NoSQL & NewSQL Databases

Responsible:
Dr. Tegawendé F. BISSYANDE
tegawende.bissyande@uni.lu

Course Author:
Dongsun Kim
dongsun.kim@uni.lu

Teacher Assistant:
Médéric Hurier
mederic.hurier@uni.lu

Course Features

- Sep. 20th - **Introduction to Big Data**

Part 1. Databases and Query Models for Big Data

- Sep. 27th - **Relational Databases: Reminders**
- Oct. 4th - **Relational Databases: Internals**
- Oct. 11th - **NoSQL & NewSQL Databases**
- Oct. 18th - **MapReduce Model**
- Oct. 25th - **Hadoop and Spark**
- Nov. 8th - **Datalog Model**

Part 2. Data Analysis and Machine Learning

- Nov. 15th - **Statistics and Probabilities**
- Nov. 22th - **Communication and Visualization**
- Nov. 29th - **Features Engineering and Supervised Learning**
- Dec. 5st - **Unsupervised and Reinforcement Learning**
- Dec. 12th - **Homework Time**

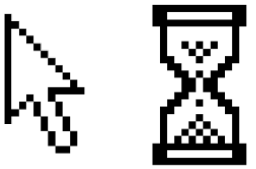
Section Features

- Motivation
- CAP Theorem
- Characteristics
- Database Landscape
- Example: MongoDB

Motivation

In a time far away ...

Everybody was happy about RDBMS/SQL !

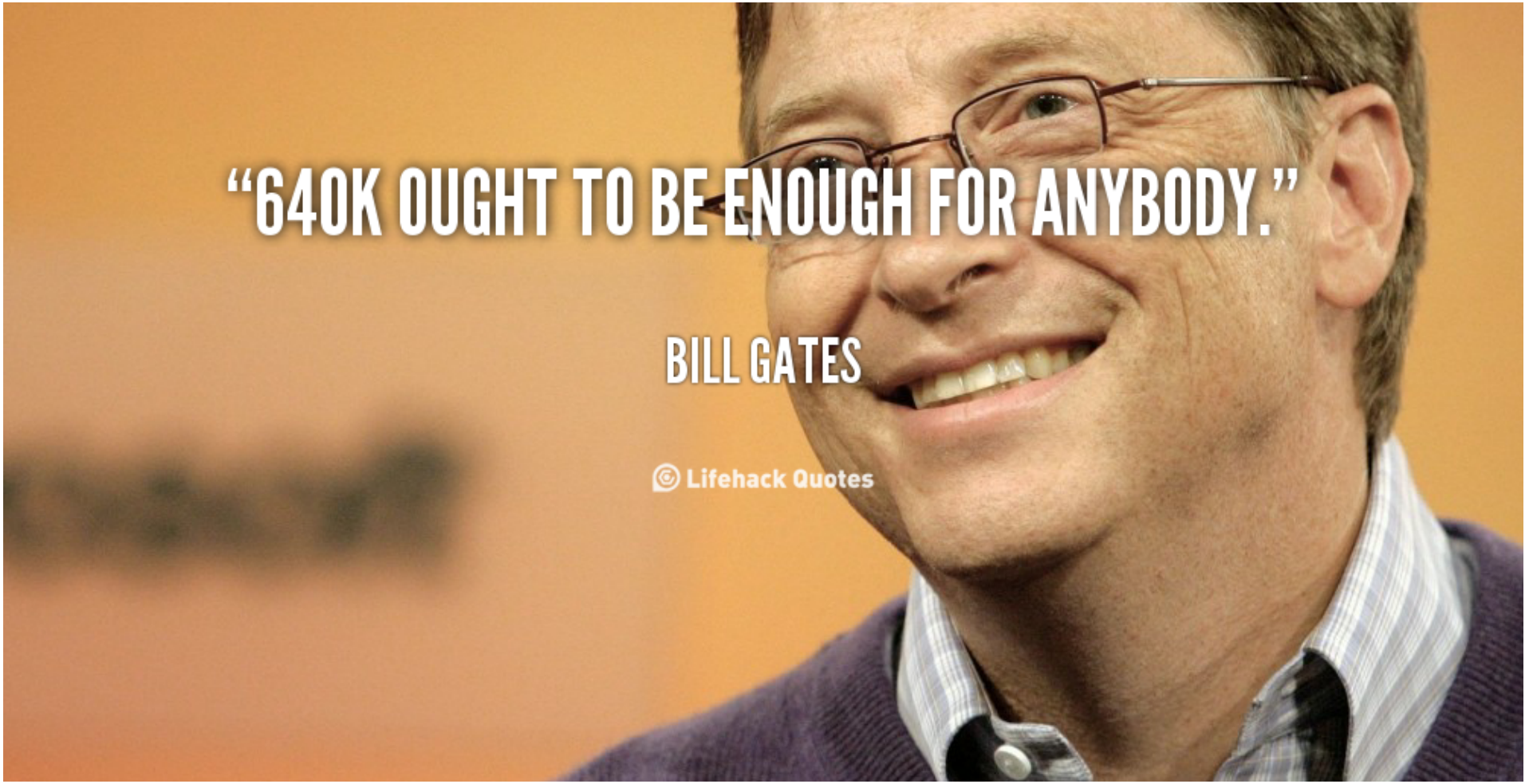


- Centralized
- Easy Maintenance
- Available anytime
- Stable performance
- Quite reliable (ACID)

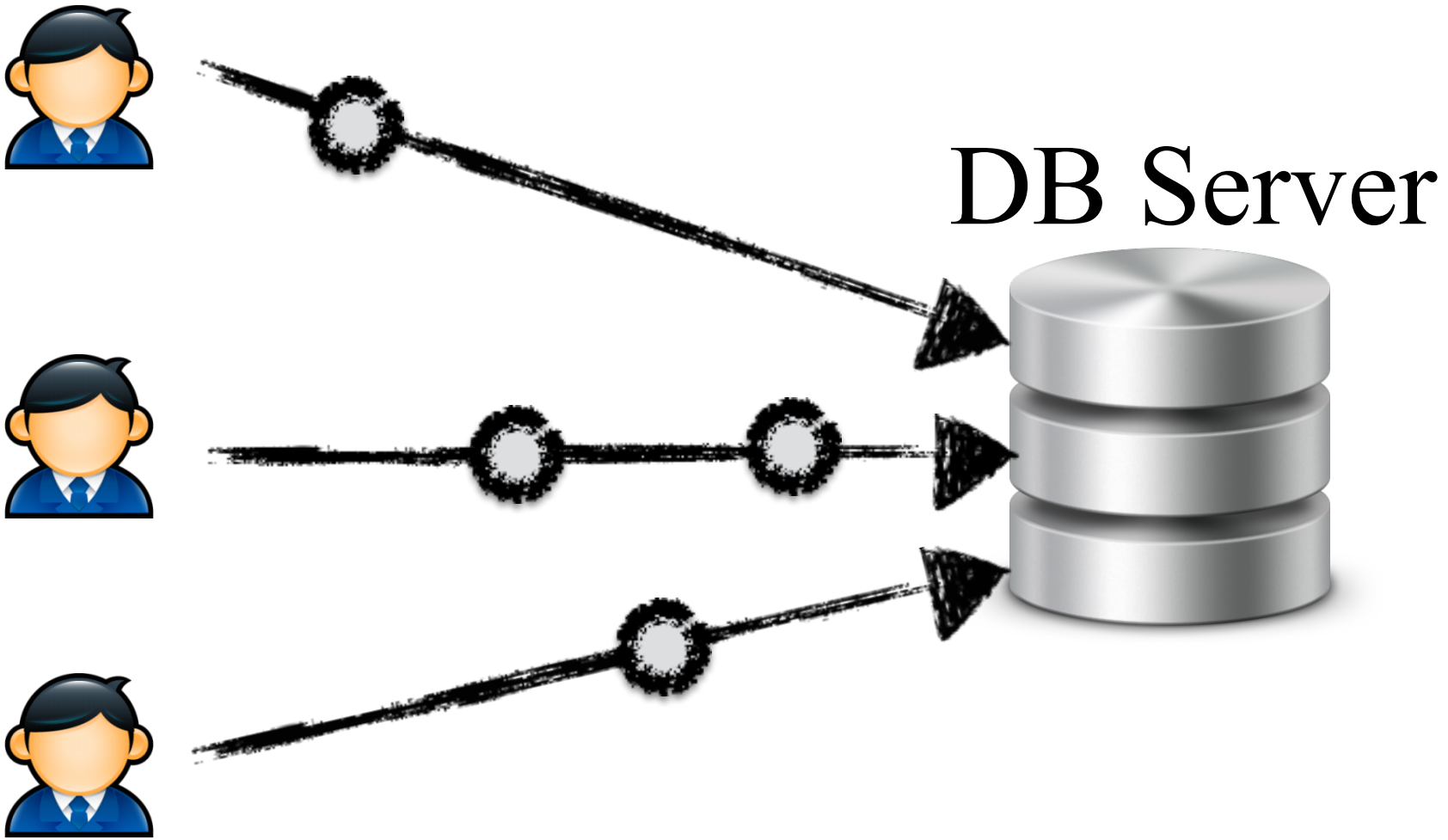
ORACLE

IBM DB2

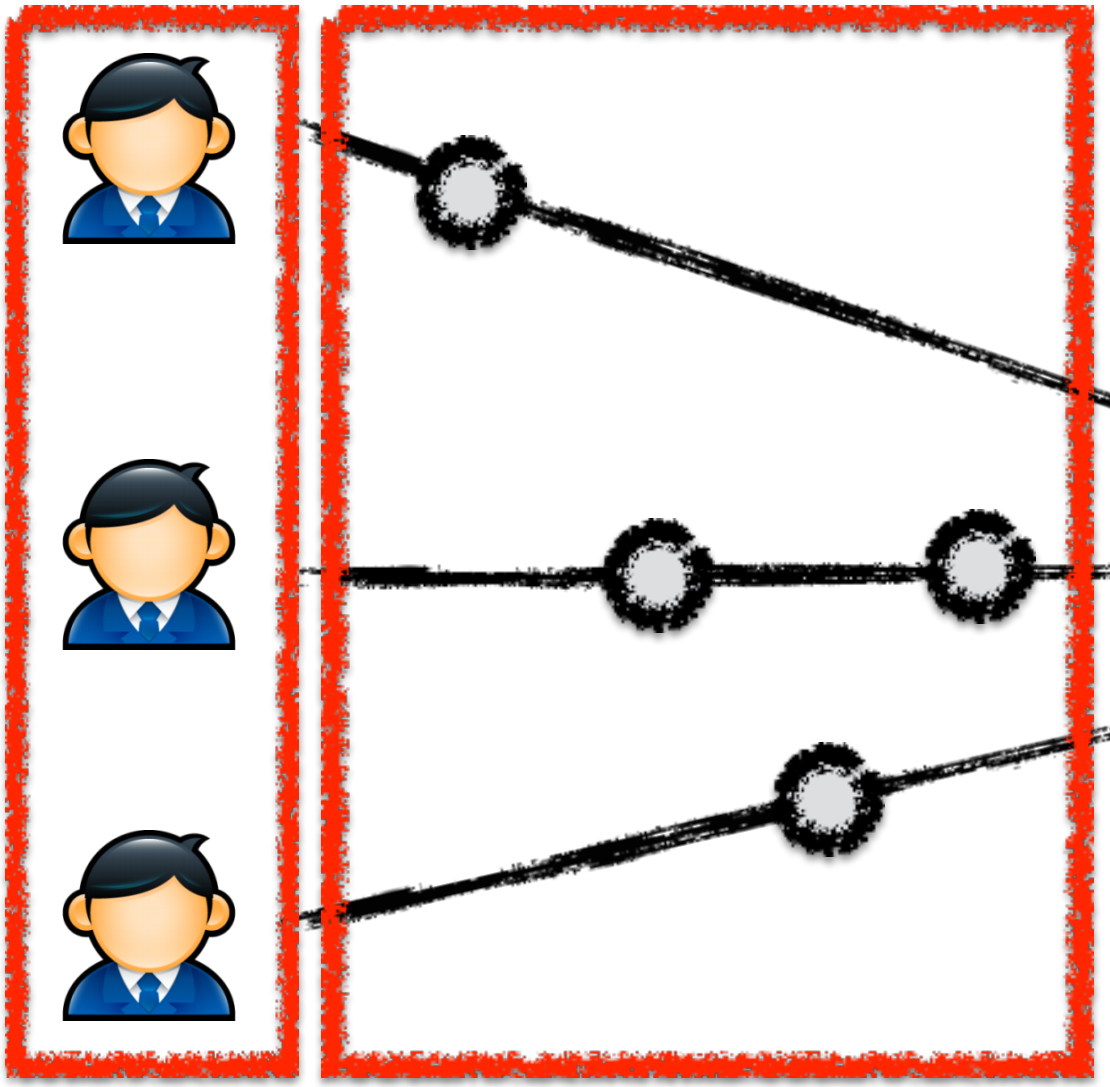
That … was enough ?



Traditional RDBMS: Past



Traditional RDBMS: Past



Few Clients

DB Server

Few tx/sec

Traditional RDBMS: Present

RDBMS is now overloaded !

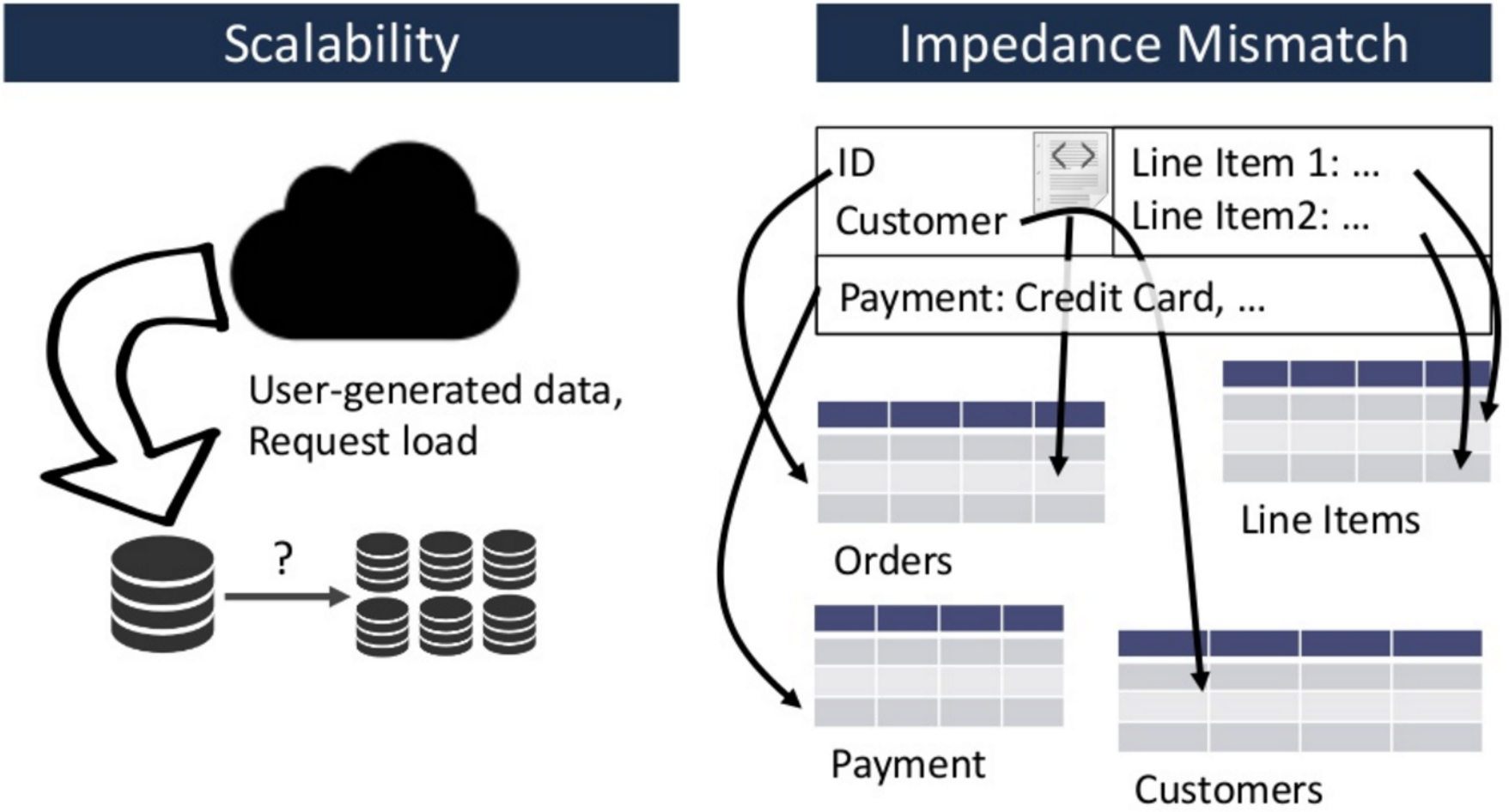
- **Large number of sources**: users, devices, sensors
- **Swarm of transactions**: e.g. 1,000tx/sec/source
- **Huge data size**: e.g. 10MB/sec stream

RDBMS is not agile enough !

Facebook added a new feature: replying with a photo (June 2013)

source: http://www.technixupdate.com/how-to-use-new-facebook-photo-reply-comment-feature/

Main motivations for NoSQL



Characteristics

Key features of NewSQL

- Conform ACID
- High-availability
- Support SQL naturally
- In-Memory Transactions
- Scalable (e.g. horizontal scaling)
- Easy replication through network

Key features of NoSQL

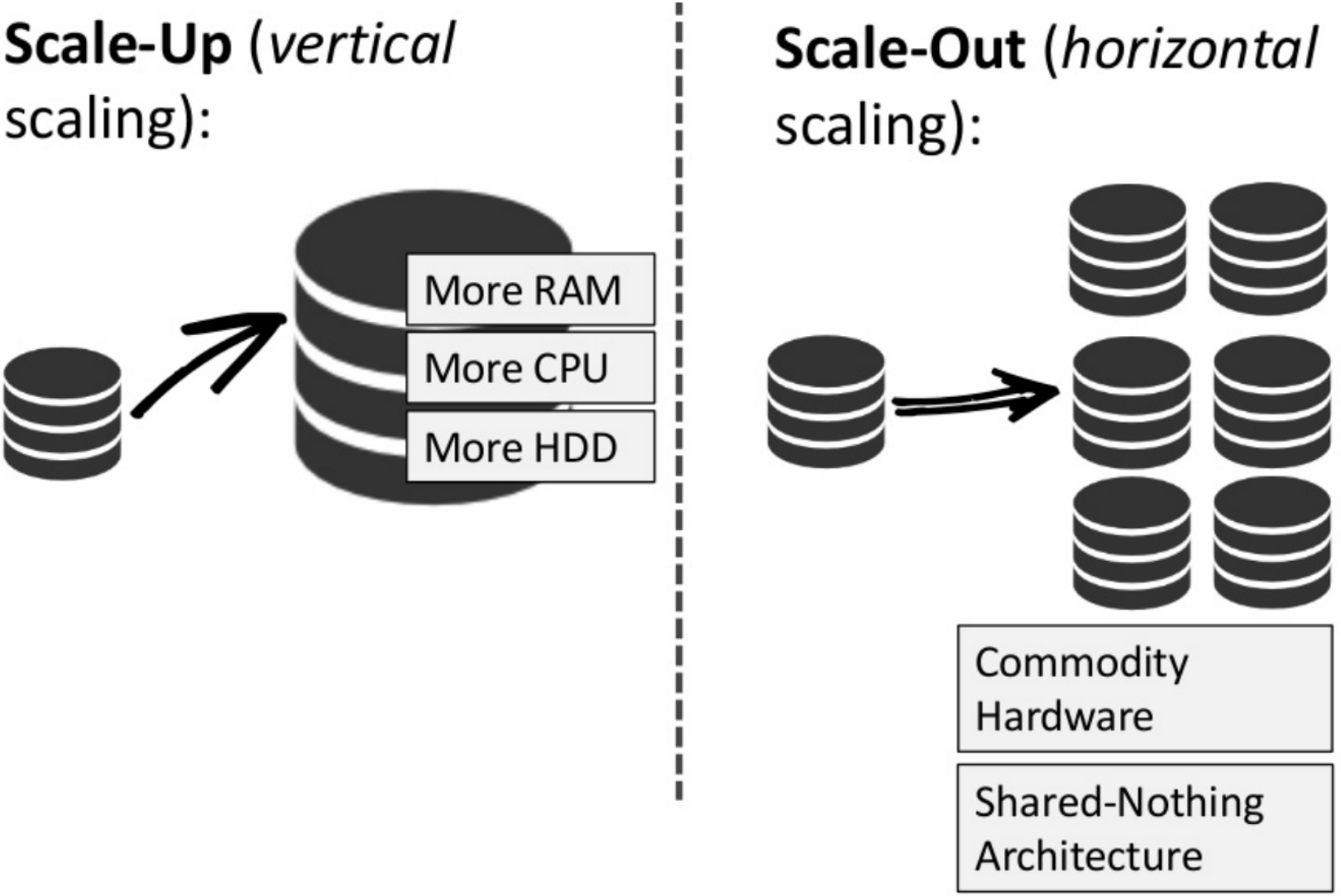
Term coined in 2009 (Not Only SQL)

- Scalable (e.g. horizontal scaling)
- Easy replication through the network
- "Simple" API language (NO query language)
- Tunable consistency model (NO ACID properties)
- Ability to add new attributes to record (NO schema)

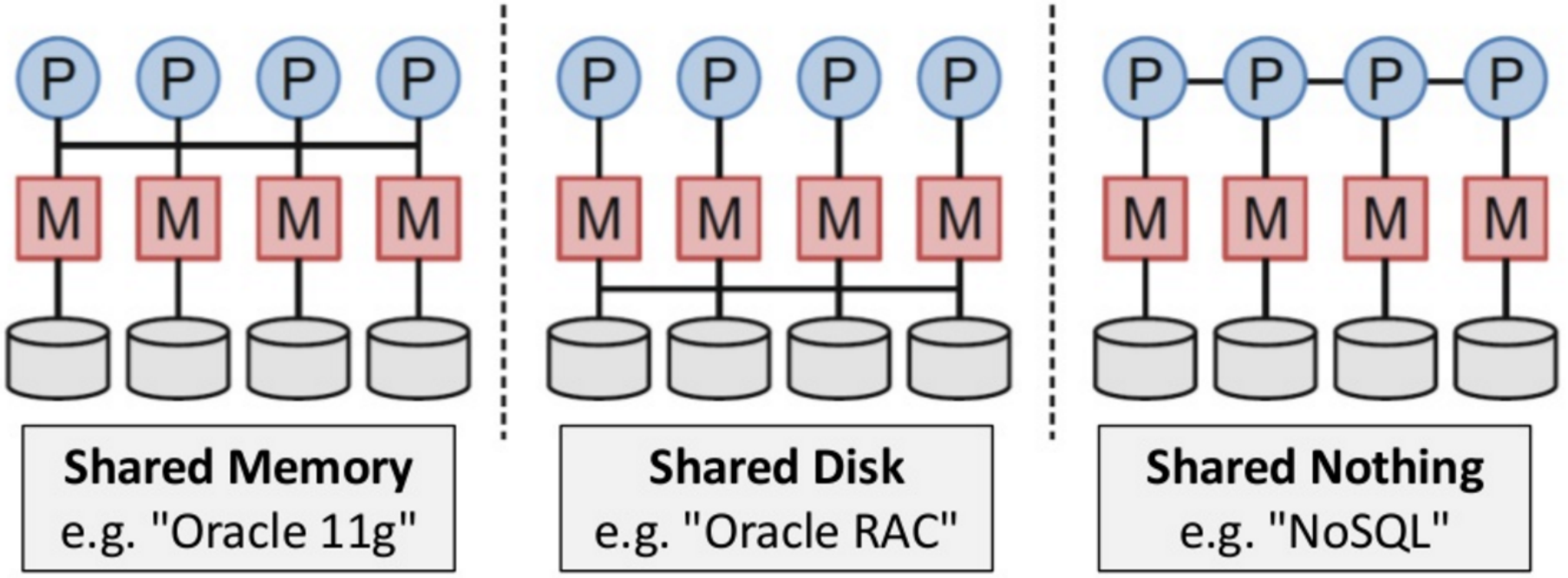
OldSQL vs. NewSQL vs. NoSQL

	OldSQL	NewSQL	NoSQL
ACID/Transaction	✓	✓	✗
Language/Algebra	✓	✓	✗
Schema/Constraints	✓	✓	✗
Scalability	✗	✓	✓ ✓ ✓
In-memory Tx	✗	✓	✓
Easy replication	✗	✓	✓

Vertical vs. Horizontal Scaling



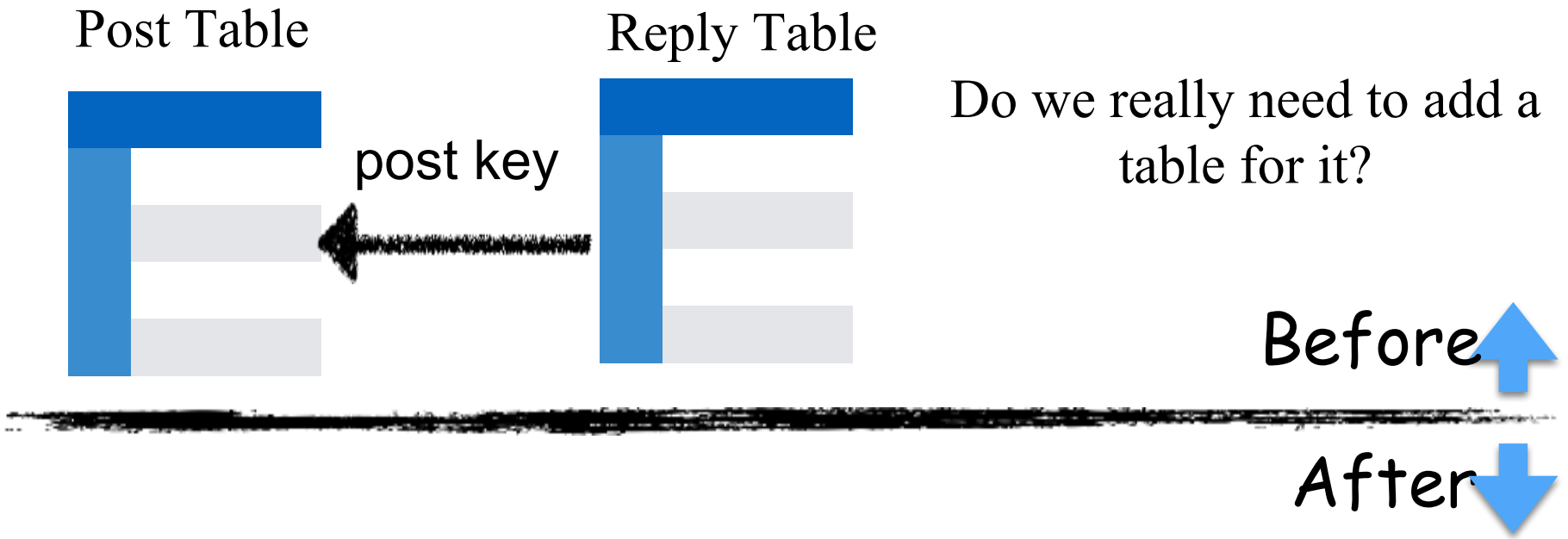
Shift toward “Shared Nothing”



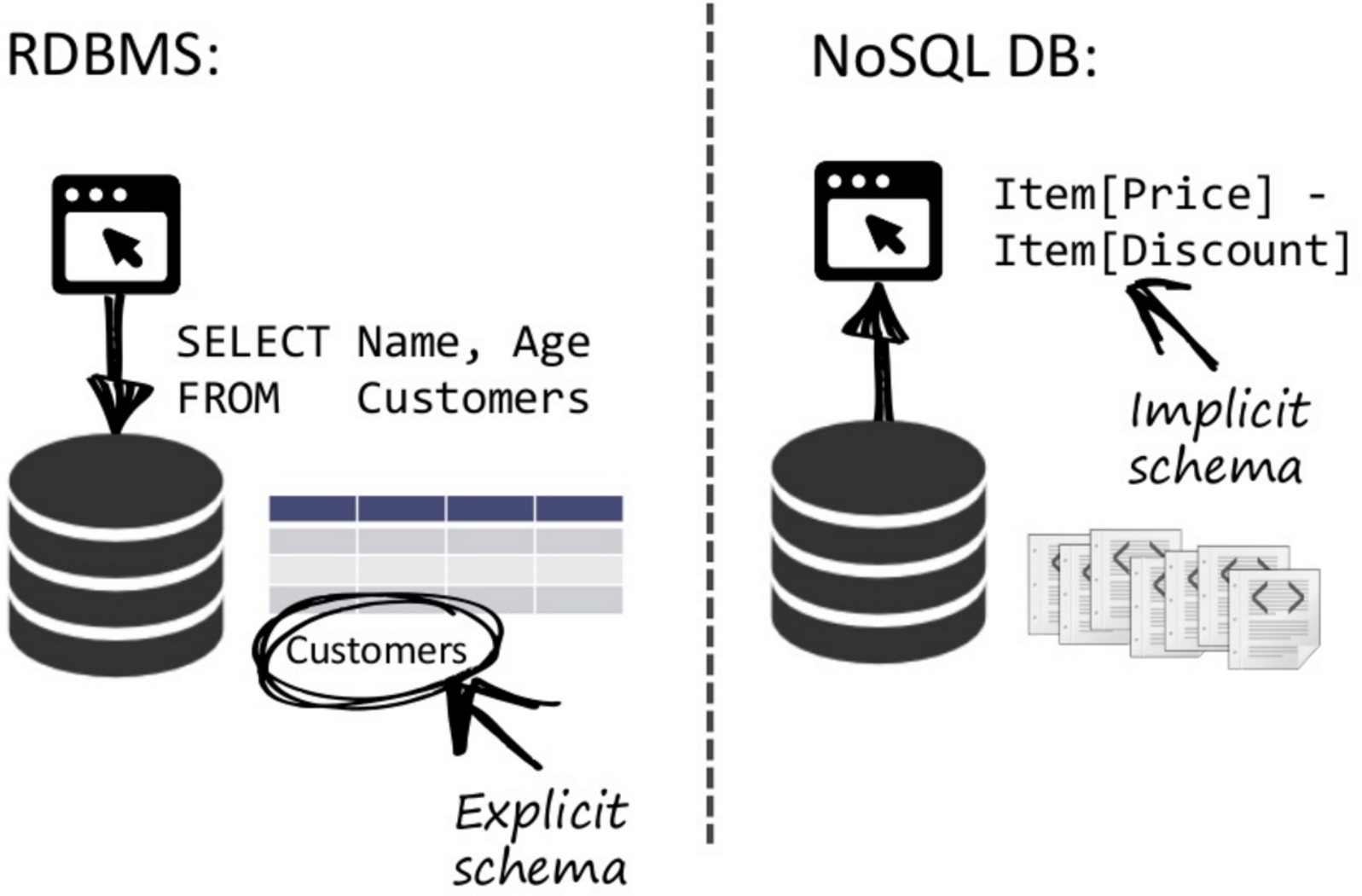
When each node is independent and self-sufficient, there is no single point of contention across the system

the potential for scaling is huge !

What if it is schema-dependent ?



What if it is schema-free ?



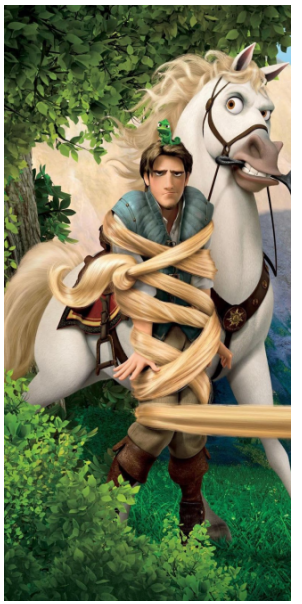
Benefits of schema-free models

Flexibilit
y

Easy to replicate

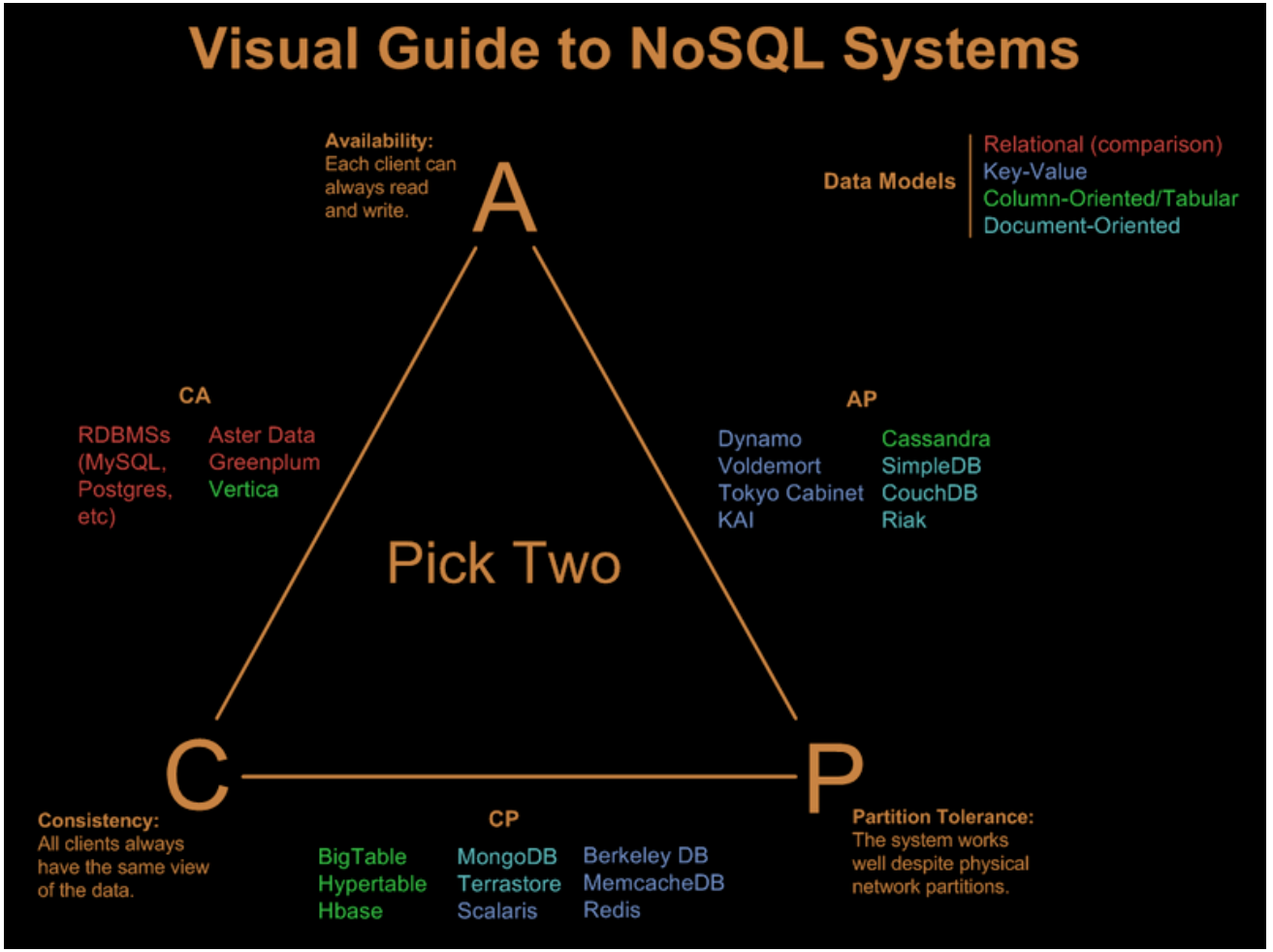
Easy to partition

Join-free



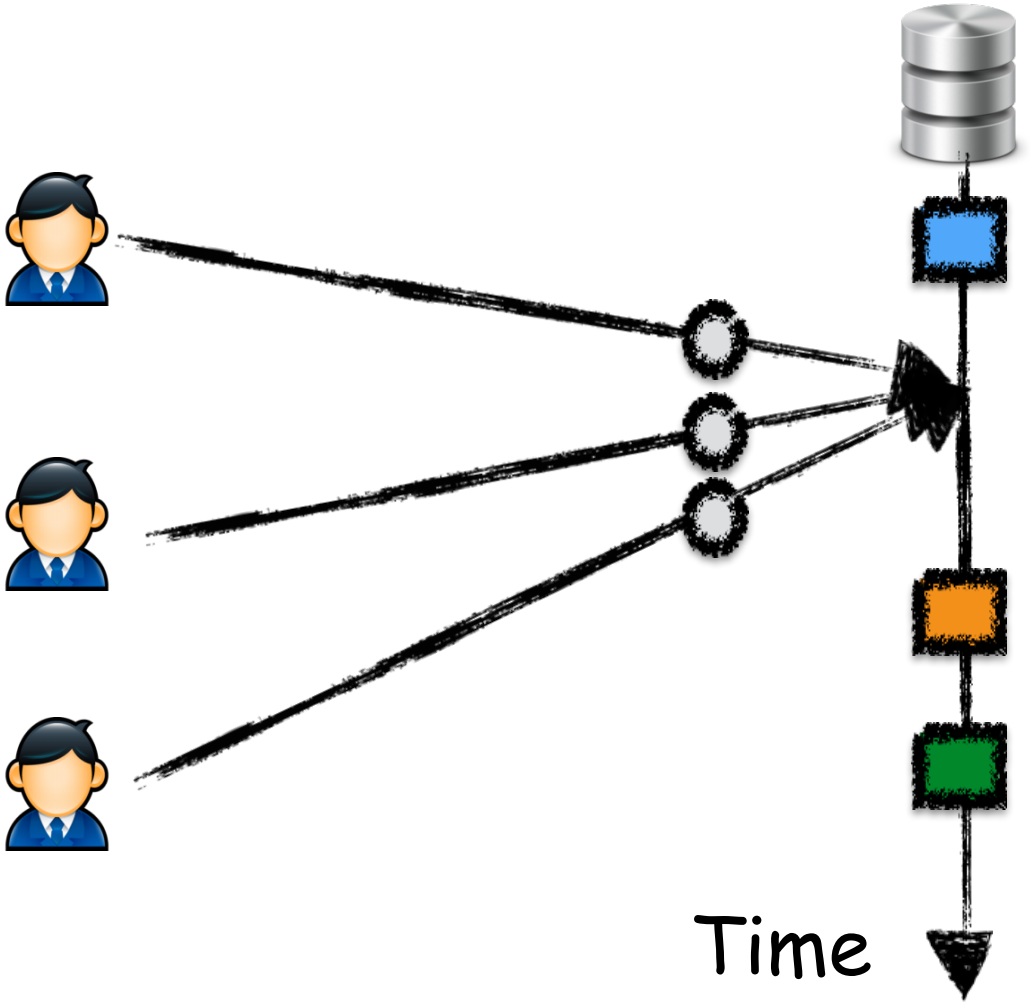
CAP Theorem

CAP Theorem



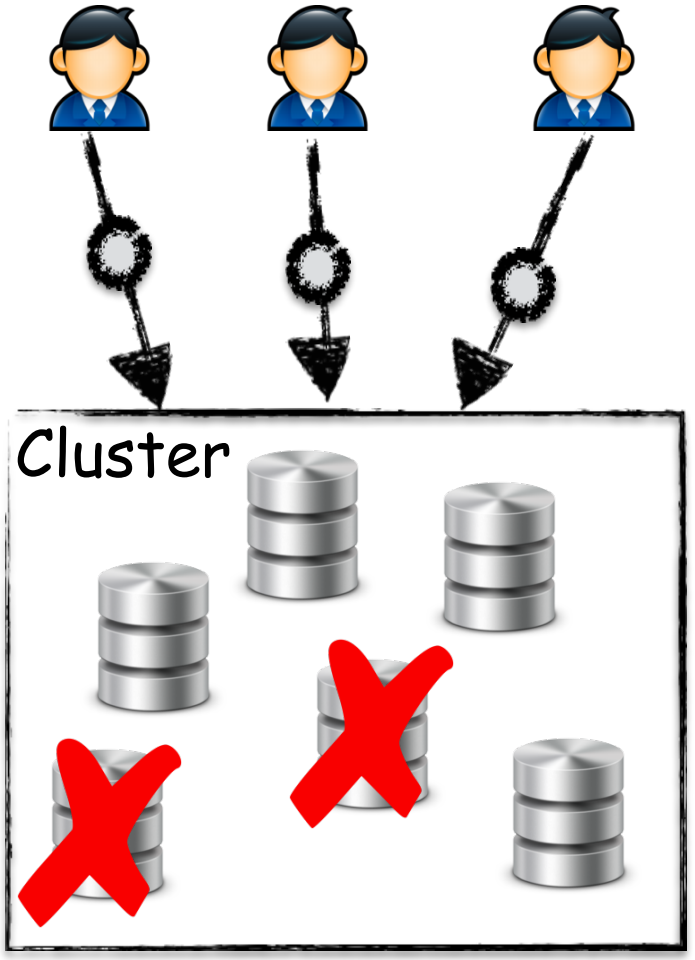
Source: http://blog.nahurst.com/visual-guide-to-nosql-systems

Consistency



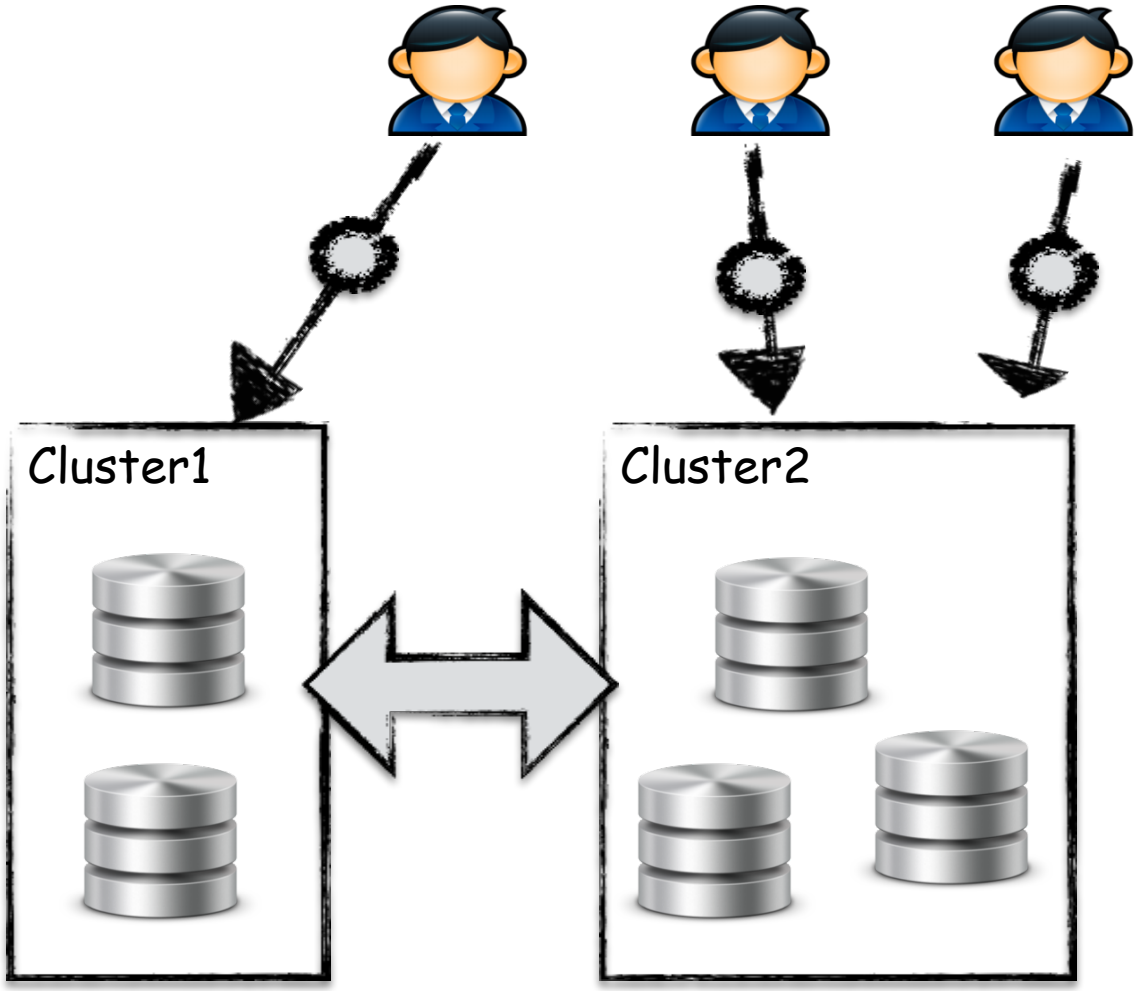
All clients have the same view on the data

Availability



Every request to a non-failed node must result in a correct response

Partition tolerance

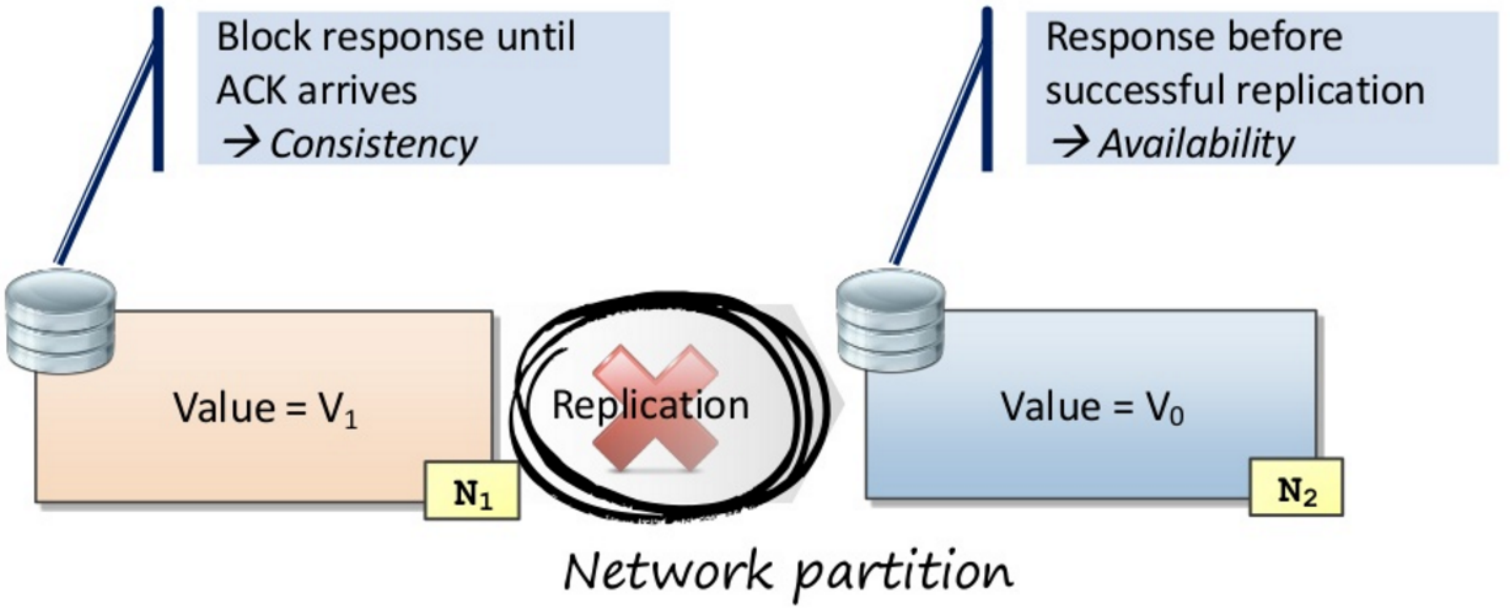


The system has to continue working, even under arbitrary network partitions

Simplified proof

When a network partition occurs,

either consistency or availability has to give up

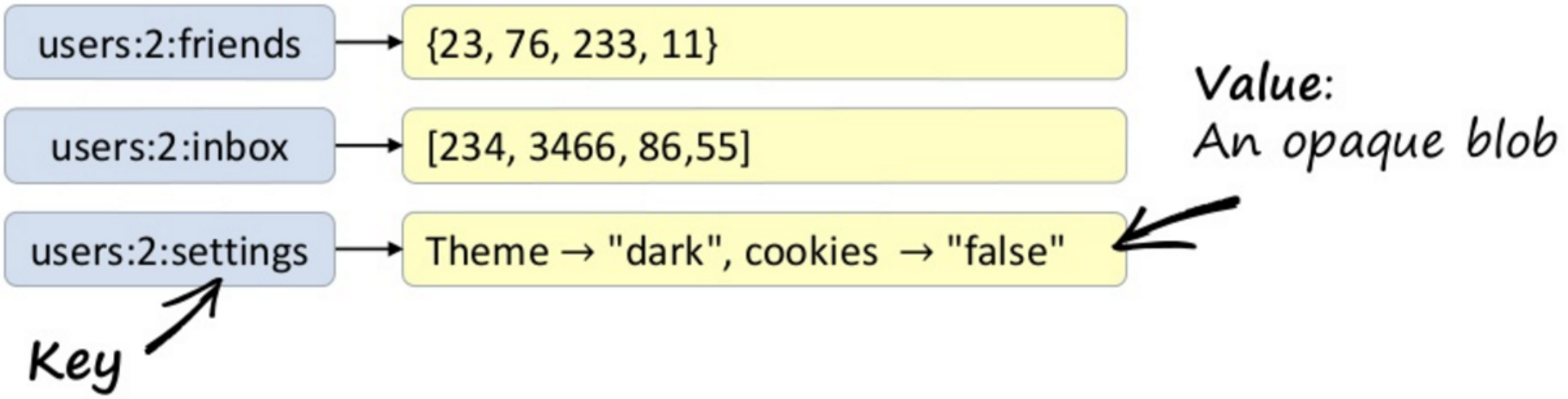


Database Landscape

NoSQL Landscape

Data Model: Key-Value

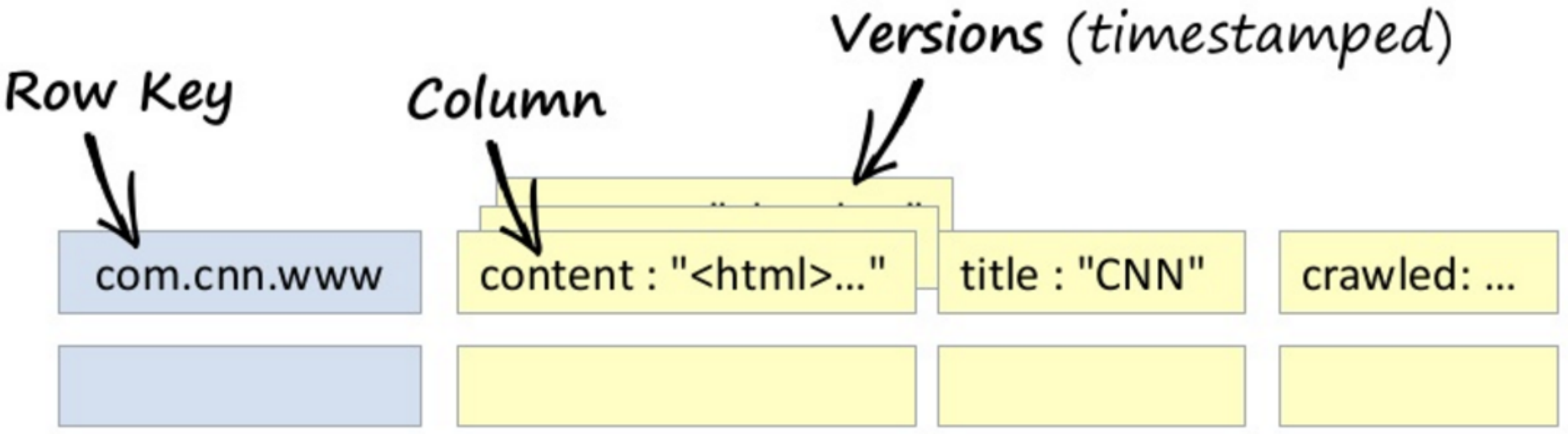
single (key) -> value, no foreign key/value indices
appropriate for instant access to data (shop basket)



Example: Dynamo (AP), Riak (AP), Redis (CP)

Data Model: Wide-Column

data indexed by a 3D map: (row, col, time) -> val
appropriate for aggregation and analytics (reports)

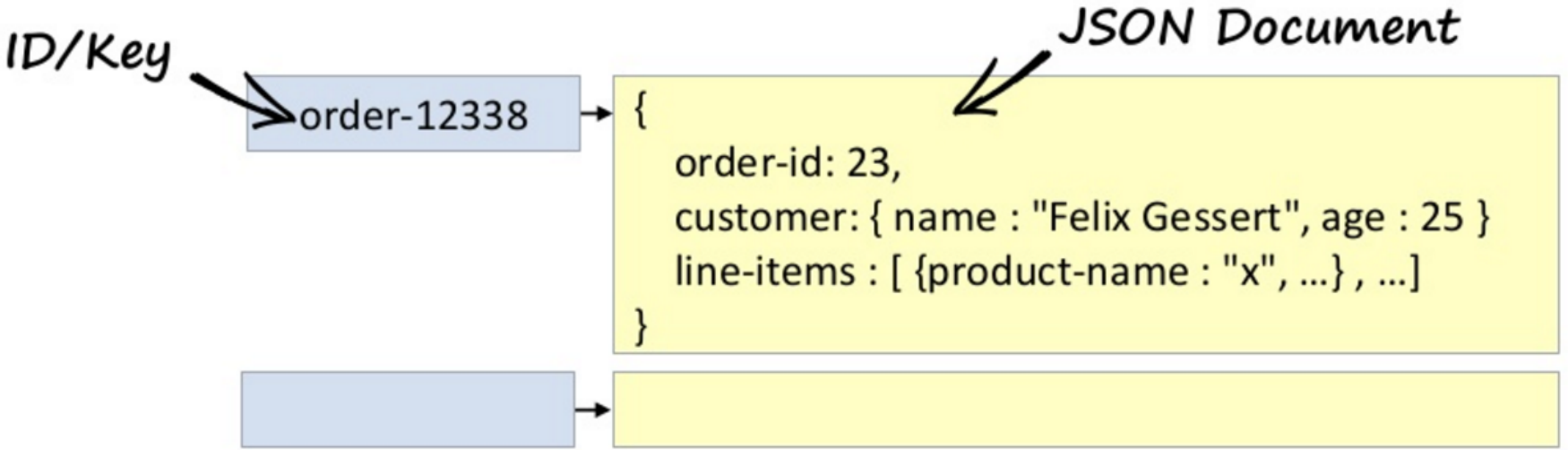


Example: Cassandra (AP), BigTable (CP), HBase (CP)

Data Model: Document

similar to JSON/XML: (coll, key) -> doc

appropriate for data not frequently changing (logs)

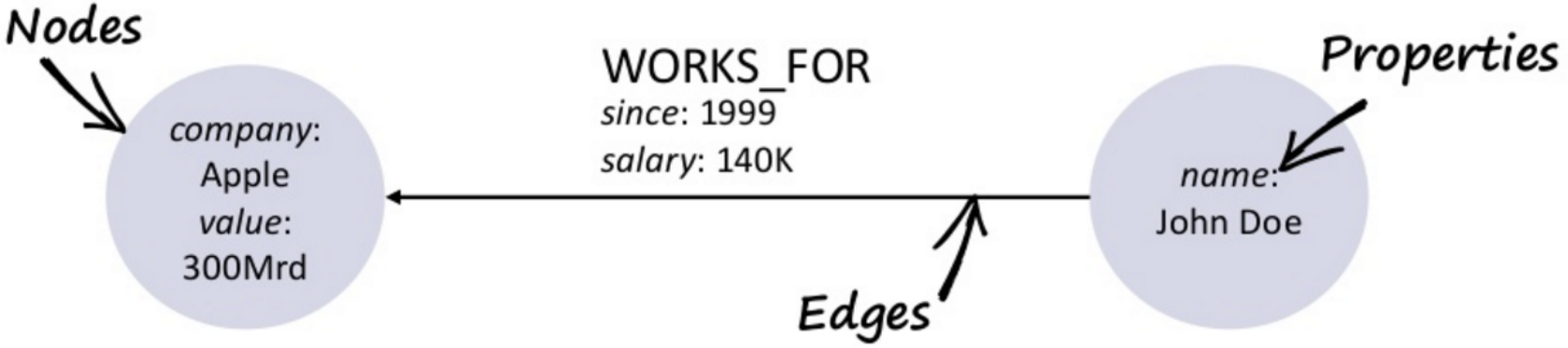


Example: CouchDB (AP), SimpleDB (AP), MongoDB (CP)

Data Model: Graph

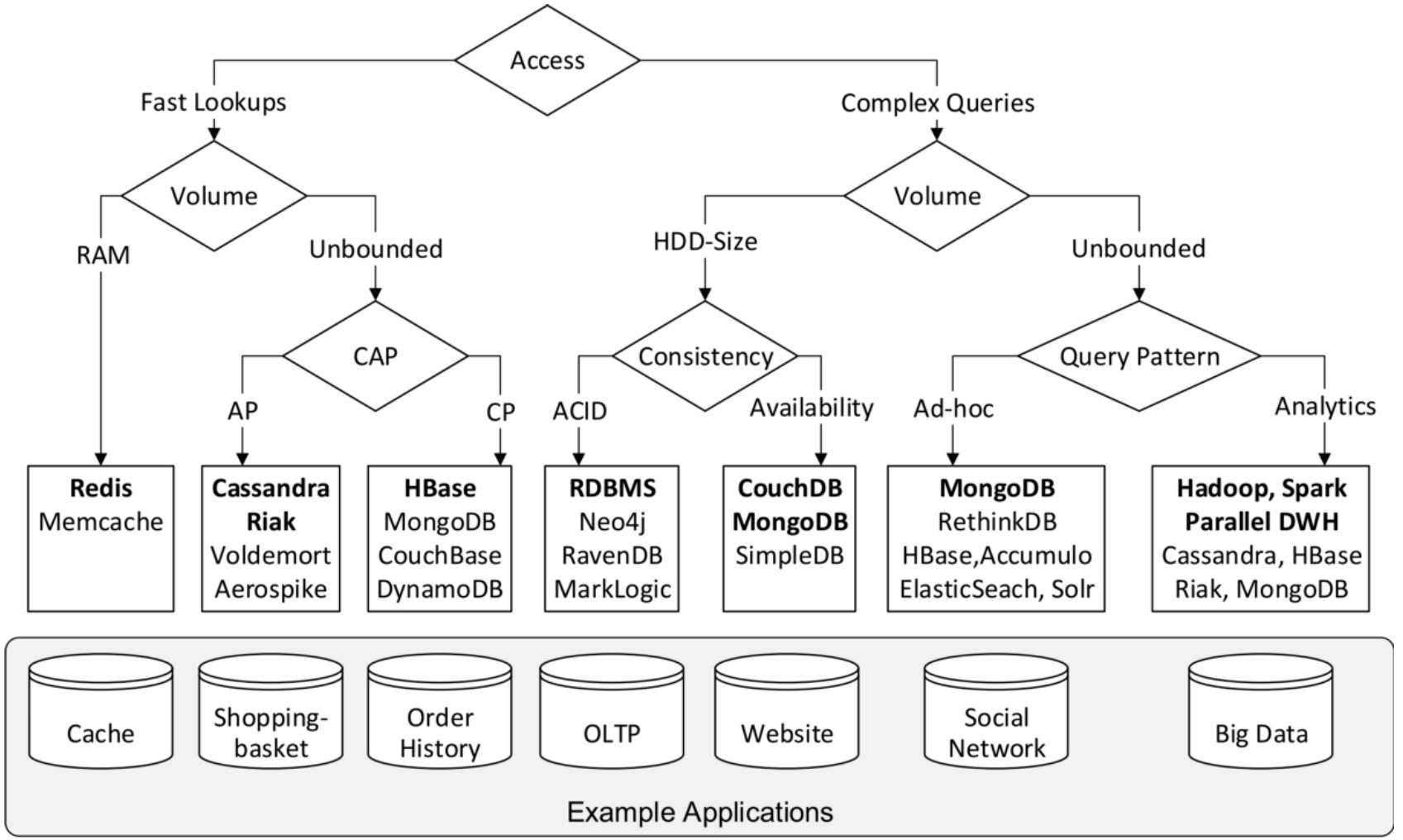
data represented by vertices/edges: G = (V, E)

appropriate for complex structures (social network)



Example: Neo4j (CA), InfiniteGraph (CA), OrientDB (CA)

Decision tree for mapping requirements



Source: https://medium.baqend.com/nosql-databases-a-survey-and-decision-guidance-ea7823a822d

Database Ranking

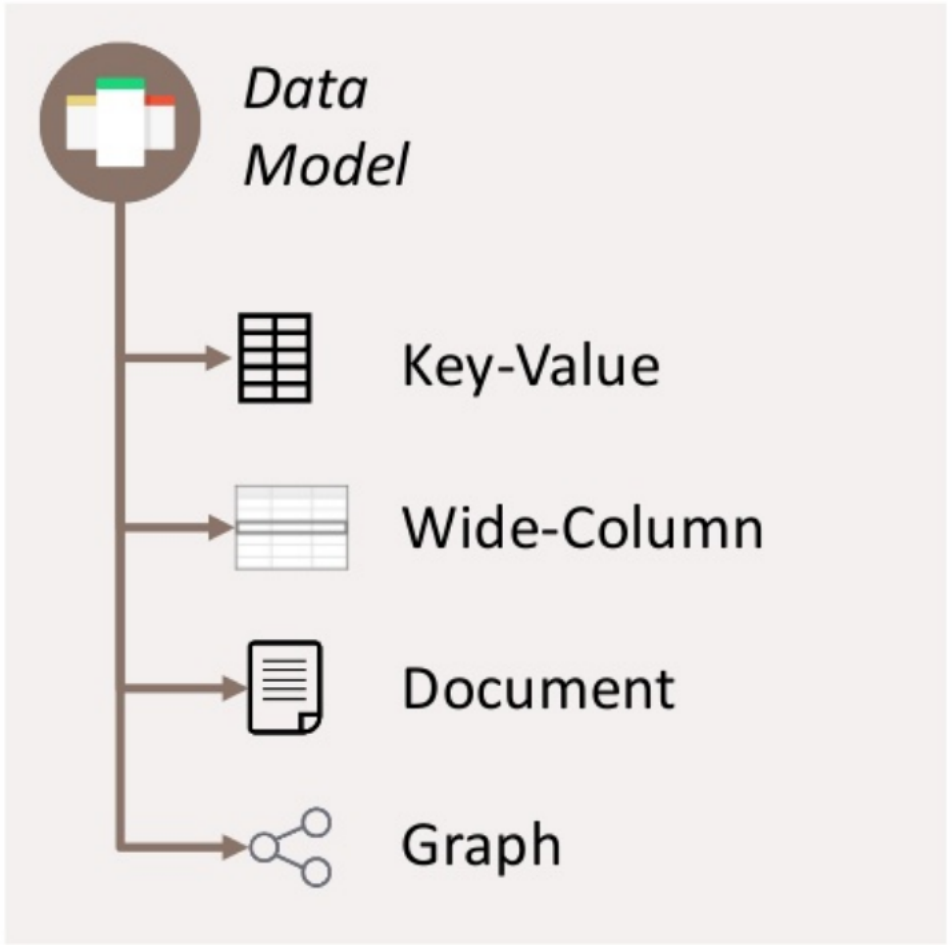
http://db-engines.com/en/ranking

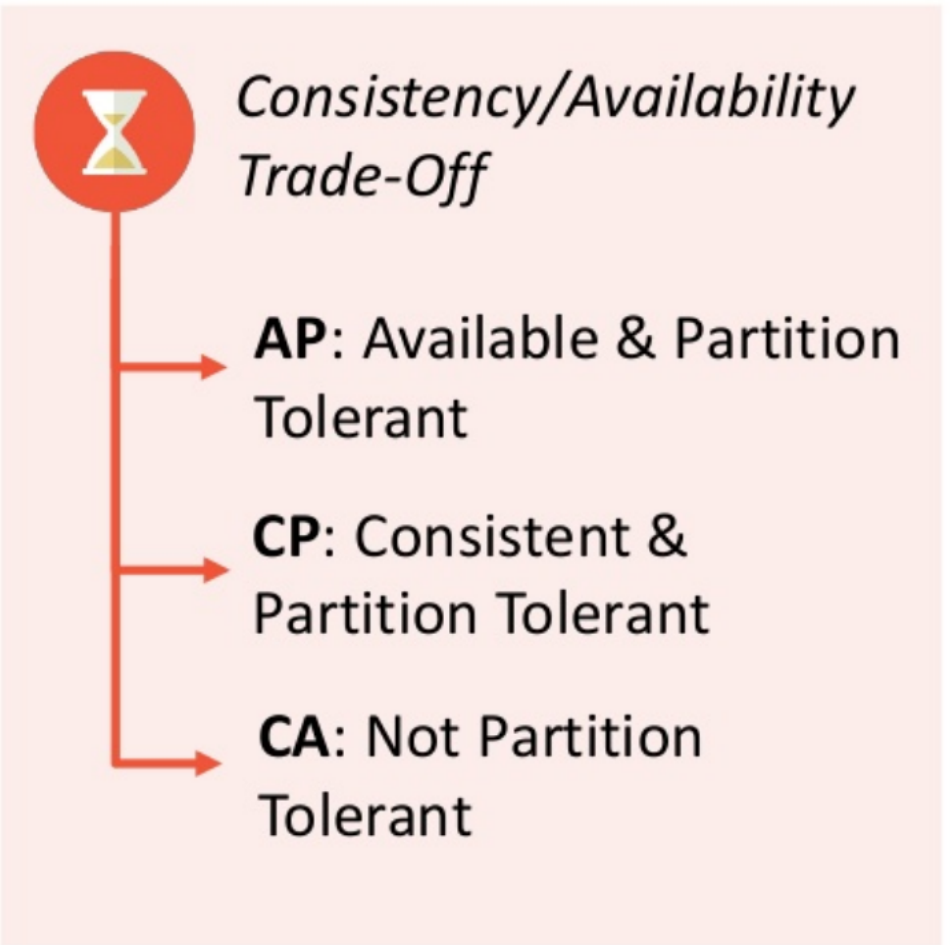
316 systems in ranking, October 2016

Rank Oct 2016	Rank Sep 2016	Rank Oct 2015	DBMS	Database Model	Score Oct 2016	Score Sep 2016	Score Oct 2015
1.	1.	1.	Oracle	Relational DBMS	1417.10	-8.46	-49.85
2.	2.	2.	MySQL	Relational DBMS	1362.65	+8.62	+83.69
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1214.18	+2.62	+90.95
4.	↑ 5.	4.	MongoDB	Document store	318.80	+2.81	+25.54
5.	↓ 4.	5.	PostgreSQL	Relational DBMS	318.69	+2.34	+36.56
6.	6.	6.	DB2	Relational DBMS	180.56	-0.62	-26.25
7.	7.	↑ 8.	Cassandra	Wide column store	135.06	+4.57	+6.05
8.	8.	↓ 7.	Microsoft Access	Relational DBMS	124.68	+1.36	-17.16
9.	↑ 10.	↑ 10.	Redis	Key-value store	109.54	+1.75	+10.75
10.	↓ 9.	↓ 9.	SQLite	Relational DBMS	108.57	-0.05	+5.90

Summary

The era of one-size-fits-all database systems is over !





Example: MongoDB

MongoDB in a nutshell

- Created by MongoDB Inc. in Oct. 2007
  - MongoDB <= huMONGOus DB
- Document-oriented data model
  - BSON (similar to JSON)
- Supports:
  - indexing
  - replication
  - load balancing
  - server-side JavaScript execution.

Insert/Update/Delete

- **Insert**: db.<collection>.insert(<document>)
  - db.users.insert({name:”john”,city:”Luxembourg”})
  - no need to create a table upfront !

- **Update**: db.<collection>.update(<query>, <values>)
  - db.users.update({name:”john”}, {$set: {city:”Seoul”}})

- **Delete**: db.<collection>.remove(<query>)
  - db.users.remove({name:”john”})

Select/Queries

- **Simple Select**: db.<collection>.find(<query>)
  - db.users.find({name:”john”})
- **Nested Select**: use dot-notation
  - db.users.find({"score.math":30})
- **Show specific columns only**
  - db.users.find({},{“score.math”:1, “score.lang”:1})
- **Query if to test if exists**
  - db.users.find({"score.math":{$exists:true}}, {name:1})

Other useful operations

- **count**(): counting the number of documents
  - db.users.find().count()
  - db.users.count({name:”john”})
- **sort**(): sorting documents
  - db.users.find().sort({name:-1})
- **List** collections (tables) and databases
  - show collections/tables ; show databases
- **limit**(): Show a specified number of documents
  - db.users.find().limit(5)