Data Storage & Infrastructure

Amanda Farghli, Fatoumata Drammeh, Khadiza Khanom, Maisha Islam, Alexandr Voronovich, Sharmin Zaman

Relational Database Systems

STRUCTURED

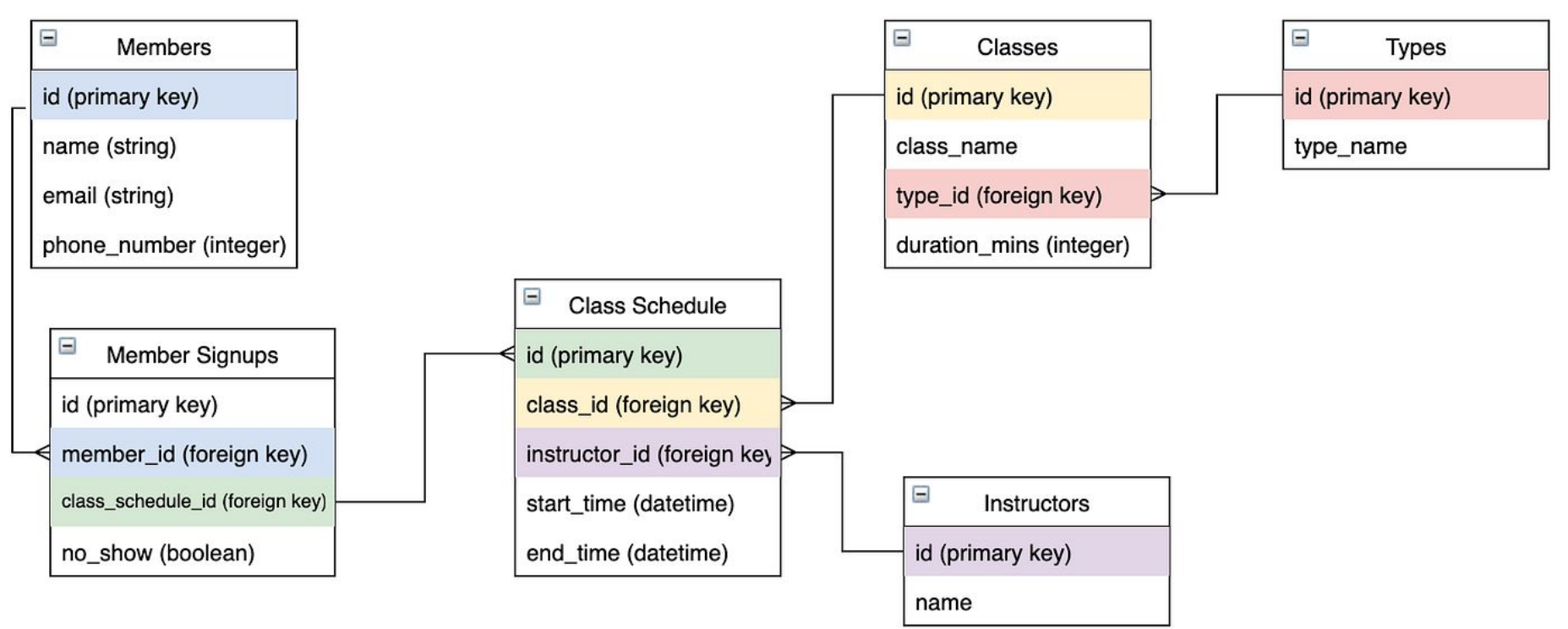
- Data stored in tables
- Not difficult to analyze

A relational database system follows a more rigid, structured model.

The database is fractured into various tables with an identifying key which defines data elements and their relationships to each other.

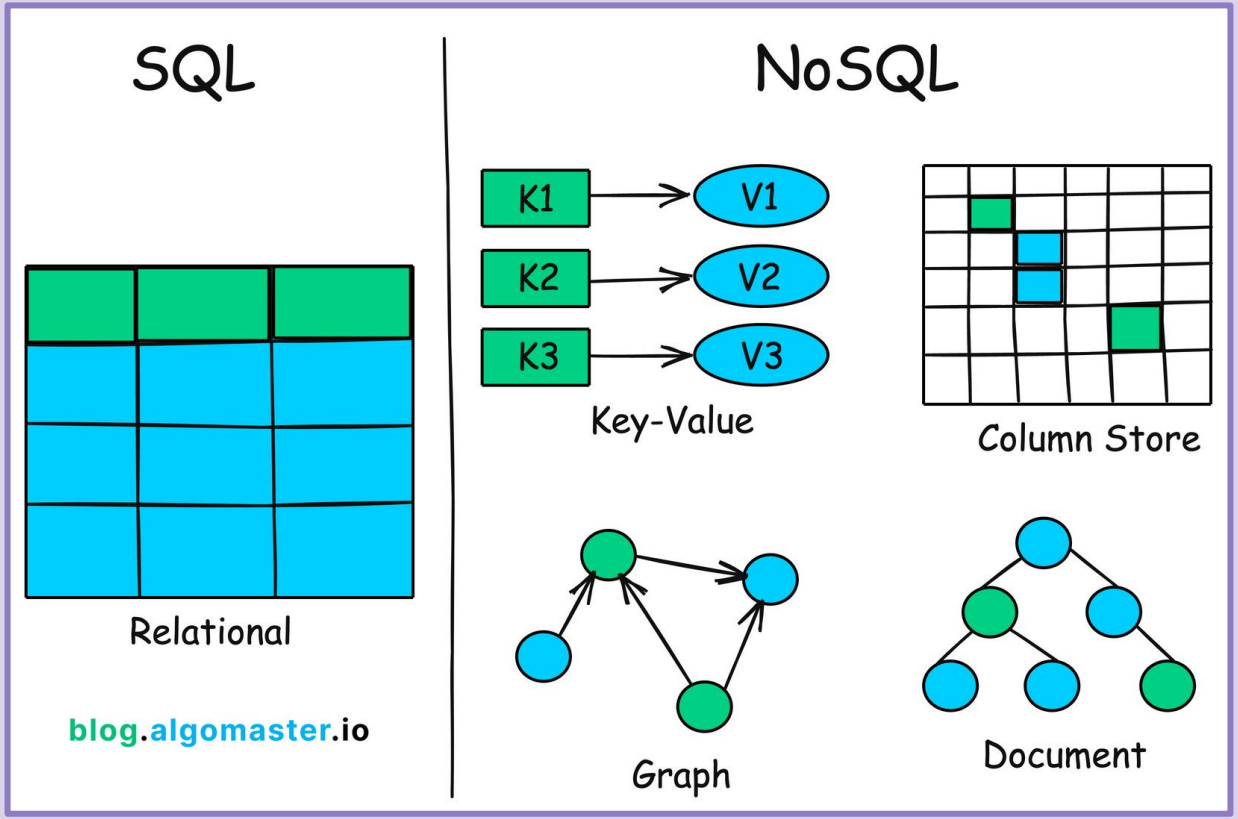
Key	Value
Name	John
ID_number	12345

Relational Database Systems

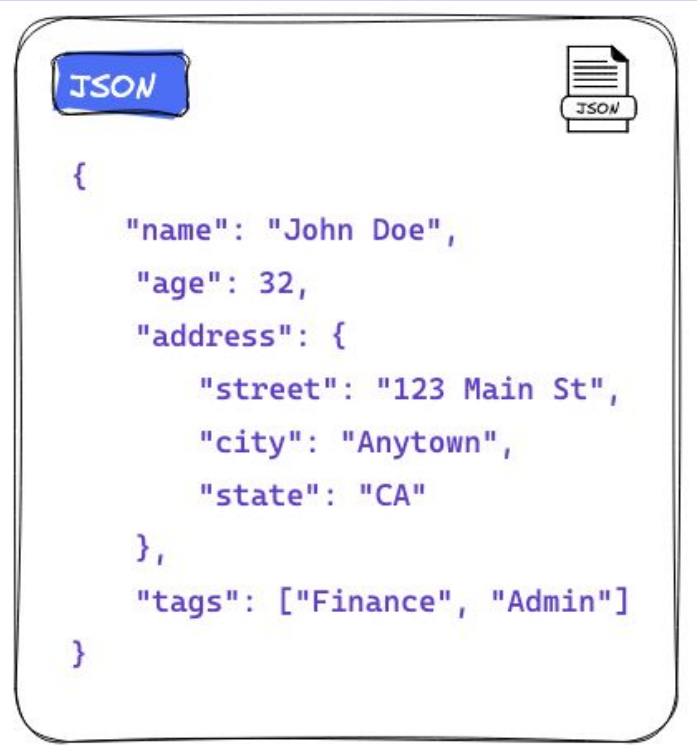


MYSQL & Postgre

SQL VS NoSQL


SQL
NoSQL
K1
V1
K2
V2
K3
V3
Key-Value
Column Store
Relational
blog.algomaster.io
Graph
Document

Document-oriented databases



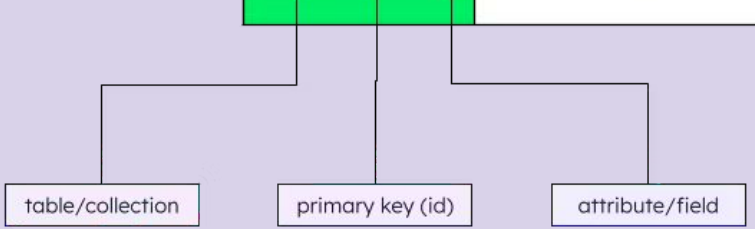
mongoDB.

cb

Key Value

Key value database storage

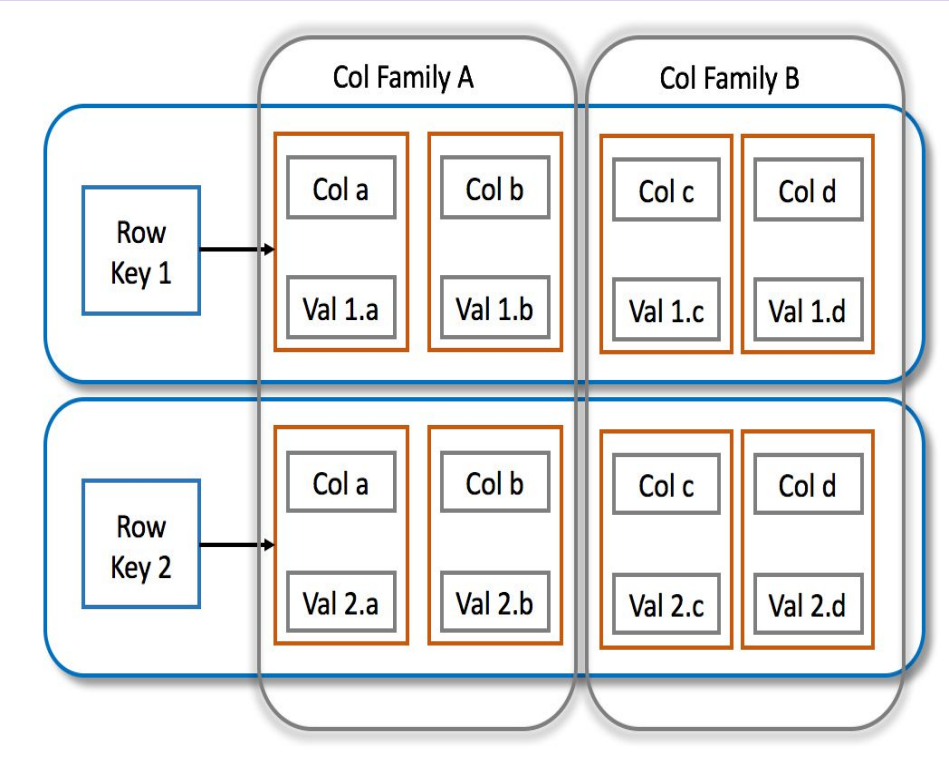
Key	Value
customer:1:name	John Drake
customer:1:email	john.drake@gmail.com
customer:1:dob	24/11/1982
customer:1:mobile	7843241098



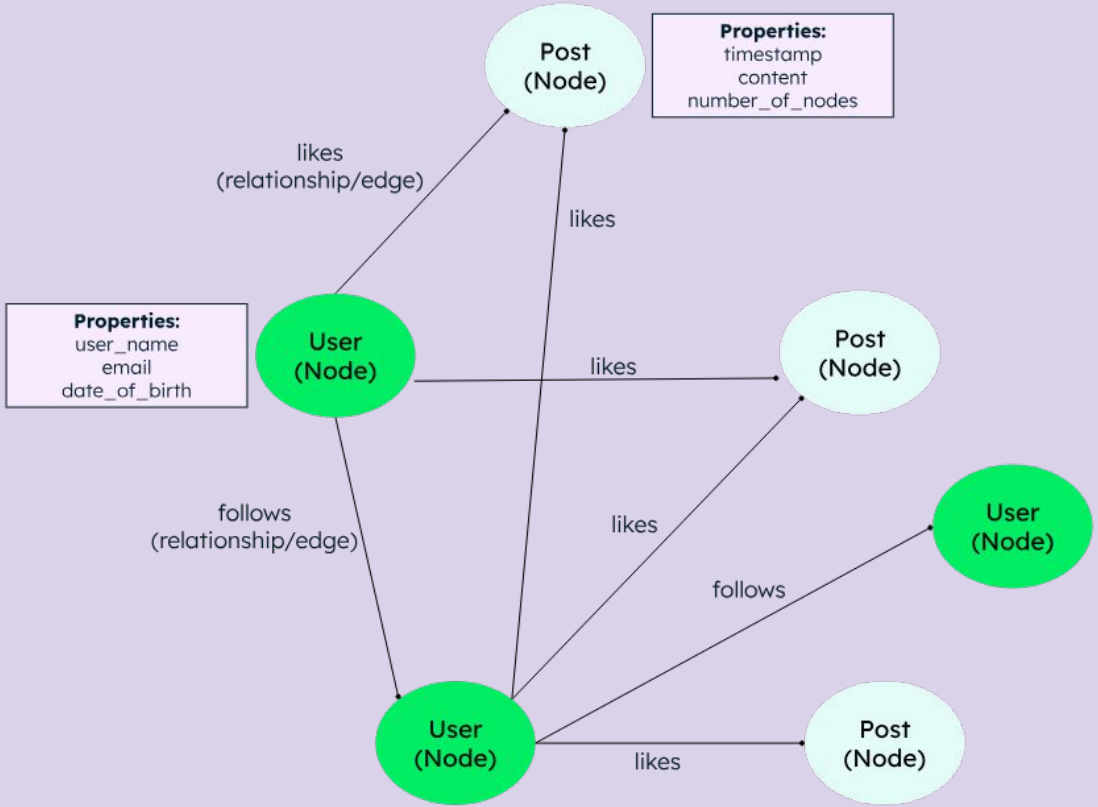
Amazon DynamoDB

redis

Wide-column stores



Graph databases

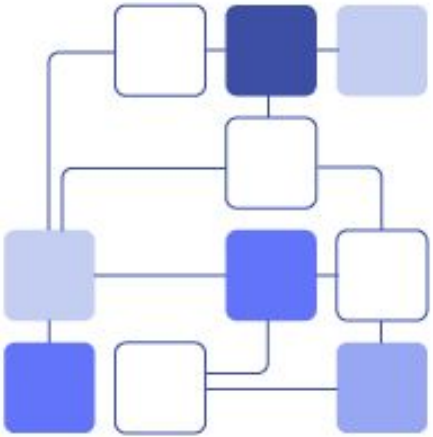


neo4j

Amazon
Neptune

NoSQL

Unstructured



- Dynamic schema
- Non relational model
- Horizontally scalable
- Suited for hierarchical data storage

Elasticsearch & Cassandra

- Full-text search & real-time analytics
- JSON-based, automatically indexed data
- Part of ELK stack (Logstash, Kibana)

- Distributed, column-based NoSQL database
- Built for scalability & uptime
- Handles massive data across servers

Which Database to Use & When

Relational Databases (SQL)

- Examples: MySQL, PostgreSQL, Oracle, SQL Server

NoSQL Databases

- Examples: MongoDB, CouchDB (JSON-like documents)

Relational Databases (SQL)

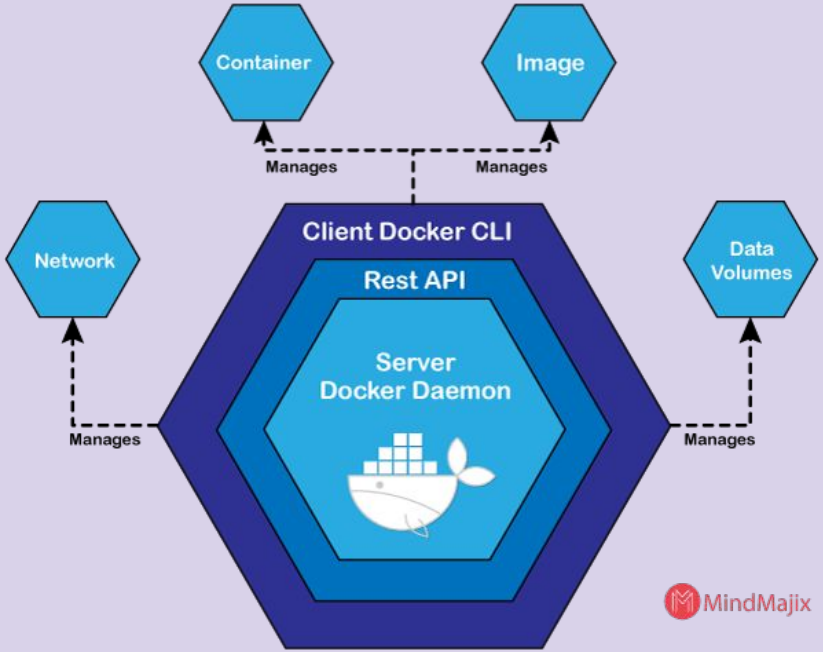
- Use for structured data with predictable schemas, like customer records or financial transactions. Ideal for complex queries, reporting, or apps needing strong consistency

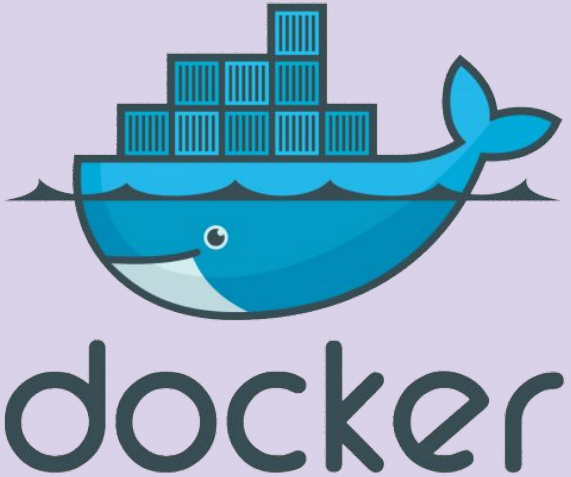
NoSQL Databases

- Use for unstructured or semi-structured data, rapid scaling, or flexible schemas. Choose document stores (MongoDB) for content management, key-value (Redis) for caching, column-family (Cassandra) for big data

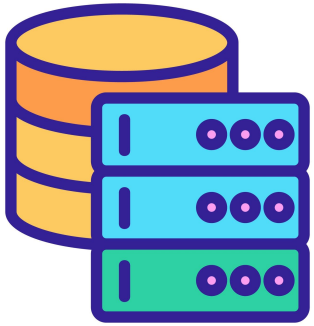
What is Docker?

A platform for developing, packaging, and running applications inside containers





What are the uses for Docker



Collaboration & Team Development

Docker allows for team members to work on the same shared database environment ensuring inconsistency.

Connecting Databases

Docker links your database to your project within a single container. It allows for seamless API calls and data requests.

Flexibility

Docker provides flexibility by supporting connections to many kinds of databases such as MySQL, Postgre, NoSQL databases, etc. No need to download and run 5 different database types.

Containers

Dockers allows for multiple different projects to exists in their own containers, creating isolated environments that prevent conflict.,

How to Create a Container, push and pull an Image(MySQL)

1. **Create a Container:** docker run --name containerName -e MYSQL_ROOT_PASSWORD=password -p 3306:3306 -d mysql
2. **Make edits to container:** CREATE DATABASE project; USE project; CREATE TABLE students (id INT…);
3. **Make a repository:** Got to https://hub.docker.com and create a repository and give it a name.
4. **Build & Tage the Image:** docker build -t yourUserName/repoName:tagName .
5. **Push to repository in Docker Hub:** docker push yourUserName/repoName:tagName
6. **Pull image:** docker pull yourUserName/repoName:tagName
7. **Run a container on pulled image:** docker run --name teammate -e MYSQL_ROOT_PASSWORD=password -p 3306:3306 -d yourUserName/repoName:tagName

This only initializes the databases. Any changes will have to be shared through dumps.