

MODULE 6: Adding a Database Layer

Friday, October 10, 2025 2:10 PM

I. Scenario:

As a cloud architect, I should evaluate the available database system before choosing a suitable data management which can optimize performance. Moreover, securing data from threats is also important

II. Database Layer Consideration:

1. Scalability:

- How much throughput is needed?
- Will it scale?

2. Storage requirements:

Will the database need to be sized in gigabytes, terabytes, or petabytes of data?

3. Data characteristics:





- What is your data model?
- What are your data access patterns?
- Do you need low latency?

4. Durability:

- What level of data durability, availability, and recoverability is required?
- Do regulatory obligations apply?

II. Relational and Non-relational Database

Features	Relational Databases	Non-Relational Database
Structure	Tabular form of columns and rows	Variety of structure models (key-value pairs, document, or graph-based)
Schema	Strict schema rules	Flexible schemas
Benefits	Ease of use, data integrity, reduced data storage, and common language (SQL)	Flexibility, Scalability, and high-performance
Use Case	When migrating an on-premises relational workload or if your workload involves online transactional processing	When a caching layer is needed to improve read performance, when storing JSON documents, or when a single-digit millisecond data retrieval is needed
Optimization	Optimized for structured data stored in tables, supports complex one-time queries through joins	Optimized for fast access to structured, semi-structured, or unstructured data with high read and write throughput

Relational databases	Non-relational databases
<div> Amazon RDS</div>	<div><div> Amazon DynamoDB</div><div> Amazon Neptune</div><div> Amazon ElastiCache</div></div>
Managed database service that provides seven familiar database engines to choose from, including Amazon Aurora	Variety of services designed for databases such as key-value, graph, and in demand

Less responsibility with managed AWS database services

Tasks	Host Database on Premises	Host Database in Amazon EC2	Host Database in a Managed AWS Database Service
Power, HVAC, and Network	X		
Rack and Stack	X		
Server Maintenance	X		
OS Installation	X		
OS Patches	X		
Database Installation	X	X	
Database Patches	X	X	
Database Backups	X	X	
High Availability	X	X	
Scaling	X	X	
Application Optimization	X	X	X

Database Capacity Planning

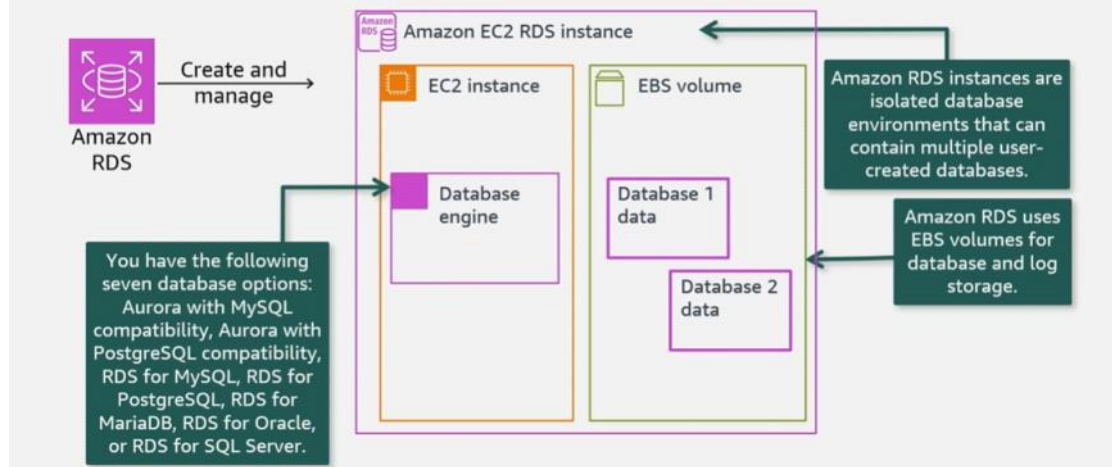
Designing with the end goal in mind

1. Analyze current storage capacity
2. Predict capacity requirements
3. Determine if horizontal scaling, vertical scaling, or a combination is needed.

III. Amazon RDS (Amazon Relational Database Service)

- Is managed relational database service to deploy and scale relational database
- Supports multiple database engines
- Uses Amazon Elastic Block Store (Amazon EBS) volumes for database and log storage.

Amazon RDS database architecture



- Aurora:
 - + Is a relational database management system (RDBMS) built for the cloud with full MySQL and PostgreSQL compatibility
 - + Is managed by Amazon RDS
 - + Provides high performance and availability at one-tenth of cost
 - + Delivers Multi-AZ deployments with Aurora Replicas
- Aurora Serverless:
 - + Is an on-demand auto scaling configuration for Aurora
 - + Provides hands-off capacity management
 - + Provides fine-grained scaling
 - + Is suitable with following:
 - * Variable workloads
 - * New applications
 - * Development and testing
 - * Capacity planning

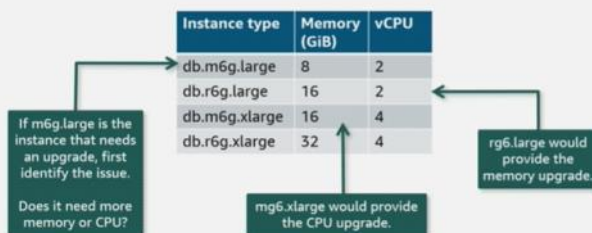
Amazon RDS use case: Banking transactions



Transaction ID	Date	Transaction Description	Transaction Type	Transaction Amount
0079834514	2023-11-05	Utility	Withdrawal	100.00
0079834513	2023-11-05	Employer name	Direct deposit	1000.00
0079834512	2023-11-04	Interest payment	Deposit	0.07

Amazon RDS EC2 instance types and sizing

- General purpose (T4g, T3, M6g, and M5)
- Memory-optimized (R6g, R5, X2g, and X1E)

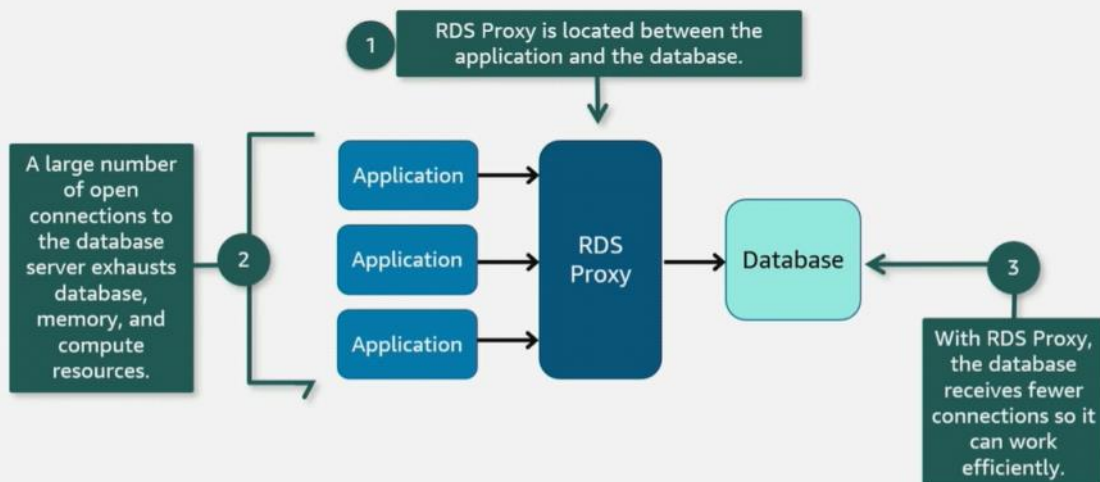


IV. Amazon RDS proxy connection management

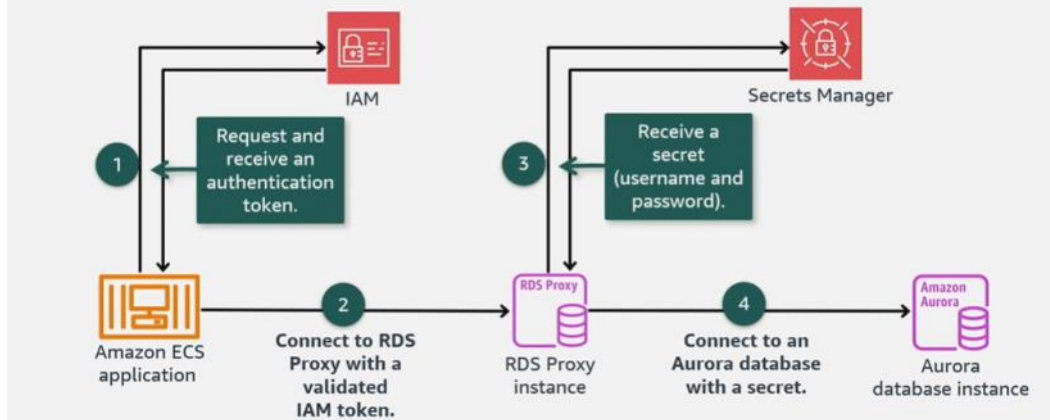
- Fully managed, highly available database proxy for Amazon RDS

More Scalable	More Resilient	More Secure
Pools and shares database connections for improved application scaling	Reduces database failover times for Aurora and Amazon RDS databases by up to 66 percent for Amazon RDS Multi-AZ databases	Enforces IAM authentication and stores credentials in AWS Secrets Manager

Connection pooling: Improved scalability



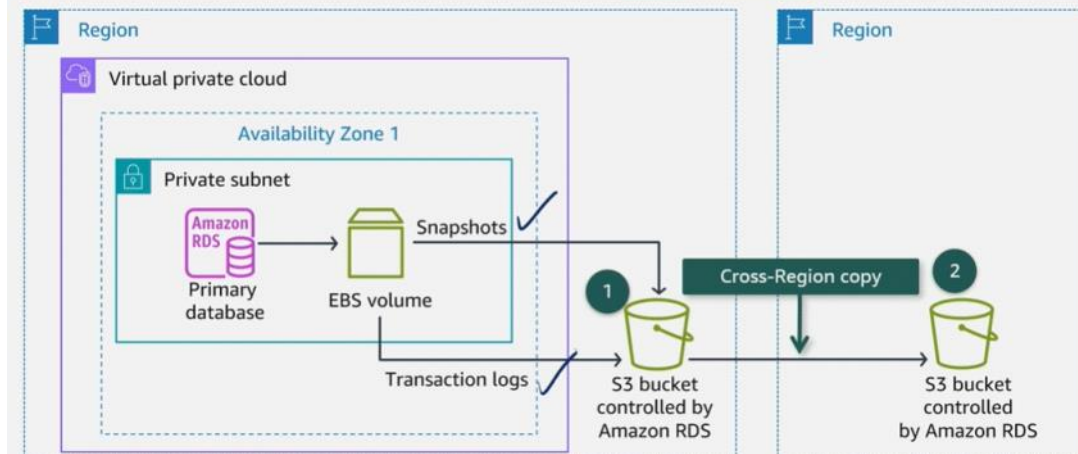
Streamlined authentication: Improved application security



- Backing up data in Amazon RDS

Features	Automated Backups	Database Snapshots
Use Case	Restore a database instance to a specific point in time	Back up a database instance in a known state, and then restore it to that specific state
Backup Frequency	Daily during your backup window (transaction logs are captured every 5 minutes)	User-initiated (as frequently as the user choose0
Retention Period	The default is 7 days but it can be set to up to 35 days. The backups can be automatically deleted after any retention period	Kept until the user explicitly deletes it
Sharing with Other AWS Accounts	Cannot be shared (needs to be copied to a manual snapshot first)	Can be shared (shared snapshots can be copied by other AWS accounts)

Amazon RDS cross-region backups



Amazon RDS encryption for backups

Amazon RDS can encrypt your RDS DB instances.

- Data at rest by using AWS KMS
- Data in transit by using SSL/TLS

Steps to back up an unencrypted database:

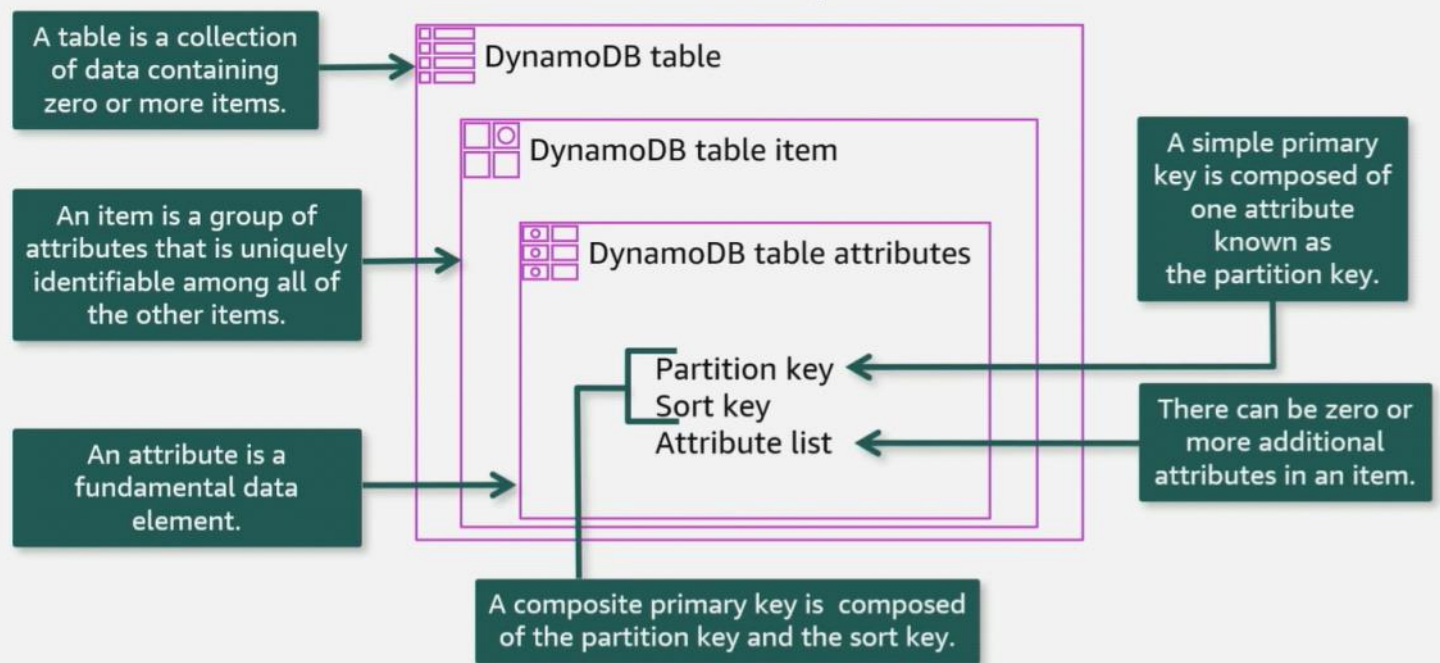


V. Amazon DynamoDB

- Is fully managed, serverless, NoSQL database
- Supports key-value and document data models
- Delivers millisecond performance and can automatically scale tables to adjust for capacity
- Is used for developing applications, mission-critical workloads that prioritize speed, scalability, and data durability
- DynamoDB Use Cases:

Develop software applications	Create media metadata stores	Scale gaming platforms
Build internet-scale applications that support user-content metadata and caches require high accuracy	Scale throughput and concurrency for media and entertainment workloads, such as real-time video streaming and interactive content	Build out your game platform with player data, session history, and leaderboards for millions of concurrent users

Amazon DynamoDB data structure



Amazon DynamoDB sample base table

Partition key = Device ID: 1	Sort key = Timestamp: 2023-11-20 15:42:00	Attribute = Temperature: 41.9	Attribute = Error Status: Low
Partition key = Device ID: 1	Sort key = Timestamp: 2023-11-20 15:42:30	Attribute = Temperature: 42	
Partition key = Device ID: 1	Sort key = Timestamp: 2023-11-20 15:43:00	Attribute = Temperature: 39	Attribute = Error Status: Low
Partition key = Device ID: 2	Sort key = Timestamp: 2023-11-20 15:42:00	Attribute = Temperature: 47	Attribute = Error Status: Low
Partition key = Device ID: 2	Sort key = Timestamp: 2023-11-20 15:42:30	Attribute = Temperature: 49	Attribute = Error Status: High
Partition key = Device ID: 2	Sort key = Timestamp: 2023-11-20 15:43:00	Attribute = Temperature: 46.9	

Alternate schema using a global secondary index

Base table	Partition key = Device ID: 1	Sort key = Timestamp: 2023-11-20 15:42:00	Attribute = Temperature: 41.9	Attribute = Error Status: Low
	Partition key = Device ID: 1	Sort key = Timestamp: 2023-11-20 15:42:30	Attribute = Temperature: 42	
	Partition key = Device ID: 2	Sort key = Timestamp: 2023-11-20 15:42:30	Attribute = Temperature: 49	Attribute = Error Status: High
	Partition key = Device ID: 2	Sort key = Timestamp: 2023-11-20 15:43:00	Attribute = Temperature: 46.9	
Global Secondary Index (GSI) [Eventual consistency]	Partition key = Temperature: 49	Sort key = Device ID: 2	Attribute = Timestamp: 2023-11-20 15:42:30	Attribute = Error Status: High

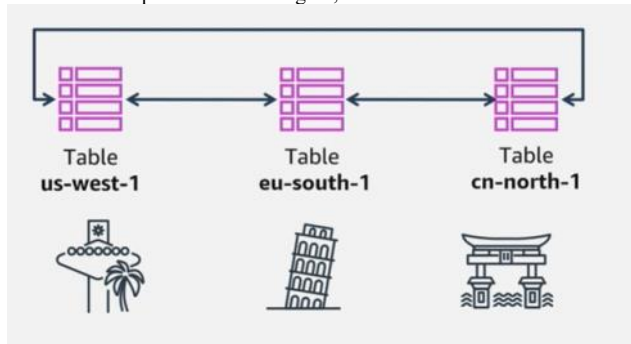
Query 1: For a given device id and timestamp, find the associated temperature and error status.

Query 2: For a given temperature and device id, find the associated error status and timestamp.

Alternate partition key as base table

Alternate sort key as base table

- Multi-region replication: Amazon DynamoDB global tables
- Global tables provide a multi-region, multi-active database for fast local read and write performance for global applications.



- DynamoDB security best practices

Preventative	Detective
<ul style="list-style-type: none"> • Use IAM roles to authenticate access. • Use IAM policies for DynamoDB base authorization. • Use IAM policy conditions for fine-grained access control. • Use a VPC endpoint and policies to access DynamoDB 	<ul style="list-style-type: none"> • Use AWS CloudTrail to monitor AWS managed AWS KMS key usage. • Monitor DynamoDB operations by using CloudTrail • Monitor DynamoDB configuration with AWS Config • Monitor DynamoDB compliance with AWS Config rules.

VI. Purpose-built databases

- The evolution of purpose-built databases

Opportunity	Database Evolution	AWS Examples
Improve on hierarchical databases' limited abilities to define relationships among data	Relational	Amazon RDS

Improve performance by separating read-heavy reporting from an application's transactional database	Data warehouse/OLAP	Amazon RedShift
Analyze the more varied types of data being generated in large amounts on the internet	Non-relational	DynamoDB
Take advantage of cloud computing's freedom to scale data stores up and down based on actual usage, and the ease of connecting different microservices to purpose-built data stores	Purpose build: <ul style="list-style-type: none"> • Document • Wide-column • In-memory • Graph • TimeSeries • Ledger 	Fully managed database services: <ul style="list-style-type: none"> * Amazon DocumentDB * Amazon Keyspaces * MemoryDB * Neptune * Timestream * Amazon QLDB

- Amazon RedShift:

- + Is a fully managed, cloud-based data warehousing service designed to handle petabyte-scale analytics workloads.
- + Achieves optimum query performance with columnar storage
- + Has an Amazon RedShift Serverless option
- + Is used for online analytics processing (OLAP)

- Matching a database to user's business need

- + Suitable workloads: Analyze user's workload requirements to see if they match the database's capabilities
- + Data model: Understand the characteristics of the data model that user's would need to use with the database
- + Features and Benefits; Familiarize yourself with key features and configuration options to optimize performance.
- + Common use cases: Review common use cases to find reference architectures and examples

- Amazon Keyspaces:

- + **Suitable workloads:**
 - * Fast querying capability of high volumes of data
 - * Scalability and consistent performance on heavy write loads
- + **Data Model:**
 - * Is a wide column data model
 - * Stores data in flexible columns, which permits data to evolve over time
 - * Partitions across distributed database systems
- + **Features and benefits:**
 - * Scalable, highly available, and managed Apache Cassandra-compatible database service.
- + **Common use cases:** Process data at high speeds for applications that require millisecond latency:
 - * Industrial equipment maintenance
 - * Trade monitoring
 - * Route optimization

- MemoryDB:

- + **Suitable workloads:**
 - * Latency-sensitive workloads
 - * High request rate
 - * High data throughput
 - * No data loss
- + **Data model:**
 - * Is an in-memory database service
 - * Relies primarily on memory for data storage
 - * Minimizes response time by eliminating the need to access disks
- + **Features and benefits:** Stores an entire dataset in memory and leverages a distributed transactional log to provide the following:
 - * In-memory speed and consistency
 - * Data durability and recoverability
- + **Common use cases:**
 - * Caching
 - * Game leaderboards
 - * Bank user transactions

- Neptune:

- + **Suitable workloads:**
 - * Find connections or paths in data
 - * Combine data with complex relationships across data silos
 - * Navigate highly connected datasets, and filter results based on certain variables
- + **Data models:**
 - * Is a graph data model
 - * Stores data and the relationships of that data to other data as equally important to the data itself.
 - * Quickly creates and navigates relationships between data
- + **Features and benefits:**
 - * Supports graph applications that require high throughput and low latency graph queries
 - * Creates and navigates data relations quickly
- + **Common use cases:**
 - * Recommendation engines
 - * Fraud detection
 - * Knowledge graphs

- * Drug discovery
- * Social networking

- Timestream:

+ Suitable workloads:

- * Identify patterns and trends over time
- * Determine value or performance over time
- * Rely on efficient data processing and analytics
- * Require ease of data management

+ Data model:

- * Is a timeseries data model
- * Is a sequence of data points recorded over a time interval for measuring events that change over time
- * Provides the ability to collect, store, and process data sequenced by time

+ Features and benefits:

- * Simplifies timeseries data access
- * Has built-in timeseries functions for quick analysis of timeseries data by using SQL

+ Common use cases:

- * Identify trends from data generated by IoT applications
- * Improve performance by analyzing web traffic data for user's applications in real time

- Amazon Quantum Ledger Database (Amazon QLDB):

+ Suitable workloads:

- * Maintain an accurate history of application data
- * Track the history of financial transactions

+ Data model:

- * Is a ledger database
- * Provides an immutable and verifiable history of all changes to application data

+ Features and benefits:

- * Provides a transparent, immutable, and cryptographically verifiable transaction log
- * Provides built-in data integrity
- * Provides a consistent event store

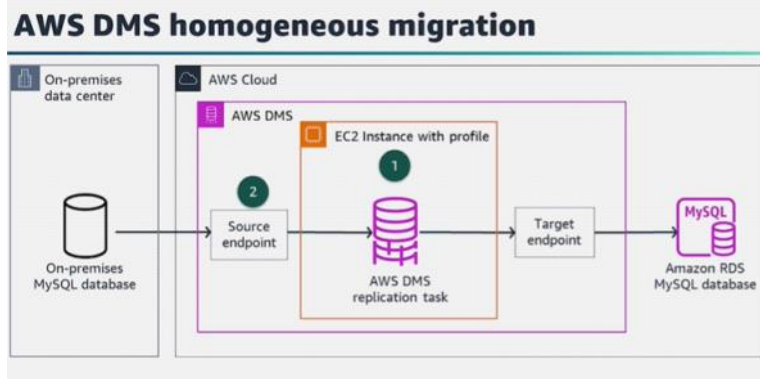
+ Common use cases:

- * Storing financial transactions
- * Maintaining claim history

VII. Migrating data into AWS databases

- AWS DMS (AWS Data Migration Service):

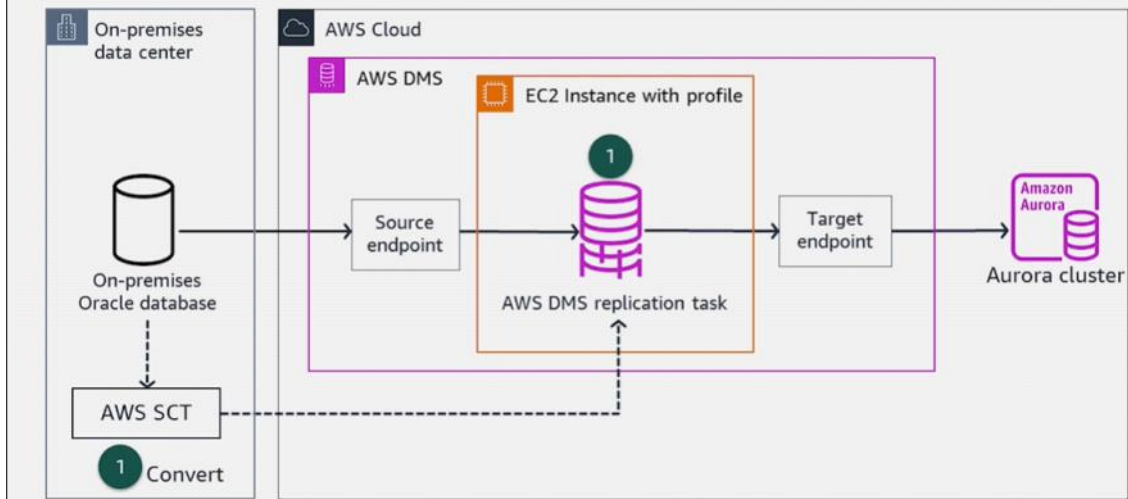
- + Is a managed migration and replication service
- + Helps move existing database and analytics workloads to and within AWS
- + Supports most widely used commercial and open source databases
- + Replicates data on demand or on a schedule to replicate changes from a service



- Tools for heterogeneous database migrations

Database Discovery Tool	Schema Conversion Tools	Schema Conversion Tools
AWS DMS Fleet Advisor	AWS Schema Conversion Tool (AWS SCT)	AWS DMS Schema Conversion
Automatically inventories and assesses on-premises database and analytics server fleet	Converts user's source and SQL code into an equivalent target schema and SQL code	Is a centrally managed service for schema assessment and conversion available within AWS DMS workflows

AWS DMS heterogeneous migration with AWS SCT



VIII. Applying Well-Architected Framework database pillar

- Performance Efficiency:

- + Evaluate how trade-offs impact customers and architecture efficiency
- + Use a data-driven approach for architectural choices
- + Factor cost into architectural decisions

- Cost Optimization:

- + Select the resource type, size, and number based on data

- Security:

- + Implement security key management
- + Enforce encryption at rest