

Normal

ADVANCED DATABASES

Databases II

Author: Eng. Carlos Andrés Sierra, M.Sc.
cavirguezs@udistrital.edu.co

Full-time Adjunct Professor
Computer Engineering Program
School of Engineering
Universidad Distrital Francisco José de Caldas

2025-III



Outline

1 Object-Oriented Databases (OODB)

2 NoSQL Databases

3 Parallel Databases

4 Distributed Databases



Outline

1 Object-Oriented Databases (OODB)

2 NoSQL Databases

3 Parallel Databases

4 Distributed Databases



What is an OODB?

~2000s

- Combines object-oriented programming and database principles.
- Stores **objects** with their methods and attributes.
- Supports complex data and inheritance.

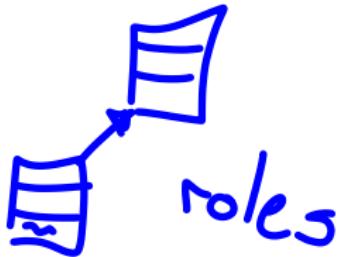
*calculations
queries*



What is an OODB?

- Combines **object-oriented programming** and **database principles**.
- Stores **objects** with their **methods** and **attributes**.
- Supports **complex data** and **inheritance**.

trees
graphs



OODB Features

- Encapsulation, Inheritance, Polymorphism.
- Direct object storage and identity.
- Supports OQL (Object Query Language).



OODB Features

- Encapsulation, Inheritance, Polymorphism.
- Direct object *storage* and identity.
- Supports OQL (*Object Query Language*).

Demo time!



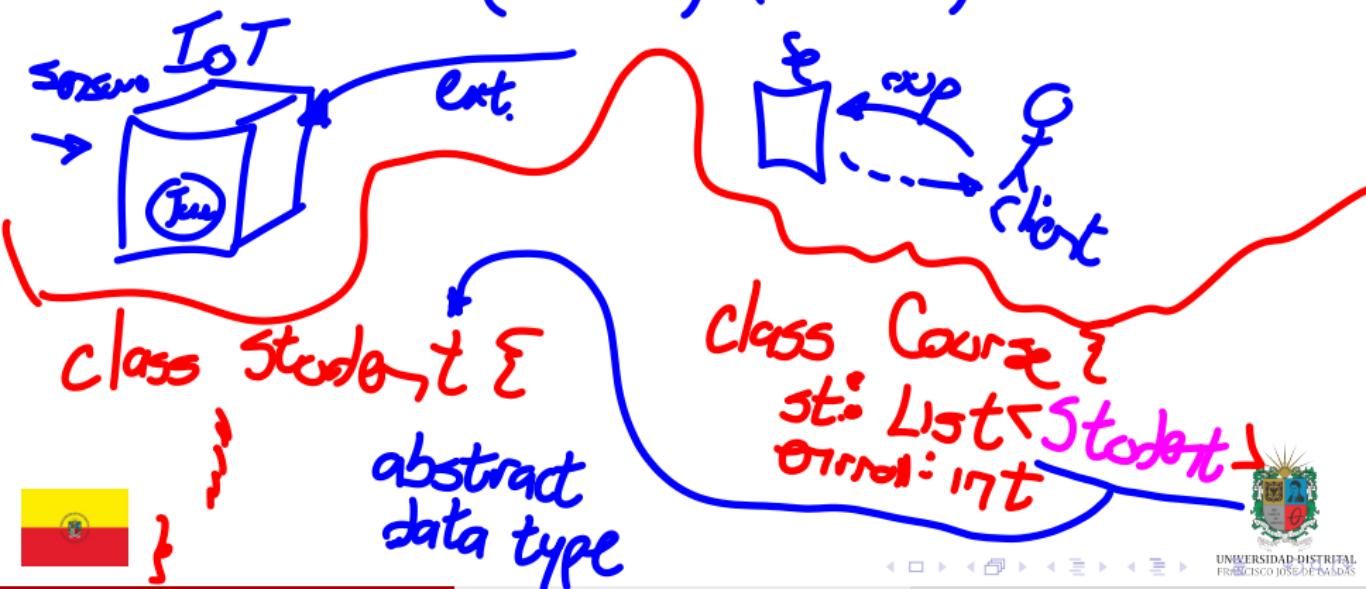
OODB Architecture

- Integrates:

JVM / MICSIL



- Common deployment (embedded) or client-server



Persistence Mechanisms

- By Reachability: Root object persistence.

Rule of Least Knowledge

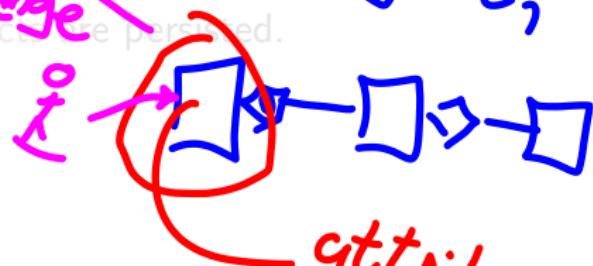
- By Depth Traversing: Annotate persistable objects.

- By Modification: Modified objects are persisted.

bad practice

`a.getB(); getc();`

Composite, aggregation,



attribute
abstract
type *data*



Persistence Mechanisms

- By Reachability: Root object persistence.
- By Explicit Marking: Annotate persistable objects.
- By Modification: Modified objects are persisted.

@Save

```
class A {  
    b:B; x  
    c:C; /  
}
```

```
class B {  
};  
{
```

@Save

```
class C {  
};  
{
```



Persistence Mechanisms

- **By Reachability:** Root object persistence.
- **By Explicit Marking:** Annotate persistable objects.
- **By Modification:** Modified objects are persisted.

att.: null
att2: "a"



OODB Query Example (OQL)

OOI

Student s = new Student()

```
SELECT s.name FROM Student s WHERE s.average() > 4.0
```

- Queries can follow object references.
- Better for nested, recursive, or complex types.

class Student {

```
    public String name;
    private double[] grades;
    public double average()
        return Math.avg(grades);
}
```

Demo time!



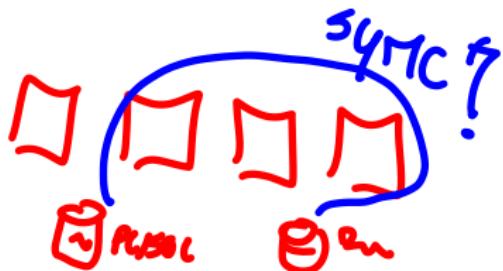
OODB Query Example (OQL)

OQL

```
SELECT s.name FROM Student s WHERE s.average() > 4.0
```

- **Queries** can follow object references.
- Better for **nested**, **recursive**, or **complex types**.

Demo time!



Outline

- 1 Object-Oriented Databases (OODB)

- 2 NoSQL Databases

- 3 Parallel Databases

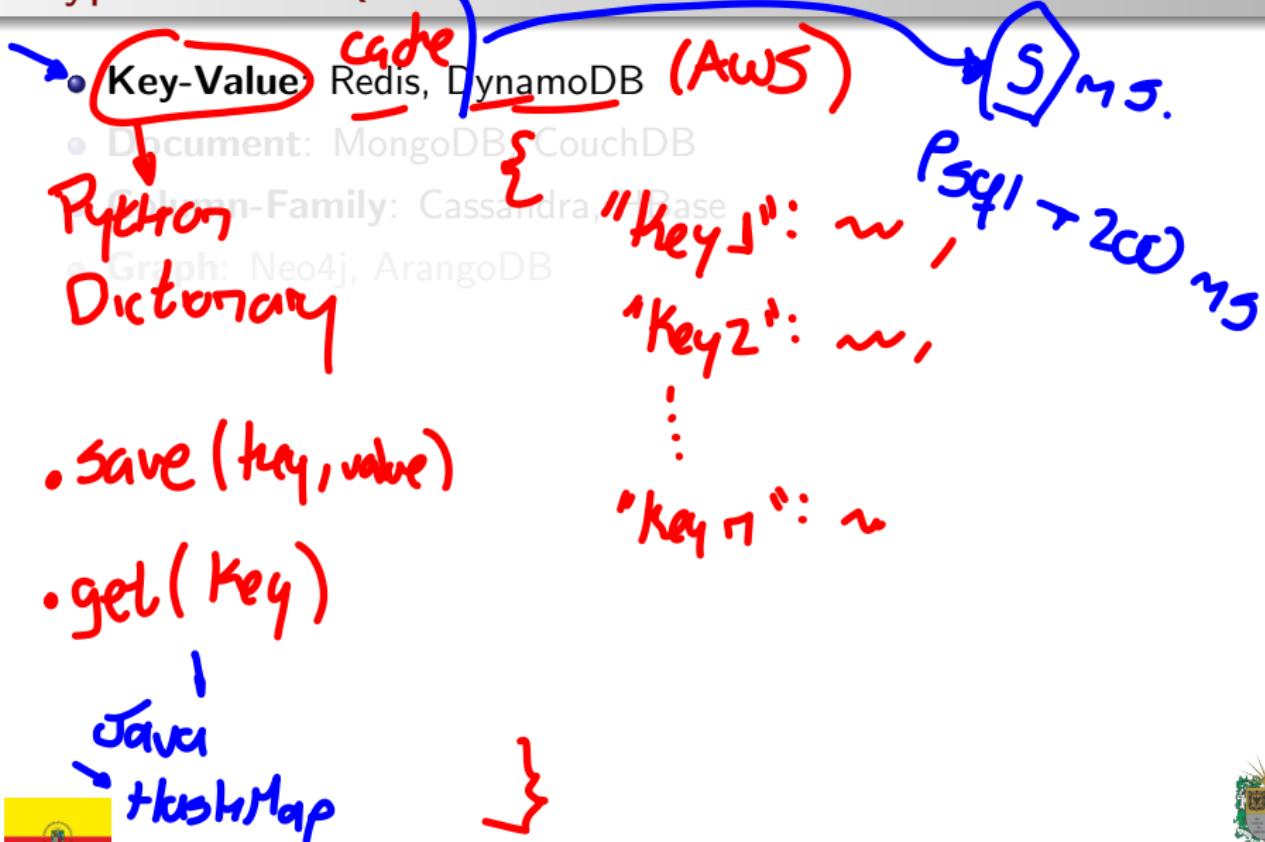
- 4 Distributed Databases

No mandatory

Non-structured
semi-structured



Types of NoSQL Databases



Types of NoSQL Databases

- **Key-Value:** Redis, DynamoDB
- **Document:** MongoDB, CouchDB

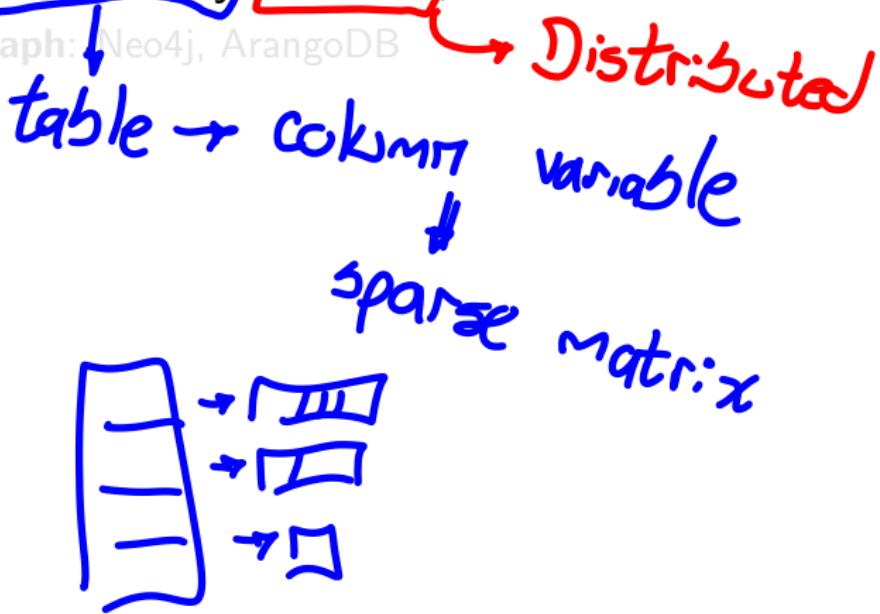
- **Column-Family:** Cassandra, HBase
- **Graph:** Neo4j, ArangoDB

LIST OF Dictionaries
JSON

document
Collection
→ no same attributes in all documents

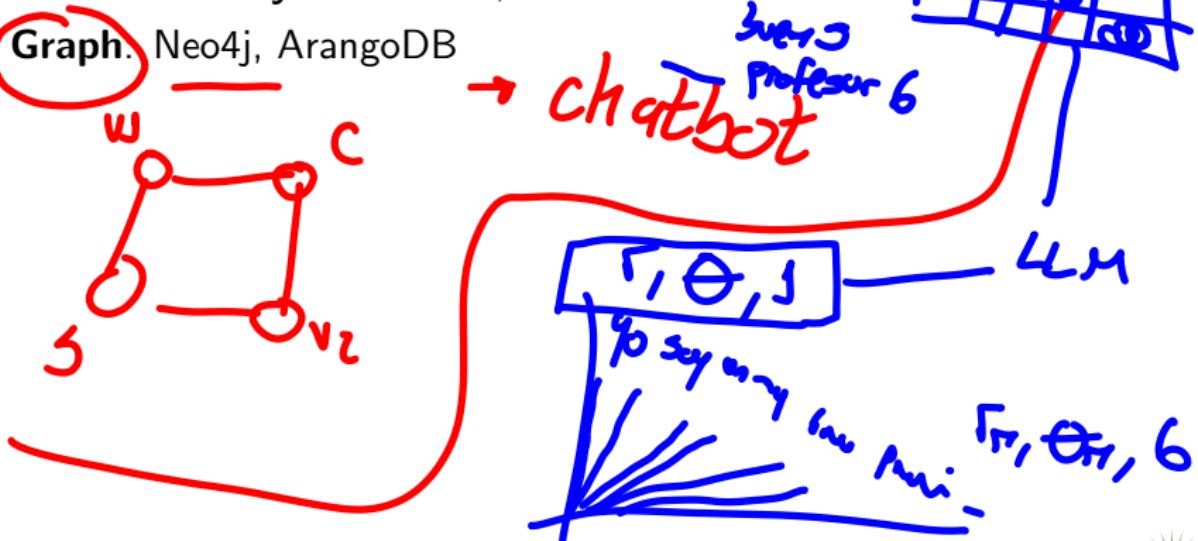
Types of NoSQL Databases

- **Key-Value:** Redis, DynamoDB
- **Document:** MongoDB, CouchDB
- **Column-Family:** Cassandra, HBase
- **Graph:** Neo4j, ArangoDB



Types of NoSQL Databases

- **Key-Value:** Redis, DynamoDB
- **Document:** MongoDB, CouchDB
- **Column-Family:** Cassandra, HBase
- **Graph.** Neo4j, ArangoDB



CAP Theorem Deep Dive

C Consistency (C)

Latest write is visible to all reads.



A Availability (A)

System always responds.

→ no loss request

P Partition Tolerance (P)

Tolerates network splits

Only two can be guaranteed at the same time.



CAP: Practical Examples

Database	CAP Preference	Comment
MongoDB	A + P	Eventual Consistency
Cassandra	A + P	Tunable consistency
Spanner	C + P	Sacrifices availability

Google
DataLake

deadlines
by sync.

clusters
Atlas



NoSQL Architecture Patterns

- **Shared-nothing architecture** enables scalability.

- Sharding: Distributes data across partitions.

- Replication: Ensures fault-tolerance.

Routers coordinate requests across shards (e.g., mongos in MongoDB).



data
sync

no memory
no disk
no processing

→ Cost ↑

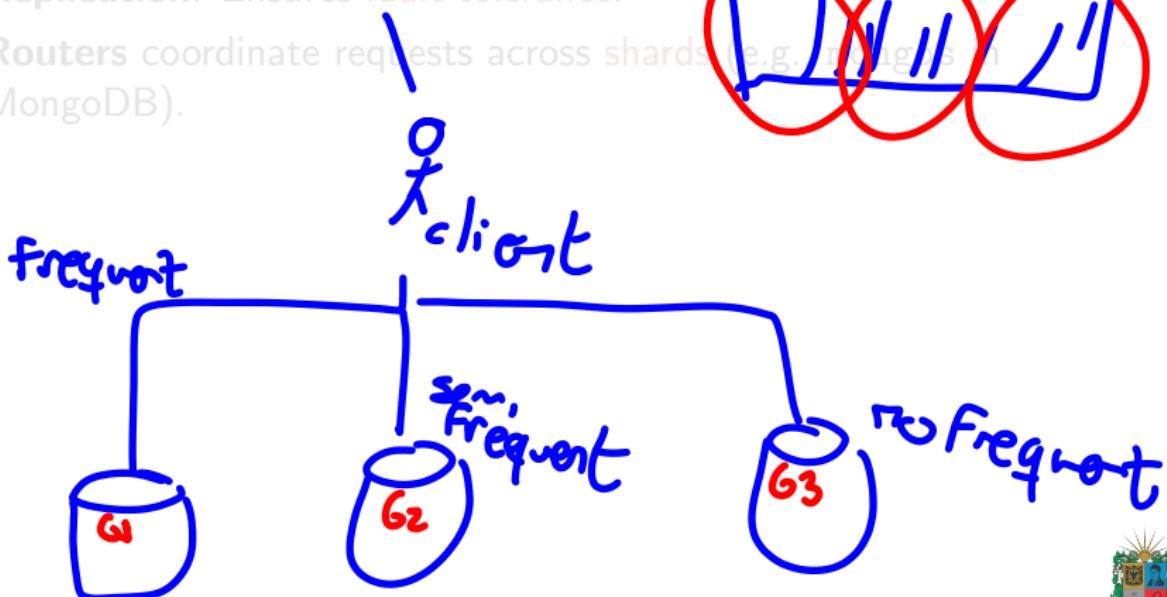
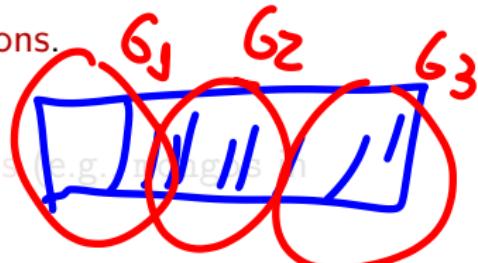


NoSQL Architecture Patterns

- **Shared-nothing architecture** enables **scalability**.
- **Sharding:** Distributes data across **partitions**.

• **Replication:** Ensures **fault-tolerance**.

• **Routers** coordinate requests across **shards** (e.g., mongos in MongoDB).



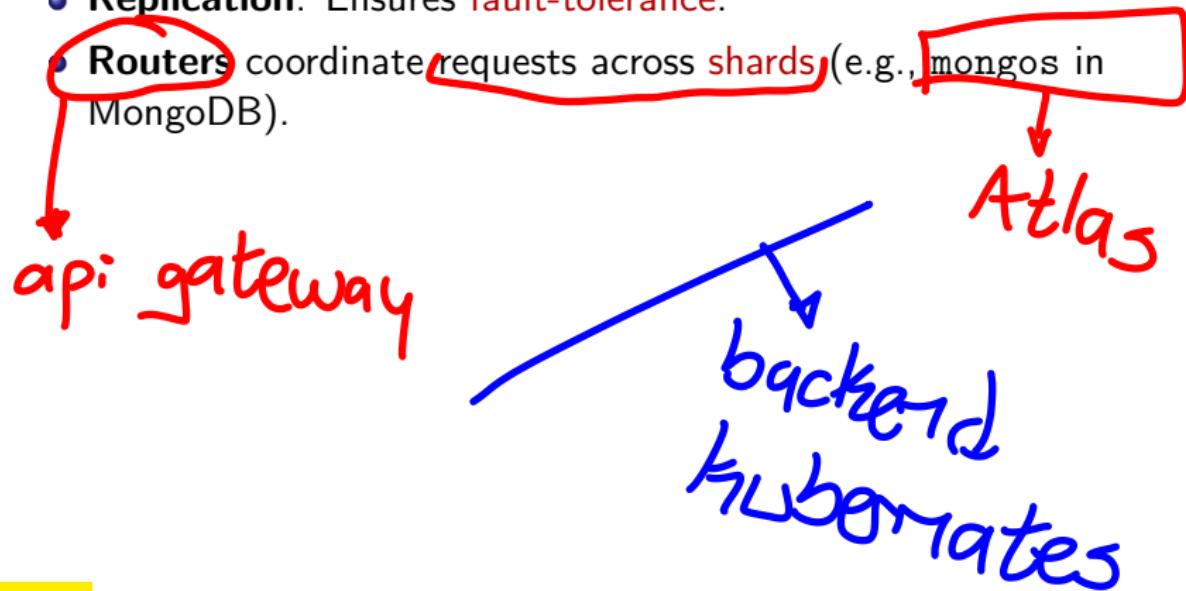
NoSQL Architecture Patterns

- **Shared-nothing architecture** enables **scalability**.
- **Sharding:** Distributes data across **partitions**.
- **Replication:** Ensures **fault-tolerance**.
- Routers coordinate requests across **shards** (e.g., mongos in MongoDB).

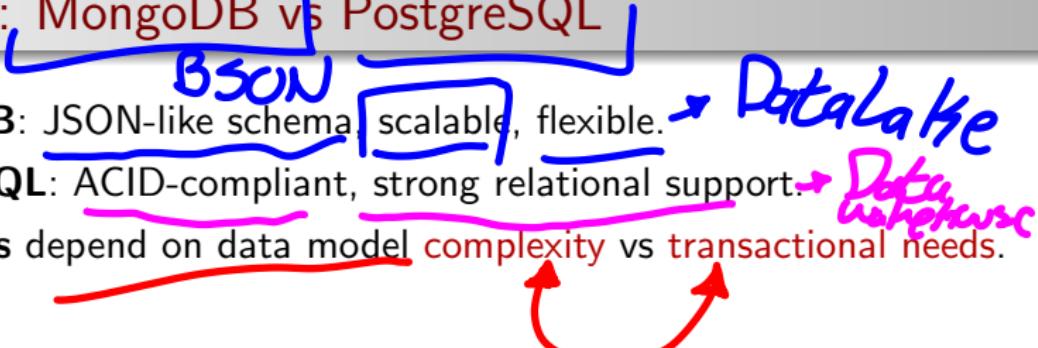


NoSQL Architecture Patterns

- **Shared-nothing architecture** enables **scalability**.
- **Sharding**: Distributes data across **partitions**.
- **Replication**: Ensures **fault-tolerance**.
- **Routers** coordinate requests across **shards** (e.g., **mongos** in MongoDB).



Case Study: MongoDB vs PostgreSQL

- **MongoDB:** JSON-like schema, scalable, flexible.
 - **PostgreSQL:** ACID-compliant, strong relational support.
 - **Trade-offs** depend on data model complexity vs transactional needs.
- 

Demo time!



Case Study: MongoDB vs PostgreSQL

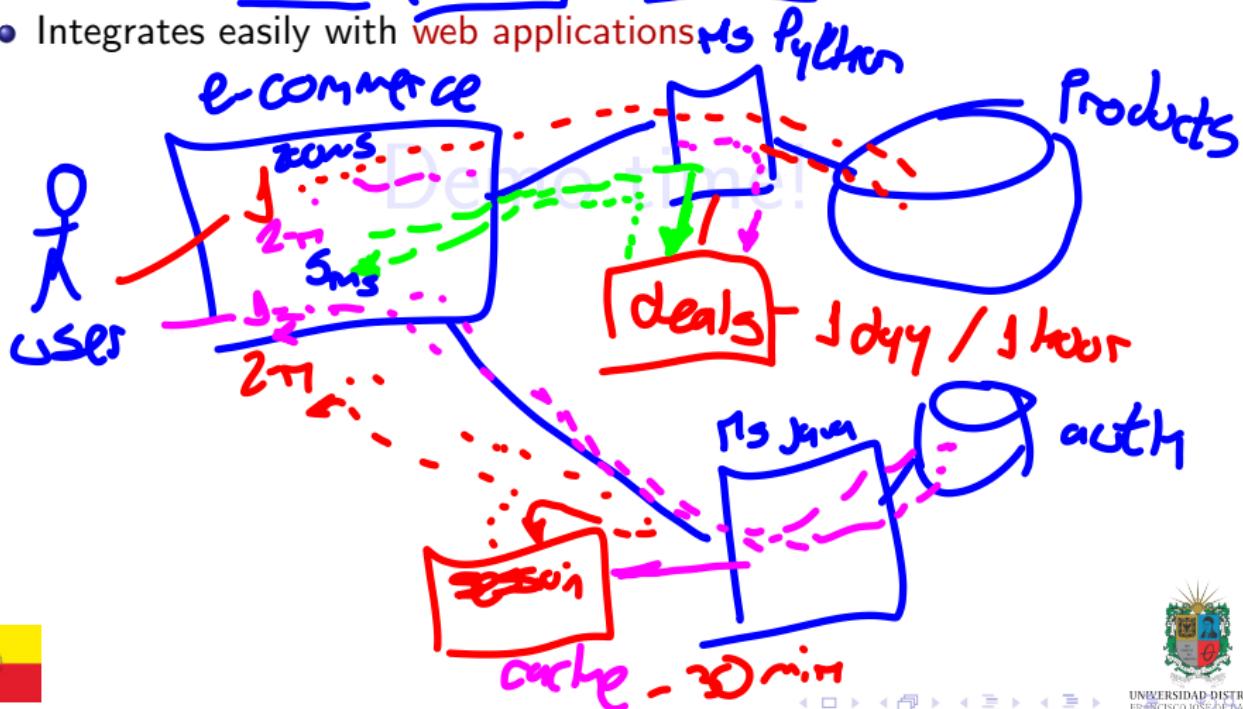
- **MongoDB:** JSON-like schema, scalable, flexible.
- **PostgreSQL:** ACID-compliant, strong relational support.
- **Trade-offs** depend on data model **complexity** vs **transactional needs**.

Demo time!



Case Study: Redis with Web API

- In-memory key-value store
- Excellent for caching, sessions, and real-time analytics.
- Integrates easily with web applications MS Python



Case Study: Redis with Web API

- In-memory key-value store.
- Excellent for caching, sessions, and real-time analytics.
- Integrates easily with web applications.

Demo time!



Outline

1 Object-Oriented Databases (OODB)

2 NoSQL Databases

3 Parallel Databases

4 Distributed Databases



Parallel DB Concepts

- Uses multiple processors to speed up query execution.
- Exploits parallelism in data access and computation.
- Reduces query response time for big datasets.



Partitioning Strategies

- **Horizontal:** Distributes **rows**.
- Vertical: Splits **columns**.
- **Hash Partitioning:** Uniform distribution via **hash**.
- **Range Partitioning:** Based on value **intervals**.



Partitioning Strategies

- **Horizontal:** Distributes **rows**.
- **Vertical:** Splits **columns**.
- **Hash Partitioning:** Uniform distribution via **hash**.
- **Range Partitioning:** Based on value **intervals**.



Partitioning Strategies

- **Horizontal:** Distributes **rows**.
- **Vertical:** Splits **columns**.
- **Hash Partitioning:** Uniform distribution via **hash**.
- **Range Partitioning:** Based on value **intervals**.



Partitioning Strategies

- **Horizontal:** Distributes **rows**.
- **Vertical:** Splits **columns**.
- **Hash Partitioning:** Uniform distribution via **hash**.
- **Range Partitioning:** Based on value **intervals**.



Parallel Query Cost Model

- **Startup cost (S)**: Fixed overhead.
- **Communication cost (C)**: Inter-node data transfer.
- **Computation cost (T)**: Local processing time.
- Total: $T_{total} = S + C + T$



Parallel DB Architectures

- **Shared Memory:** Easy communication, **poor scalability**.
- **Shared Disk:** Easier **fault-tolerance**, contention risk.
- **Shared Nothing:** Best scalability, **harder coordination**.



Parallel DB Architectures

- **Shared Memory:** Easy communication, **poor scalability**.
- **Shared Disk:** Easier **fault-tolerance**, contention risk.
- **Shared Nothing:** Best scalability, **harder coordination**.



Parallel DB Architectures

- **Shared Memory:** Easy communication, **poor scalability**.
- **Shared Disk:** Easier **fault-tolerance**, contention risk.
- **Shared Nothing:** Best scalability, **harder coordination**.



Case Study: PostgreSQL + Citus

- Open-source extension for **PostgreSQL** enabling parallel, distributed queries.
- **MPP architecture** (massively parallel processing) for scaling out across nodes.
- Good for **real-time analytics**, multi-tenant SaaS, and large OLAP workloads.
- Supports **sharding**, **replication**, and **distributed transactions**.

Demo time!



Case Study: PostgreSQL + Citus

- Open-source extension for **PostgreSQL** enabling parallel, distributed queries.
- **MPP architecture** (massively parallel processing) for scaling out across nodes.
- Good for **real-time analytics**, multi-tenant SaaS, and large OLAP workloads.
- Supports **sharding**, **replication**, and **distributed transactions**.

Demo time!



Outline

- 1 Object-Oriented Databases (OODB)
- 2 NoSQL Databases
- 3 Parallel Databases
- 4 Distributed Databases



What is a Distributed DB?

- **Data stored** across multiple **physical nodes**.
- *Appears* as a **single logical database**.
- Must ensure **consistency**, **availability**, and **partition tolerance**.



Transparency Goals

- **Location:** Hide physical location of data.
- **Replication:** Hide duplication.
- **Fragmentation:** Hide data partitioning.



Distributed DBMS Architecture

- **Client-Server Model:** Clients query; servers store data.
- **Federated Model:** Semi-autonomous DBs collaborate.
- **Peer-to-Peer:** Nodes act as both client and server.
- **Layers:** Global schema, transaction manager, local engines.



Distributed DBMS Architecture

- **Client-Server Model:** Clients query; servers store data.
- **Federated Model:** Semi-autonomous DBs collaborate.
- **Peer-to-Peer:** Nodes act as both client and server.
- **Layers:** Global schema, transaction manager, local engines.



Distributed DBMS Architecture

- **Client-Server Model:** Clients query; servers store data.
- **Federated Model:** Semi-autonomous DBs collaborate.
- **Peer-to-Peer:** Nodes act as both client and server.
- **Layers:** Global schema, transaction manager, local engines.



Distributed DBMS Architecture

- **Client-Server Model:** Clients query; servers store data.
- **Federated Model:** Semi-autonomous DBs collaborate.
- **Peer-to-Peer:** Nodes act as both client and server.
- **Layers:** Global schema, transaction manager, local engines.



Two-Phase Commit Protocol (2PC)

- ① **Prepare Phase:** Coordinator asks all participants to **prepare**.
- ② **Commit Phase:** If all vote yes, **coordinator commits**.

Issue: Blocking if coordinator crashes during commit.



Paxos (Simplified Consensus)

- Needed for agreement in **distributed systems**.
- **Roles:**
 - *Proposer*: Suggests a value.
 - *Acceptor*: Votes.
 - *Learner*: Learns chosen value.
- Ensures **consistency** even if nodes fail.



Case Study: Apache Cassandra

- Highly available and scalable NoSQL database.
- Designed for big data applications.
- Supports multi-data center replication.
- Offers a rich query language (CQL).

Demo time!



Case Study: Apache Cassandra

- Highly available and scalable NoSQL database.
- Designed for big data applications.
- Supports multi-data center replication.
- Offers a rich query language (CQL).

Demo time!



Outline

- 1 Object-Oriented Databases (OODB)
- 2 NoSQL Databases
- 3 Parallel Databases
- 4 Distributed Databases



Conclusion

- Data systems have evolved for **scalability** and **complexity**.
- Choosing the **right DB** model depends on the workload requirements.
- Understanding design **trade-offs** is key for architects.



Thanks!

Questions?



Repo: <https://github.com/EngAndres/ud-public/tree/main/courses/databases-ii>

