Python → Pandas SQL mean sort df.groupby(columns)

Advanced Databases

Databases II

Author: Eng. Carlos Andrés Sierra, M.Sc.

cavirguezs@udistrital.edu.co

Lecturer

Computer Engineer

School of Engineering

Universidad Distrital Francisco José de Caldas

Queue

index

date

2025-I

trade-off

Data Engineer

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Outline

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

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Outline

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

What is an OODB?

(2000) → Java → 95%

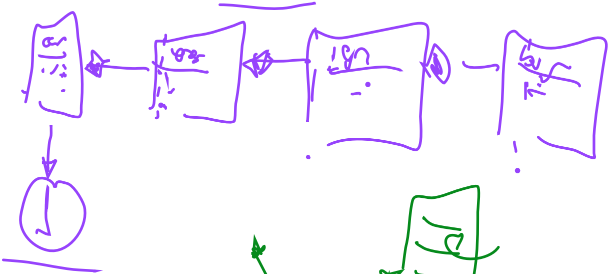
- Combines object-oriented programming and database principles.
- Stores **objects** with their methods and attributes.
- Supports complex data and inheritance.

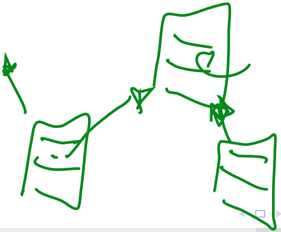
What is an OODB?

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

Design

máx 4 ← Join → 6-8-10





JSP/JSF

Java Serve-Pages

OODB Features

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

common
diff
diff

serialization

Student
-grade

public isScholarship : bool {
return grade >4.5 ? true : false;
}

public isApprove : bool {
return grade ≥ 3.0 ? true : false;
}

get set
grant

Validation
grade >= 0 && <= 5.0;

Persistence Mechanisms

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

Persistence Mechanisms

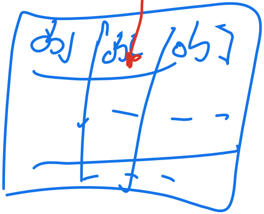
- **By Reachability**: Root object persistence.
- **By Explicit Marking**: Annotate persistable objects.
- **By Modification**: Modified objects are persisted.

```
@time
public int getAge(){
    ⋮
}
```

```
@persist
public class Student(){
}
```

Persistence Mechanisms

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



UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

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.

SELECT s.name FROM Student s
WHERE s.isScholarship();
(Legacy)

OODB Architecture

- Integrates:
  - **Object-Oriented Language Runtime**
  - **Persistence Engine**
  - **Query Processor (OQL)**
- Common deployment: embedded or client-server.

Object — attributes, methods

Trees Optimization Execution

Outline

1 Object-Oriented Databases (OODB)

2 NoSQL Databases

3 Parallel Databases

4 Distributed Databases

Types of NoSQL Databases

Apache – FOSS

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

Dict → {
"key": values
"key2": val
}

redis['key'] = [];

overwrite

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Types of NoSQL Databases

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

dict
dict
dict
dict
collection
C
query

Types of NoSQL Databases

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

No table

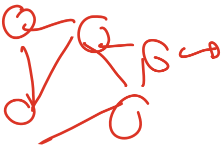
list of columns

write-update

reads

Types of NoSQL Databases

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



knowledge

CAP Theorem Deep Dive

Consistency (C)

Latest write is visible to all reads.

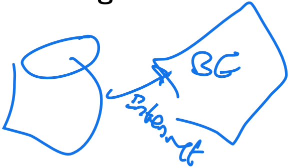
Availability (A)

System always responds.

Partition Tolerance (P)

Tolerates network splits.

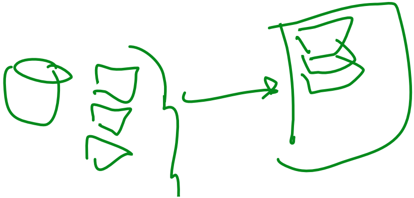
Only two can be guaranteed at the same time.



CAP: Practical Examples

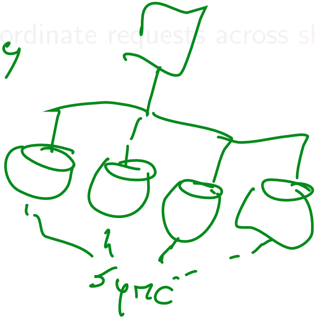
Database	CAP Preference	Comment
MongoDB	A + P	Eventual Consistency
Cassandra	A + P	Tunable consistency
Spanner	C + P	Sacrifices availability

TrueTime



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).

partition

sales

Jan 1 - dec 31 2021

Jan 1 - dec 31 2022

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).

middleware

Case Study: MongoDB vs PostgreSQL

BSON → sharding

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

& SQL

Hybrid → multi-model systems

Outline

1 Object-Oriented Databases (OODB)

2 NoSQL Databases

3 Parallel Databases

4 Distributed Databases

Parallel DB Concepts

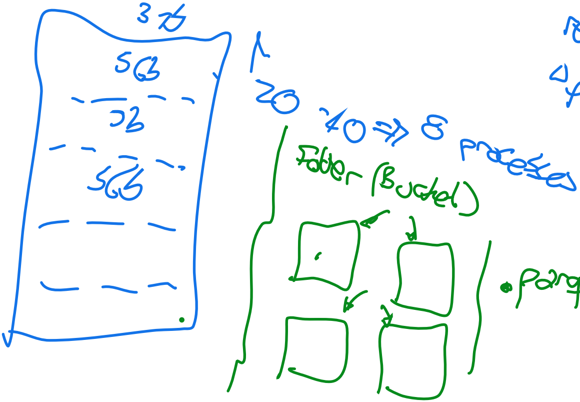
cores

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

independiente

read

update

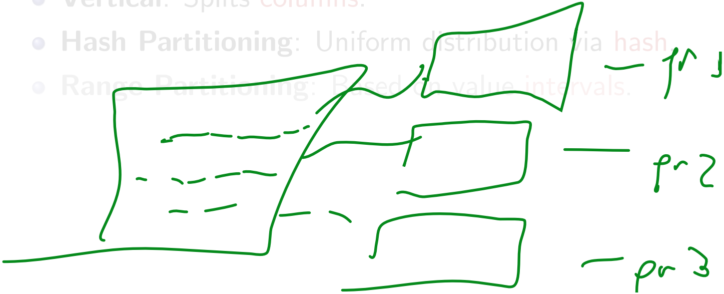


• parquet

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Partitioning Strategies

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



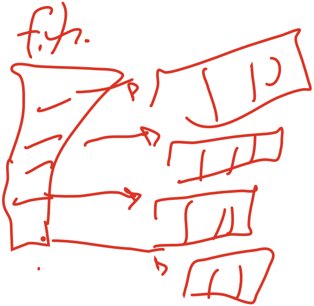
Partitioning Strategies

- **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.

rows



Partitioning Strategies

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

index 1 - 10 — pr1
11 - 20 — pr2
21 - 30 — pr3

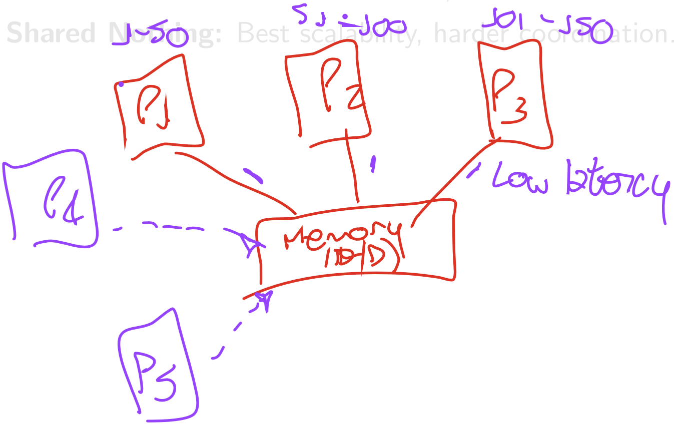
Parallel Query Cost Model

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

workers
jobs
cores

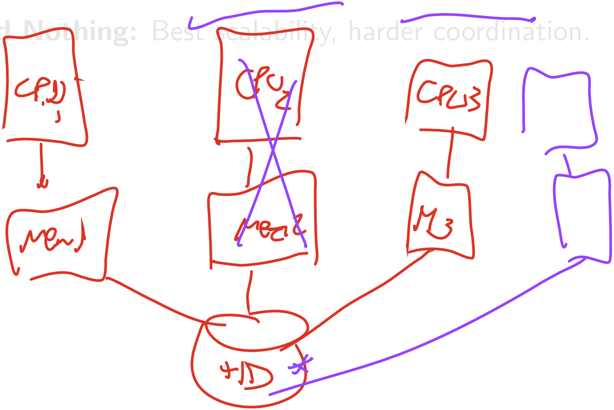
Parallel DB Architectures

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



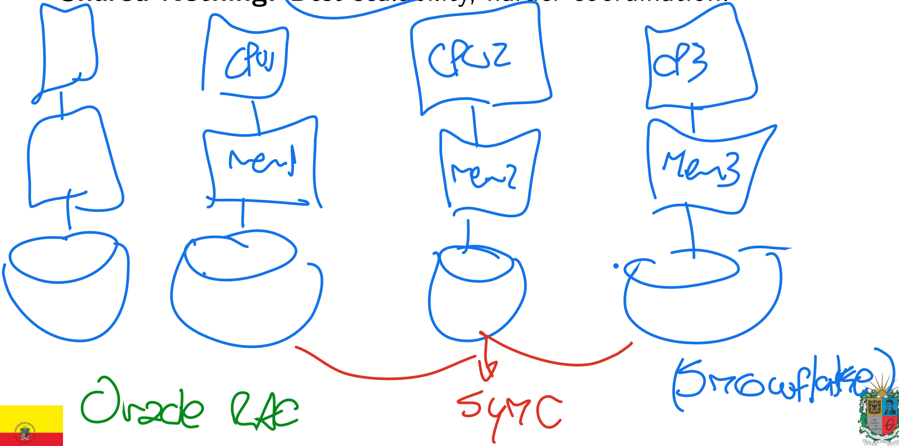
Parallel DB Architectures

- **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: Greenplum

relational

- Open-source parallel DB based on PostgreSQL.
- MPP architecture (massively parallel processing).
- Good for OLAP and analytics workloads.

10-100 cores

online

superset (SQL Builder)

DB lake

OLAP

App

real-time (5 min)

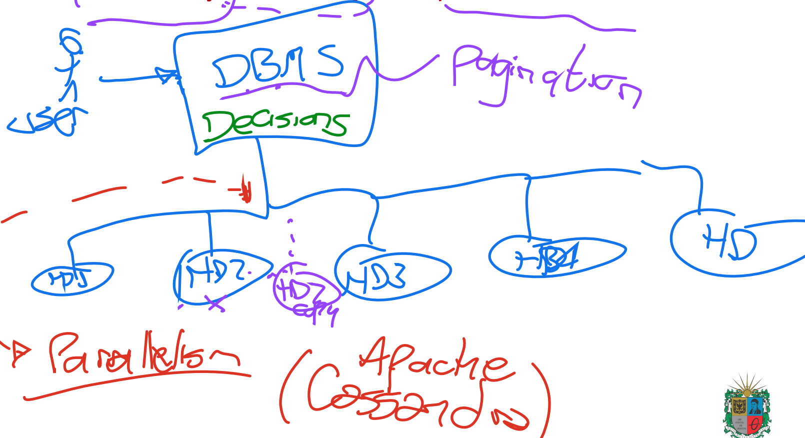
(latency ~ms)

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

user doesn't know

- **Location**: Hide physical location of data.
- **Replication**: Hide duplication. → scalability (redundant)
- **Fragmentation**: Hide data partitioning.

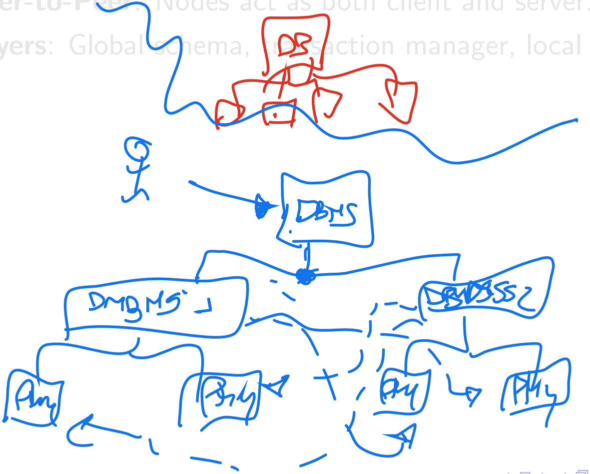
Distributed DBMS Architecture

- **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.

client
BE
async
Server
SQL
DB1
DB2
DB3
Duty Segregation
cluster (k8s)

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.

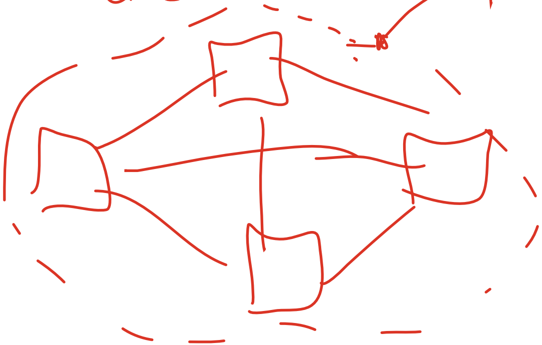


Distributed DBMS Architecture

- **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.

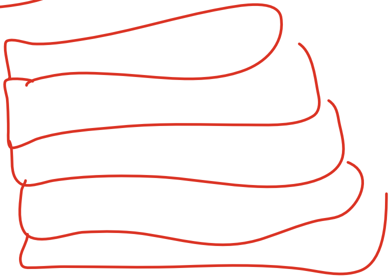
CouchBase

DBMS



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.



local

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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.

DBMS

Paxos (Simplified Consensus)

- Needed for agreement in distributed systems.
- **Roles**:
  - *Proposer*: Suggests a value.
  - *Acceptor*: Votes.
  - *Learner*: Learns chosen value.
- Ensures consistency despite node failures.

Consensus algorithm

blockchain

Case Study: Google Spanner

- Globally distributed RDBMS.
- Uses *TrueTime API* (atomic clocks + GPS).
- Offers external consistency and global transactions.

CONCURRENCY

Conclusion

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

twitter
amazon
uber

qualit. attr.
(security)

qualit. attr.
(scalability)

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Outline

1 Object-Oriented Databases (OODB)

2 NoSQL Databases

3 Parallel Databases

4 Distributed Databases

Thanks!

Questions?

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

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS