Episode-12 | Databases - SQL & NoSQL

Q: what is a database ?

A:

> In computing, a **database** is an organized collection of data or a type of data store based on the use of a **database management system** (**DBMS**), the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a **database system**. Often the term "database" is also used loosely to refer to any of the DBMS, the database system or an application associated with the database.

Types of databases

1. **Relational DB** - MySQL, PostgreSQL:

Relational databases like MySQL and PostgreSQL use structured tables with predefined schemas, making them ideal for handling complex queries and transactions. They ensure data integrity through ACID properties and are widely used for applications requiring robust, relational data models.

2. **NoSQL DB** - MongoDB:

   MongoDB is a NoSQL database that stores data in flexible, JSON-like documents, allowing for dynamic schemas. It's highly scalable and ideal for handling large volumes of unstructured or semi-structured data, making it popular for modern web applications.

3. **In-memory DB** - Redis:

   Redis is an in-memory database known for its high-speed data processing capabilities. It supports various data structures like strings, hashes, and lists, making it suitable for caching, real-time analytics, and message brokering.

4. **Distributed SQL DB** - CockroachDB:

   CockroachDB is a distributed SQL database designed to scale horizontally across multiple nodes while providing strong consistency and ACID transactions. It's ideal for applications requiring high availability and resilience across different geographic locations.

5. **Time Series DB** - InfluxDB:

   InfluxDB is a time series database optimized for handling high write and query loads, particularly for time-stamped data. It's commonly used for monitoring, real-time analytics, and IoT applications where time-based data is crucial.

6. **OO DB** - db4o:

   db4o is an object-oriented database that stores data as objects, closely aligning with object-oriented programming languages. It simplifies development by allowing direct storage and retrieval of objects without the need for conversion to relational tables.

7. **Graph DB** - Neo4j:

   Neo4j is a graph database that excels at handling complex relationships between data entities. It uses a graph structure with nodes, relationships, and

properties, making it ideal for applications like social networks, recommendation engines, and fraud detection.

8. **Hierarchical DB** - IBM IMS:

   IBM IMS is a hierarchical database that organizes data in a tree-like structure with parent-child relationships. It's used primarily in legacy systems for high-performance transaction processing and is known for its reliability in handling large-scale, mission-critical applications.

9. **Network DB** - IDMS:

   IDMS (Integrated Database Management System) is a network database that represents data using a graph of record types and set relationships. It allows more complex relationships than hierarchical databases and is often used in legacy systems requiring high performance.

10. **Cloud DB** - Amazon RDS:

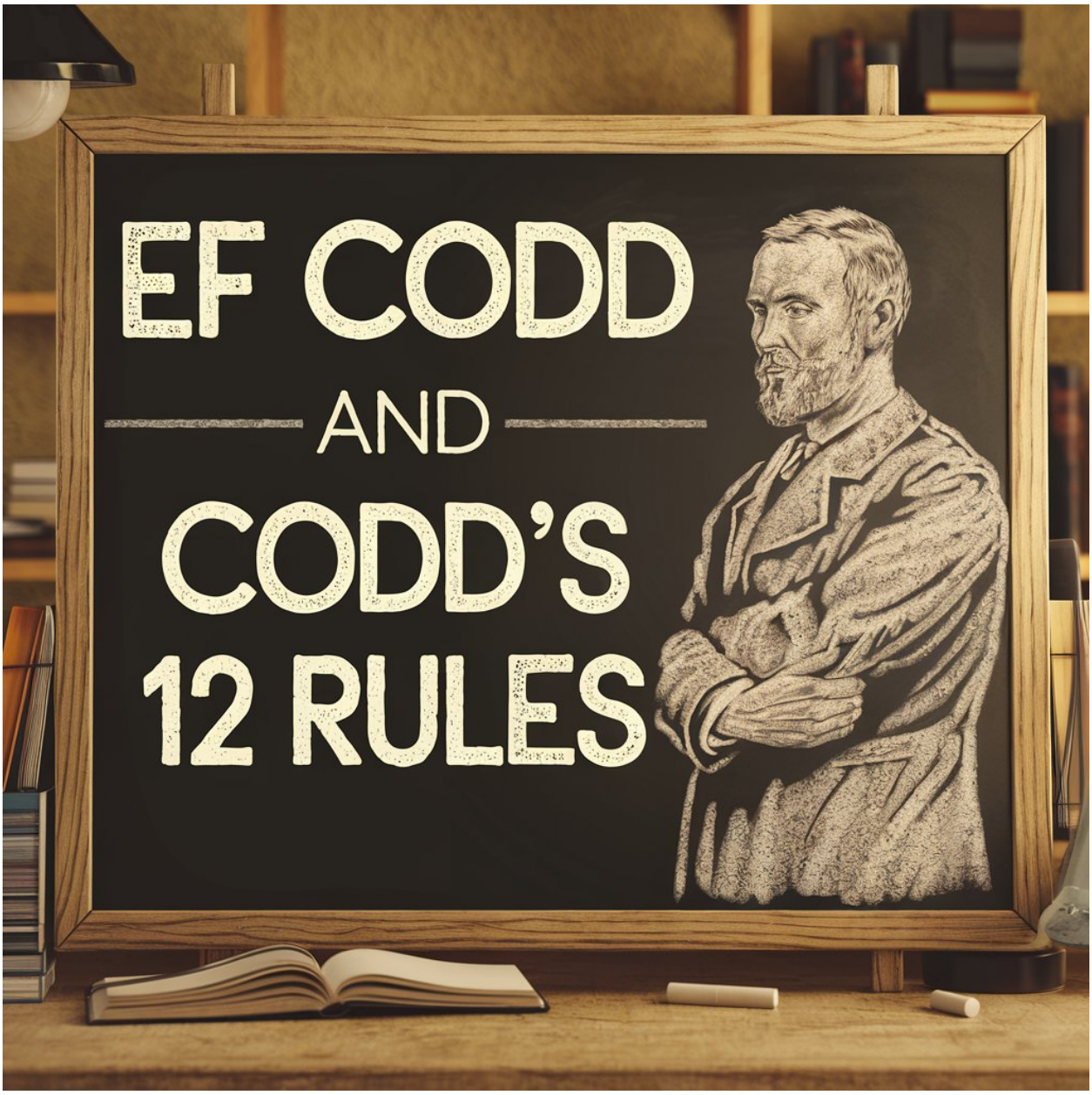
    Amazon RDS (Relational Database Service) is a managed cloud database service that supports multiple relational database engines, including MySQL, PostgreSQL, and Oracle. It automates tasks like backups, patching, and scaling, making it easy to deploy and manage databases in the cloud.

Most commonly used databases are :

1. Relational DB
2. NoSQL DB

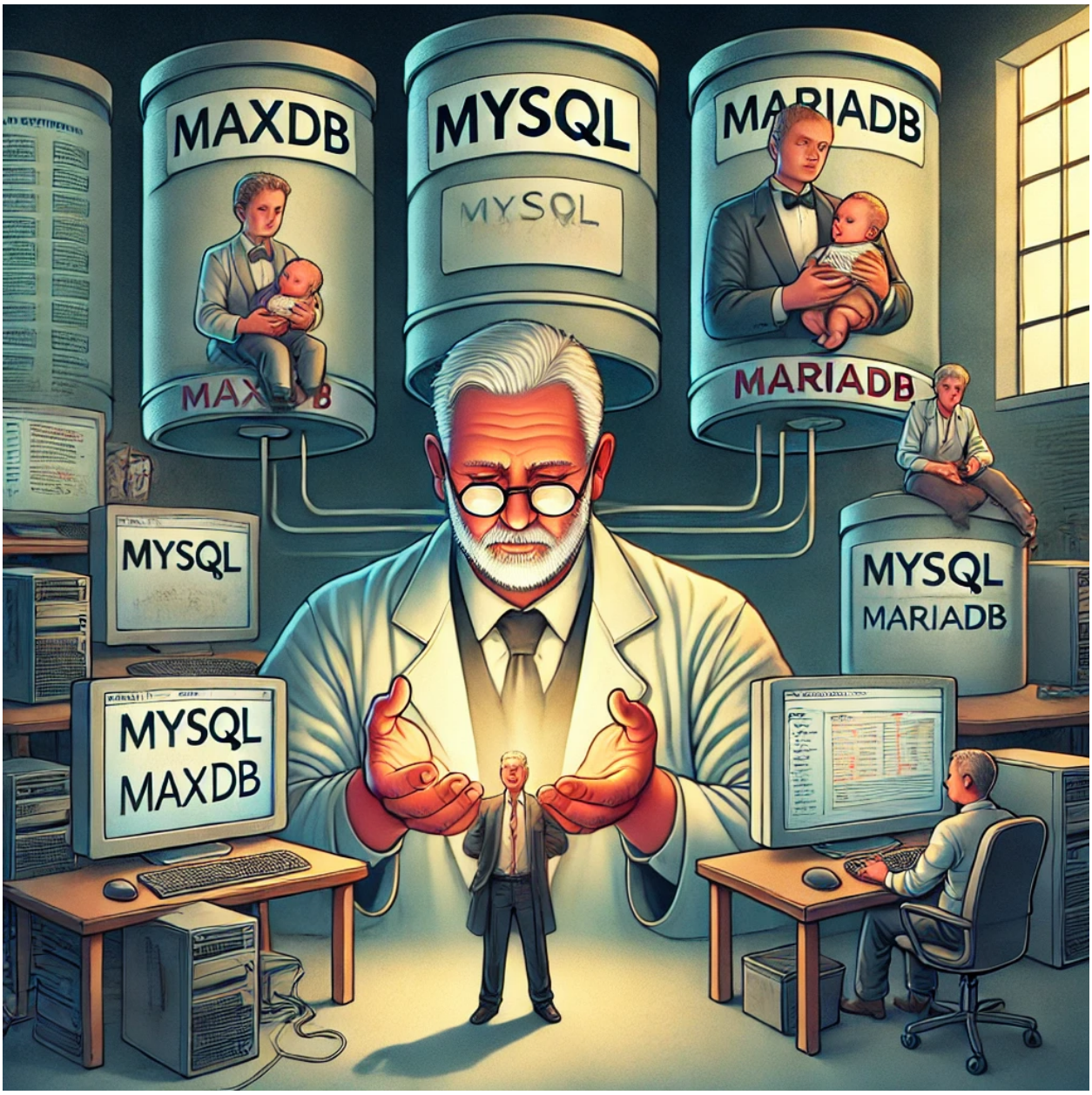
RDBMS (MySQL, PostgreSQL)

EF Codd and Codd’s 12 Rules



EF Codd was a scientist who introduced a concept known as Codd's 12 Rules. Interestingly, the rules are numbered from 0 to 12, making them a total of 13 rules. These rules were designed to determine whether a database qualifies as a relational database, which means it must have different tables and relationships between those tables.

The Story of MySQL



MySQL was developed by a scientist named Michael Widenius. The name "MySQL" comes from his first daughter's name, My. Michael also named another database after his second daughter, Max, calling it MaxDB, and he named a fork of MySQL after his third daughter, Maria, resulting in MariaDB. MySQL was eventually acquired by Sun Microsystems, which was later acquired by Oracle. Currently, MySQL is managed by Oracle. After the acquisition, Michael and others created a fork of MySQL, known as MariaDB.

The Story of PostgreSQL

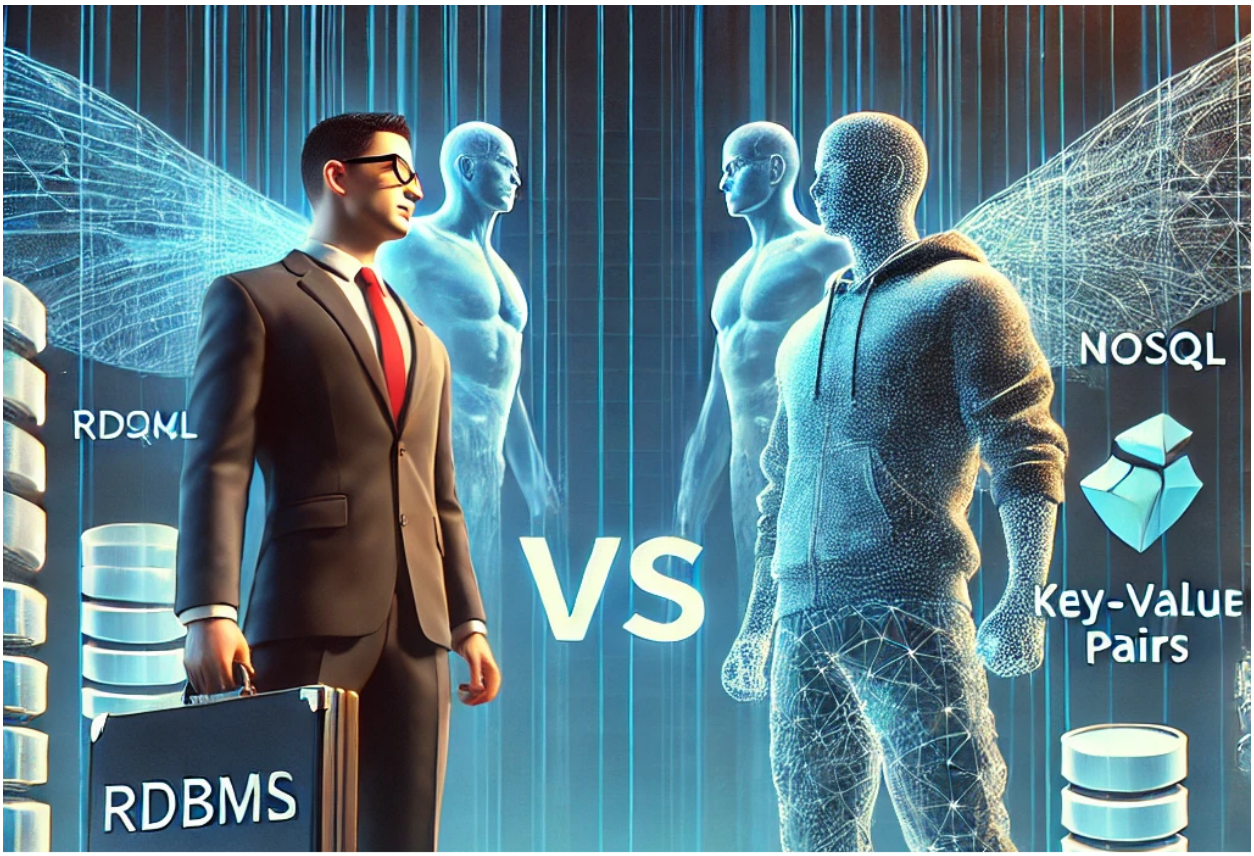
PostgreSQL was created by Michael Stonebraker, who was working on a project called Ingres at the University of California. He later began developing a project called Post Ingres, which eventually evolved into PostgreSQL because it utilized SQL (Structured Query Language) in the project. SQL is essential for interacting with relational databases, as it provides a structured way to query and manage data.

NoSQL and MongoDB

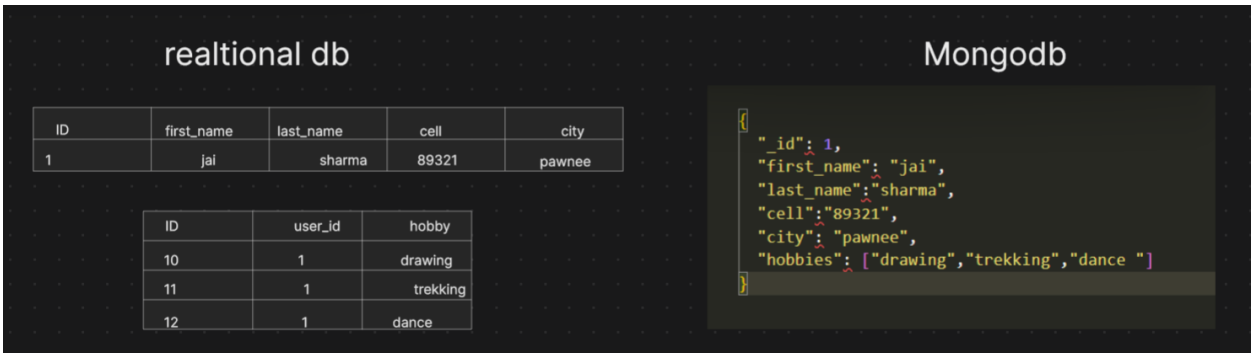
NoSQL databases can be classified into four main types:

1. Document Databases
2. Key-Value Databases
3. Graph Databases
4. Wide-Column Databases
5. Multi-Model Databases

MongoDB, a popular NoSQL database, was created in 2009 around the same time as Node.js. It’s a coincidence that both have gained popularity together, as MongoDB works exceptionally well with JSON and JavaScript objects. MongoDB was developed by a company named 10gen, and the name "MongoDB" comes from the word "humongous," reflecting its ability to handle massive amounts of data. The company later renamed itself MongoDB Inc., which continues to manage MongoDB today. MongoDB is known for significantly increasing developer productivity.



RDBMS vs NoSQL (Document)



RDBMS (Relational Database Management System)

- **Structure**: RDBMS uses a table-based structure, organizing data into rows and columns, similar to a spreadsheet or Excel sheet. Each table represents a different entity, and the columns represent attributes of the entity.
- **Relationships**: In RDBMS, relationships between tables are established using foreign keys. For example, if you want to store user hobbies, you would create separate tables for users and hobbies and then map them using a user ID. This design often requires the use of joins to retrieve related data.
- **Normalization**: Data normalization is a key concept in RDBMS. It involves organizing data to minimize redundancy and improve data integrity. This often requires multiple related tables and complex join operations to assemble a complete dataset.

NoSQL (Document Database)

- **Structure**: NoSQL document databases, such as MongoDB, use a flexible, schema-less structure. Data is stored in documents, which are similar to JSON objects. Each document is a collection of key-value pairs and can have nested structures.
- **Collections**: Instead of tables, MongoDB uses collections. A collection is a group of documents, analogous to a table in RDBMS. Each document in a collection can have a different structure, which allows for more flexible data modeling.
- **Data Storage**: In MongoDB, you can store related data within a single document. For example, instead of creating separate tables for users and their

hobbies, you can store a user's hobbies as an array within the user's document. This eliminates the need for joins and complex queries.

- **Ease of Use**: MongoDB and other document databases are designed to work well with JavaScript. Since documents are similar to JSON objects, they integrate seamlessly with JavaScript code, making it easier to work with data in web applications.
- **Schema Flexibility**: NoSQL databases offer schema flexibility, allowing you to add or modify fields without affecting existing documents. This makes it easy to adapt to changing requirements and store diverse types of data.

Feature	RDBMS (Relational Database)	NoSQL (Document Database)
Table Structure	Tables with rows and columns	Collections with documents
Data Organization	Structured data in tables	Flexible, schema-less documents
Schema	Fixed schema, predefined	Schema-less, flexible
Query Language	SQL	NoSQL queries (varies by database)
Scaling	Tough horizontal scaling	Easier horizontal scaling
Relationships	Foreign keys and joins	Embedded documents, arrays
Use Case	Read-heavy apps, transaction workloads	Flexible data models, high-performance applications
Examples	Banking apps, ERP systems	Content management systems, real-time analytics

- **Table Structure**: RDBMS organizes data into tables with rows and columns. In contrast, NoSQL document databases use collections of documents, where each document can have a flexible and nested structure.
- **Data Organization**: RDBMS has a structured format, while NoSQL offers a flexible schema, allowing documents to vary in structure.
- **Schema**: RDBMS requires a fixed schema, meaning the structure must be predefined and is not easily changeable. NoSQL databases have a schema-less design, allowing for easy changes and adaptability.

- **Query Language**: RDBMS uses SQL for querying, which is a standardized language for relational databases. NoSQL databases have their own query mechanisms, which can vary depending on the database system.
- **Scaling**: RDBMS generally faces challenges with horizontal scaling (scaling out across multiple servers). NoSQL databases are designed for easier horizontal scaling, making them more suitable for distributed environments.
- **Relationships**: RDBMS uses foreign keys and joins to manage relationships between tables. NoSQL document databases handle relationships by embedding related data within documents or using references.
- **Use Case**: RDBMS is often used in scenarios requiring complex transactions and strong consistency, such as banking systems. NoSQL databases are preferred for applications needing flexible data models and high-performance, such as content management systems and real-time data processing.
- **Examples**: RDBMS examples include banking applications that need ACID compliance and complex transactions. NoSQL examples include systems like content management and real-time analytics platforms, where flexibility and scalability are key.