examples

Amazon DynamoDB

Amazon Document DB (with MongoDB compatibility)

Amazon Neptune

AWS DynamoDB

Fully Managed **Highly available** with replication across 3 AZ

Scales to massive workloads, **distributed "serverless" database**

Millions of requests per seconds, trillions of row, 100s of TB of storage

Fast and consistent in performance

Single-digit millisecond latency – low latency retrieval

Integrated with IAM for security, authorization and administration

Low cost and auto scaling capabilities

Standard & Infrequent Access (IA) Table Class

AWS DynamoDB

AWS DynamoDB Consistency

- Eventual Consistency Read - If we read just after a write, we may not get the latest data
- Strong Consistency Read - If we read just after a write, we will get the correct data
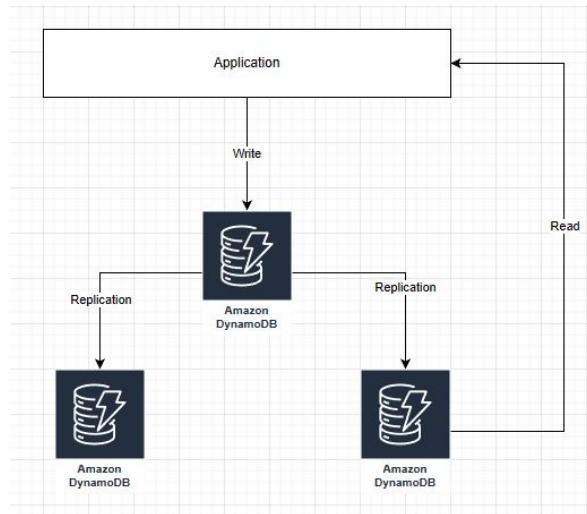
AWS DynamoDB Accelerator (DAX)

Fully Managed in-memory cache for DynamoDB

10x performance improvement – single- digit millisecond latency to microseconds latency – when accessing your DynamoDB tables

Secure, highly scalable & highly available

AWS DynamoDB Accelerator (DAX)
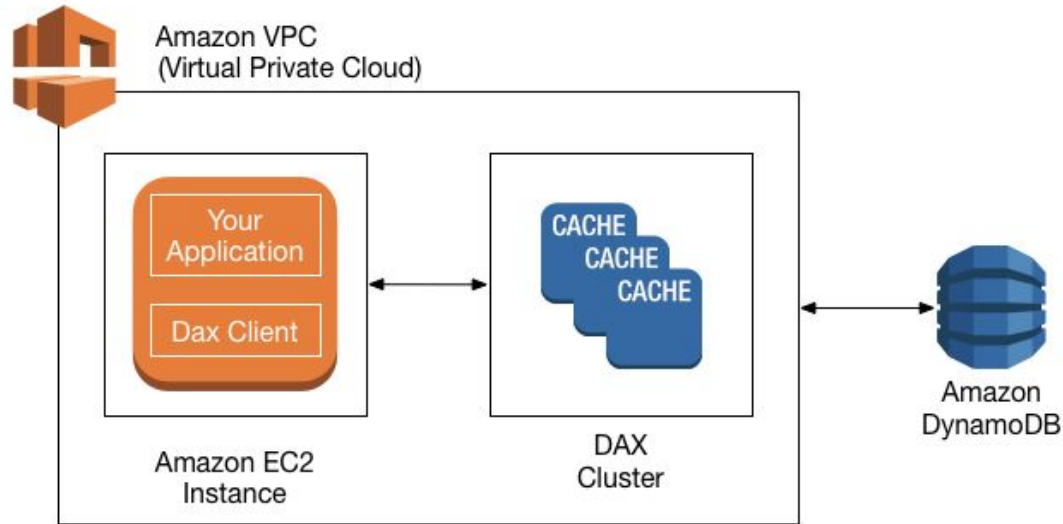
AWS DynamoDB Global Tables

Make a DynamoDB table accessible with **low latency in multiple-regions**

Active-Active replication (read/write to any AWS Region)

AWS DynamoDB Global Tables
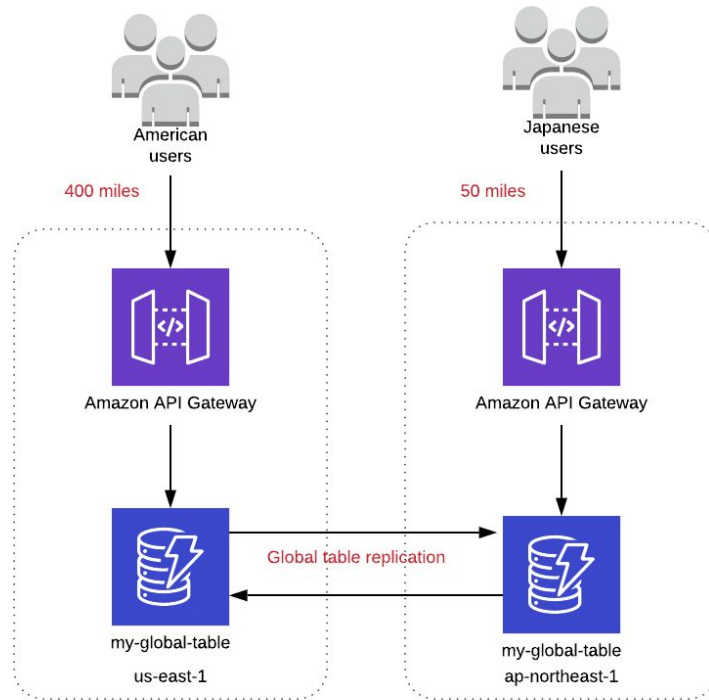
AWS DynamoDB Backups

- Can backup within same region as the table
- Can restore to same or different region
- Can be integrated with AWS Backup or periodic backups using Eventbridge / Cloudwatch with Lambda

AWS NoSQL Activity

Use case 1: Product catalog

Suppose that you want to store product information in DynamoDB. Each product has its own distinct attributes, so you need to store different information about each of these products.

You can create a `ProductCatalog` table, where each item is uniquely identified by a single, numeric attribute: `Id`.

Table name	Primary key
ProductCatalog	Partition key: `Id` (Number)

AWS NoSQL Activity

- Create a NoSQL database (DynamoDB) via console - refer to https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.CreateTables.html
- Explore the database
- Drop the database when done

Considerations - SQL

Relational data. A benefit of a relational database is that when one user updates a specific record, **every instance of the database automatically refreshes**, and that information is provided in **real-time**.

SQL and a relational database make it easy to handle a **great deal of information**, scale as necessary and allow flexible access to data.

It's also best for assessing **data integrity**. Since each piece of information is stored in a single place, there's no problem with former versions confusing the picture.

Considerations - NoSQL

Good when it's more important that the **availability of big data is fast**.

Ever-**changing requirements or messy, unstructured data**.

Scalability is important to the organization.

Activity

- What 3 words would you use to describe SQL?
- What 3 words would you use to describe NoSQL?

DBaaS

DBaaS

Allows users associated with database activities to **access and use a cloud database system without purchasing** it.

Usage is based on a **pay-as-you-go package,** similar to IaaS, PaaS and SaaS.

DBaaS consists of an **info manager element**, that controls all underlying info instances via API. This API is accessible to the user through a management console that the user might use to **manage and assemble the info** and even provision or deprovision info instances.

DBaaS

Database hosting options are **available for all database types**, including NoSQL, MySQL, and PostgreSQL. MongoDB Atlas is one example of a NoSQL DBaaS service that is easily scalable.

How Does DBaaS Work?

- Upload the data.
- Once the data has been uploaded, the DBaaS database engine itself operates in almost exactly the same way as an on-premises installation.
- The **management** of the database is **handled by the service**; you only need to worry about the data and minor configurations.
- The major difference is the **physical infrastructure** on which a cloud database runs, usually virtual machines in the respective cloud environment.

DBaaS vs Managing DB

Cloud database management is often **much simpler** than traditional on-premises equivalents; the **amount of back-end administration** required for DBaaS is much lower.

DBaaS Benefits

NANYANG TECHNOLOGICAL UNIVERSITY | SINGAPORE

Cost-Saving

Laying down infrastructure for database management is expensive; scaling it as needed is costly and often wasteful.

With DBaaS, your organization pays a **predictable periodic charge** based on the **resources you consume**—there's no need to purchase additional capacity to have on hand for hypothetical future needs.

Autoscaling

You can quickly and easily provision additional storage and computing capacity at run time if you need it, and you can scale down your database cluster during non-peak usage times to **save cost**.

Quick Deployment

With an on-premises database system, development teams typically need to request access through IT, a process that can **take days or weeks.**

In contrast, with DBaaS, developers can help themselves to database capabilities and **spin up and configure a database that's ready** to integrate with their application **in minutes.**

Data Security

Cloud database providers typically offer **enterprise grade security**, including features like default **encryption of data at rest and in-transit** and integrated identity and access management controls.

Some also meet specific **regulatory compliance standards**.

Reduced Risk

DBaaS offerings from major cloud providers typically include a **service-level agreement (SLA) guaranteeing a certain amount of uptime**.

In the unlikely event that your provider doesn't meet the requirements stipulated in the SLA, you'll be **compensated for any excess downtime** you experience.

Top Software Quality

The major cloud providers offer a wide variety of highly configurable DBaaS options—each preselected for quality, so you won't have to worry about the wading through hundreds of different databases.

Summary

Reduced Cost

Autoscaling

Quick Deployment Time

Data Security

Reduced Risk

Top Software Quality

Group Activity

You or your group have discussed a specific case or business. If you will implement MySQL or NoSQL in your business, let's answer the following questions based on your business/case

- Which will you choose to be implemented on your company?
- What is the main factor when you choose that answer?
- What is the 2nd option database that you will consider to be used?

Activity

Learner:

- Clean up AWS.
- Remove/delete/terminate all service/ resources that created.

Instructor

- Clean up AWS.
- Remove/delete/terminate all service/ resources that created.
- Check the AWS account after learner clean up.

What's Next?