

# Models and Systems for Big Data Management

## INTRODUCTION

Nacéra Seghouani

Computer Science Department, CentraleSupélec  
Laboratoire de Recherche en Informatique, LRI LaHDAK group  
[nacera.bennacer@centralesupelec.fr](mailto:nacera.bennacer@centralesupelec.fr)

2019-2020

# Schedule and Assessment

✓ **Schedule:** 12 lectures, 5 x 2 labs

Week	Day		Course	
48	Tuesday	26/11	08:30-11:45	2 lectures
48	Friday	29/11	13:45-17:00	2 lectures
49	Tuesday	03/12	08:30-11:45	2 labs
49	Friday	06/12	13:45-17:00	2 Lectures
50	Monday	09/12	17:15-18:45	1 Lecture
50	Friday	13/12	13:45-16:45	2 Labs
51	Tuesday	17/12	08:30-11:45	2 Lectures
51	Friday	20/12	13:45-16:45	2 Labs
2	Tuesday	07/01	08:30-11:45	2 Lectures
2	Friday	10/01	13:45-16:45	2 Labs - Project
3	Monday	13/01	17:15-18:45	1 Lecture
3	Tuesday	14/01	08:30-11:45	2 Labs - Project

✓ **Assessment:** Written exam 50%, Quizzes 20%, Soft installation & tests 10% and mini-project 20%

Week	Day		Course	
49	Tuesday	03/12	11:30-11:45	Soft installation & tests, Quizz
50	Friday	13/12	16:30-16:45	Soft installation & tests, Quizz
51	Friday	20/12	16:30-16:45	Soft installation & tests, Quizz
2	Friday	10/01	13:45-16:45	Project
3	Tuesday	14/01	08:30-11:45	Project
4	Monday	20/01	10:00-12:00	Written Exam

# Brief History of Database Management

1960

Indexed Sequential Access Mechanism

**Hierarchical data model**, Information Management System IMS-IBM



**Network data model**, Charles Bachman 1969, Integrated Data Store IDS- General Electric

1970

**Relational data model**, Edgar Codd in IBM 1970

SystemR (IBM), Ingres (UCBerkeley),



Oracle

1980

dBase, DB2, Informix, Sybase, SQLServer

1990

Postgres, MySQL

2000

2010

Hadoop  
NoSQL Data models  
Dynamo, Cassandra, Hbase, ...  
MongoDB, ...  
Neo4j, ...

- Relational data model based on strong theoretical foundations. SQL language-based DBMSs.
- SQL progress: images, text, data warehousing and analytics, large data.
- GAFA deploys their data centers, Web 2000. **BD XML**, multimedia.
- New hardware (FPGA, GPGPU, SSD). Cloud storage since 2010. In-memory databases.
- Scaling and data distribution -> Parallel processing on platforms Hadoop, MapReduce, Spark, Flink
- NoSQL Lack of standards



# SQL and NoSQL Data Models and Systems



- ☞ A database is a collection of structured data related to a specific topic, organized according to a given data model.  
ex. modeling the availability of rooms in hotels to provide information about vacancies.
- ☞ A Database Management System (DBMS) allows users to access/update the database. It mainly allows:
  - ✓ creation, querying, update, and administration of a database.
  - ✓ interacting with end users, applications, and the database itself to capture and analyze data.
  - ✓ it does much more ...
- ☞ A database is generally stored in a DBMS-specific format, but different DBMSs can share data using other formats and APIs/libraries.
- ☞ Two main data models: Relational theory/SQL and NoSQL (Not only SQL )






# SQL and NoSQL Data Models and Systems

## SQL DBMSs:

Oracle  MySQL  SQL Server   
PostgreSQL  DB2  ...

## NoSQL data model-based DBMSs:



**Document:** MongoDB  mongoDB Cassandra  CouchBase  , ...

**Columnar:** Apache Parquet  HBase  , ...

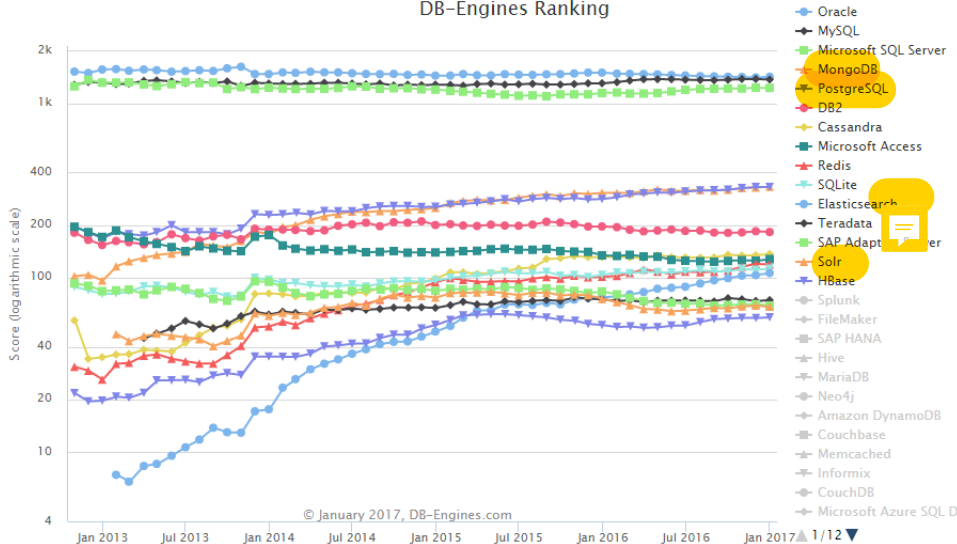
**Key-value:** DynamoDB  Oracle  InfinityDB  Redis

 , ...

**Graph:** Neo4j  Giraph  Neptune  , ...

# DBMS market

DB-Engines Ranking



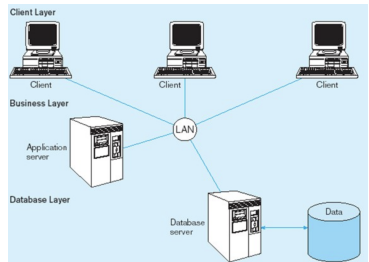
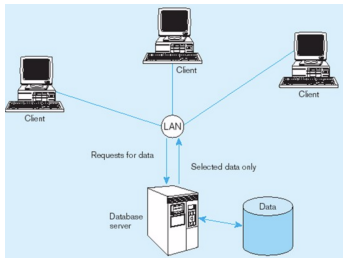
# DBMS Client-Server Architecture

☞ A DBMS is a process:




- ✓ running on a machine identified by its IP address on communication network
- ✓ providing services to clients via a given port

ex.: a web server which waits for connexions on port 80 is running on a machine IP 163.12.9.10, any web client, for instance Firefox, can communicate with this server at 163.12.9.10 :80.

- ✓ Three-Tier Architecture



# What is expected from a database?

- ✓ Persistent storage of massive data  
disk, SSD, ...
- ✓ Performance of querying   
index, query optimiser engine, ...
- ✓ Access concurrency  
transactions, locking protocols, 
- ✓ Reliability & availability, fault tolerance (failure recovery)  
failure recovery (transaction), logs/journals, replications ...
- ✓ Security, access control   
user access grants & profile, network security, ...
- ✓ Consistency of data  
All integrity constraints are satisfied (transaction).  
For ex. account balance must be  $\geq$  fixed level

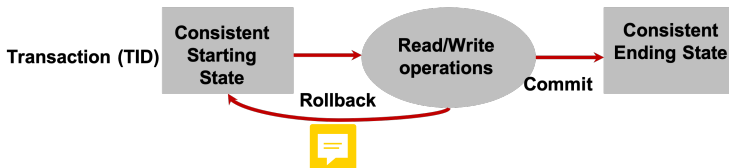


# How to deal with concurrent access: Transaction

## ☞ Some examples:

- ✓ Assume that two clients  $cl_1$  and  $cl_2$  are booking the same flight. Only one seat is available  $x = 1$ .  
 $read_1(x)$ ,  $read_2(x)$ ,  $write_2(x \leftarrow x - 1)$ ,  $write_1(x \leftarrow x - 1)$
- ✓ Assume now more complex operations: booking multiple flights from start to destination

- ☞ Transaction protects data against simultaneous access by concurrent processes (clients)
- ☞ Transaction does much more: a client process can update a subset of data as a single atomic operation. All (Commit) or nothing (Rollback) property. It may be aborted by client or DBMS server



# How to deal with concurrent access: Transaction

- Transaction provides reliable units of work that allows recovery failures, keeps consistent database even in cases of failures
- Transaction provides isolation between concurrent clients. Concurrent transaction don't interfere with one another. The final result appear as though all transactions ran sequentially (serializable in some dependent order)
- To achieve this: Locking protocol (2-Phase Locking (2PL)) & timestamping, locking granule, Private workspace(local copies), Write-ahead log, Prevent and detect deadlocks. Details come later



# ACID Transaction Properties (Harder & Reuter 1983)

- ➡ A-atomicity, the transaction takes place as a whole or not at all
- ➡ C-consistency, maps one consistent DB state to another. Updates Don't violate system invariants/integrity after transaction completion

In a banking system, a key invariant is the law of conservation of money. During internal transfer operations, this invariant may be violated But this violation isn't visible outside the transaction

- ➡ I-isolation, ensures that intermediate state of a transaction is invisible to other transactions until that transaction is committed. As a result, transactions that run concurrently appear to be serialized.
- ➡ D-durability, the results of a committed transaction are stored in disk and they persist even if a system failure occurs

# Distributed Database

- A distributed system consists of many servers that communicate over a communication network
- A distributed database is a database managed by different DBMS servers. These servers could be running in one or interconnected computers.
- The distribution is transparent, users must be able to interact with the system as if it were one logical system.
- Transactions (if handled) are transparent, each transaction must maintain database integrity across multiple databases.
- Two kind of distribution: replication and partitioning (or sharding).

# Distributed Database

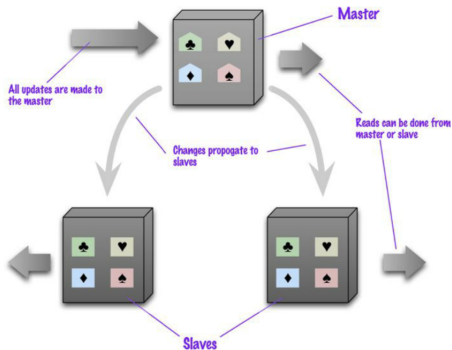
- ☞ In replication, the system maintains several identical copies in different machines to ensure that the distributed database remains up-to-date.
  - ✓ Complex, time-consuming and required network's resources, depending on the size and number of the distributed databases
  - ✓ Increased reliability and availability
  - ✓ Improved performance by allowing balance load among the database servers.
  - ✓ Replication comes in two forms:
    - Master Slave
    - Peer-to-Peer

# Distributed Database



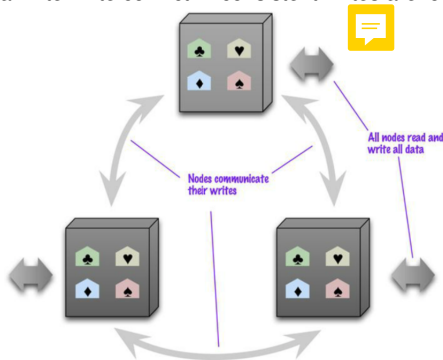
➡ Master-Slave: A replication process synchronizes the slave servers with the master server

- ✓ Master: the authoritative source for the data
- ✓ Slave: responsible for processing any updates to that data
- ✓ After a failure of the master, a slave can be appointed as new master very quickly
- ✓ More read requests
- ✓ Good for databases with a read-intensive data
- ✓ Inconsistency due to slow propagation of changes to the slaves



# Distributed Database

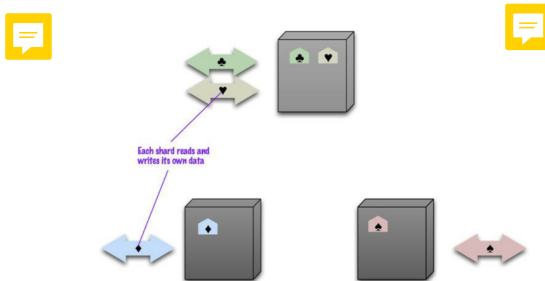
- 👉 Peer-to-Peer: All the servers have equal authority, they can all accept writes
  - ✓ The loss of any server doesn't prevent access to the data store.
  - ✓ Inconsistency due to slow propagation of changes to copies on different servers
  - ✓ Two client can update different copies of the same record stored on different nodes at the same time - a write-write conflict. Inconsistent writes are forever.



Master-slave replication reduces update conflicts but peer- to-peer replication avoids loading all writes onto a single point of failure.

# Distributed Database

- ☞ Sharding/partitioning distributes different data across multiple servers
  - ✓ Each server acts as the single source for a subset of data called partition or shard.
  - ✓ Improved performance of reads and writes and scaling (horizontal)
  - ✓ Sharding design should take into account application needs, otherwise joins become prohibitively expensive when performed across multiple servers.



Replication & Sharding could be combined in a distributed system.