AWS Database Services

A Beginner's Guide to AWS Managed Databases

Understanding Relational & Non-Relational Databases on AWS

AWS Training Series

What is a Database?

Simple Definition

A database is like a digital filing cabinet that stores and organizes information so you can easily find and use it later.

Two Main Types of Databases

📊 Relational Database

Stores data in tables with rows and columns, like a spreadsheet

📄 Non-Relational Database

Stores data in flexible formats like documents or key-value pairs

Relational Database Example

Students Table Example

Think of it like a simple spreadsheet:

ID	Name	Marks	Grade
1	John Smith	85	B
2	Sarah Johnson	92	A
3	Mike Brown	78	C

Key Features:

- Data is organized in rows and columns
- Each row represents one record (one student)
- Each column represents one attribute (name, marks, etc.)

- Uses SQL (Structured Query Language) to retrieve data

Non-Relational (NoSQL) Database Example

User Profile Document Example

Stores data as flexible documents (like JSON):

```
{
  "userId": "12345",
  "name": "John Smith",
  "email": "john@example.com",
  "preferences": {
    "theme": "dark",
    "notifications": true,
    "language": "English"
  },
  "hobbies": ["reading", "gaming", "cooking"],
  "address": {
    "street": "123 Main St",
    "city": "New York",
    "zipcode": "10001"
  }
}
```

Key Features:

- Flexible structure - each document can be different

- Can store complex nested data
- No fixed schema required
- Great for rapidly changing applications

Relational vs Non-Relational Comparison

Aspect	Relational Database	Non-Relational Database
Structure	Fixed tables with rows & columns	Flexible documents, key-value, etc.
Schema	Fixed schema (must define structure first)	Dynamic schema (can change anytime)
Query Language	SQL (Structured Query Language)	Various APIs and query methods
Best For	Complex relationships, transactions	Rapid development, flexible data
Examples	MySQL, PostgreSQL, Oracle	MongoDB, DynamoDB, Redis
Use Cases	Banking, inventory, accounting	Social media, gaming, IoT

AWS Database Services Overview

Why Use AWS Managed Databases?

- AWS handles maintenance, backups, and updates
- Built-in security and monitoring
- Easy to scale up or down
- Pay only for what you use
- High availability and disaster recovery

AWS Database Categories

🗃️ Relational Databases

- Amazon RDS
- Amazon Aurora
- Amazon Redshift

📄 Non-Relational Databases

- Amazon DynamoDB
- Amazon ElastiCache

Amazon RDS (Relational Database Service)

What is Amazon RDS?

A managed service that makes it easy to set up and run relational databases in the cloud.

Key Features:

- Supports multiple database engines (MySQL, PostgreSQL, Oracle, SQL Server)
- Automated backups and software patching
- Easy scaling of compute and storage
- Multi-AZ deployment for high availability
- Read replicas for improved performance

Real-World Use Case:

E-commerce Website: An online store uses RDS to store customer information, product catalogs, and order history. The structured data and ACID transactions ensure data consistency for financial transactions.

When to Use RDS:

- Need ACID transactions (banking, e-commerce)

- Complex queries with joins
- Existing applications using SQL
- Need strong data consistency

```
Client Application
        ↓
    Load Balancer
        ↓
    Amazon RDS
   (Multi-AZ Setup)
   Primary ←→ Standby
        ↓
   Read Replicas
```

Amazon Aurora

What is Amazon Aurora?

A high-performance, cloud-native relational database that's compatible with MySQL and PostgreSQL but much faster.

Key Features:

- 5x faster than MySQL, 3x faster than PostgreSQL
- Serverless option available (Aurora Serverless)
- Automatic scaling up to 128TB
- 6 copies of data across 3 Availability Zones
- Continuous backup to Amazon S3

Real-World Use Case:

Gaming Application: A mobile game uses Aurora Serverless to handle player data and leaderboards. During peak gaming hours, Aurora automatically scales up, and scales down during quiet periods, saving costs.

When to Use Aurora:

- Need high performance and availability
- Variable or unpredictable workloads (use Serverless)
- Want MySQL/PostgreSQL compatibility with better performance

- Global applications (Aurora Global Database)

Deployment Models:

- **Aurora Provisioned:** Fixed capacity, predictable workloads
- **Aurora Serverless:** Auto-scaling, pay per use

```
Client Application
        ↓
    Aurora Cluster
   Writer Instance
        ↓
   Shared Storage
   (Auto-scaling)
        ↓
   Reader Instances
   (Auto-scaling)
```

Amazon Redshift

What is Amazon Redshift?

A fast, fully managed data warehouse service designed for analyzing large amounts of data using SQL.

Data Warehouse vs Regular Database:

A data warehouse is optimized for reading and analyzing large amounts of historical data, while regular databases are optimized for daily transactions.

Key Features:

- Columnar storage (stores data by columns, not rows)
- Massively parallel processing (MPP)
- Compression and fast query performance
- Serverless option available
- Integrates with BI tools like Tableau, Power BI

Real-World Use Case:

Retail Analytics: A retail chain uses Redshift to analyze 5 years of sales data across 1000 stores to identify trends, seasonal patterns, and optimize inventory management.

When to Use Redshift:

- Need to analyze large datasets (TB to PB)
- Business intelligence and reporting
- Data warehousing and analytics
- Complex analytical queries

Deployment Models:

- **Redshift Provisioned:** Fixed clusters for consistent workloads
- **Redshift Serverless:** Pay per query, no cluster management

```
BI Tools / Analytics
        ↓
    Amazon Redshift
   (Data Warehouse)
        ↓
   Columnar Storage
   Node 1 | Node 2 | Node 3
        ↓
   Data Sources (S3, RDS, etc.)
```

Amazon DynamoDB

What is Amazon DynamoDB?

A fast, fully managed NoSQL database service that provides single-digit millisecond performance at any scale.

Key Features:

- Serverless - no servers to manage
- Automatic scaling based on demand
- Built-in security and backup
- Global tables for multi-region replication
- DynamoDB Streams for real-time data processing

Real-World Use Case:

Mobile App Backend: A social media app uses DynamoDB to store user profiles, posts, and comments. It can handle millions of users with consistent fast response times, even during viral content spikes.

When to Use DynamoDB:

- Need predictable, fast performance
- Serverless applications
- Mobile and web applications

- IoT applications with high write volumes
- Gaming leaderboards and session data

Advantages of AWS Managed Version:

- Zero administration - no servers to patch or maintain
- Automatic scaling - handles traffic spikes automatically
- Built-in monitoring with CloudWatch
- Point-in-time recovery and on-demand backups

```
Mobile/Web App
          ↓
     API Gateway
          ↓
     AWS Lambda
          ↓
     DynamoDB
    (Auto-scaling)
    Partition 1 | Partition 2 | Partition 3
```

Amazon ElastiCache

What is Amazon ElastiCache?

A fully managed in-memory caching service that improves application performance by storing frequently accessed data in memory.

Two Engine Options:

Redis

- Advanced data structures
- Persistence options
- Pub/Sub messaging
- Clustering support

Memcached

- Simple key-value store
- Multi-threaded
- No persistence
- Simpler to use

Real-World Use Case:

News Website: A news website uses ElastiCache Redis to cache popular articles and user sessions. Instead of querying the database every time, frequently read articles are served from memory, reducing page load time from 2 seconds to 200ms.

When to Use ElastiCache:

- Need to speed up database queries
- Store user sessions
- Cache frequently accessed data
- Real-time analytics and leaderboards
- Reduce database load

```
Web Application
          ↓
     Check Cache First
          ↓
     ElastiCache ⟷ Primary Database
    (Redis/Memcached)      (RDS/Aurora)
          ↓
     Fast Response (Cache Hit)
     or Query Database (Cache Miss)
```

Choosing the Right AWS Database Service

Decision Framework

Ask yourself these simple questions to choose the right database:

Your Need	Recommended Service	Why?
Traditional web app with structured data	Amazon RDS	Familiar SQL, ACID transactions
High-performance app, variable workload	Amazon Aurora	Better performance, serverless option
Business intelligence and analytics	Amazon Redshift	Optimized for analytical queries
Mobile/web app, flexible data	Amazon DynamoDB	NoSQL flexibility, serverless
Speed up existing database	Amazon ElastiCache	In-memory caching

Pro Tip:

You can use multiple database services together! For example: RDS for main data + ElastiCache for speed + Redshift for analytics.

Summary

🗃️ Relational Databases

- **RDS:** Managed traditional databases
- **Aurora:** High-performance, cloud-native
- **Redshift:** Data warehouse for analytics

📄 Non-Relational Databases

- **DynamoDB:** Fast NoSQL, serverless
- **ElastiCache:** In-memory caching

Key Benefits of AWS Managed Databases:

- ✅ No server management required
- ✅ Automatic backups and updates
- ✅ Built-in security and monitoring
- ✅ Easy scaling up or down
- ✅ Pay only for what you use
- ✅ High availability and disaster recovery

Next Steps:

- Try AWS Free Tier to experiment with these services
- Start with RDS or DynamoDB for your first project
- Use AWS Database Migration Service to move existing databases