Advanced Database

Course Code - ST2BDA

Specialty : IT

Course Credits: Jean-Charles Huet & Hanen OCHI

Dr. Lilia Sfaxi
Assistant professor

CM3 –NOSQL Databases

Databases for IT Engineers

Dr. Lilia Sfaxi
Assistant Professor

01

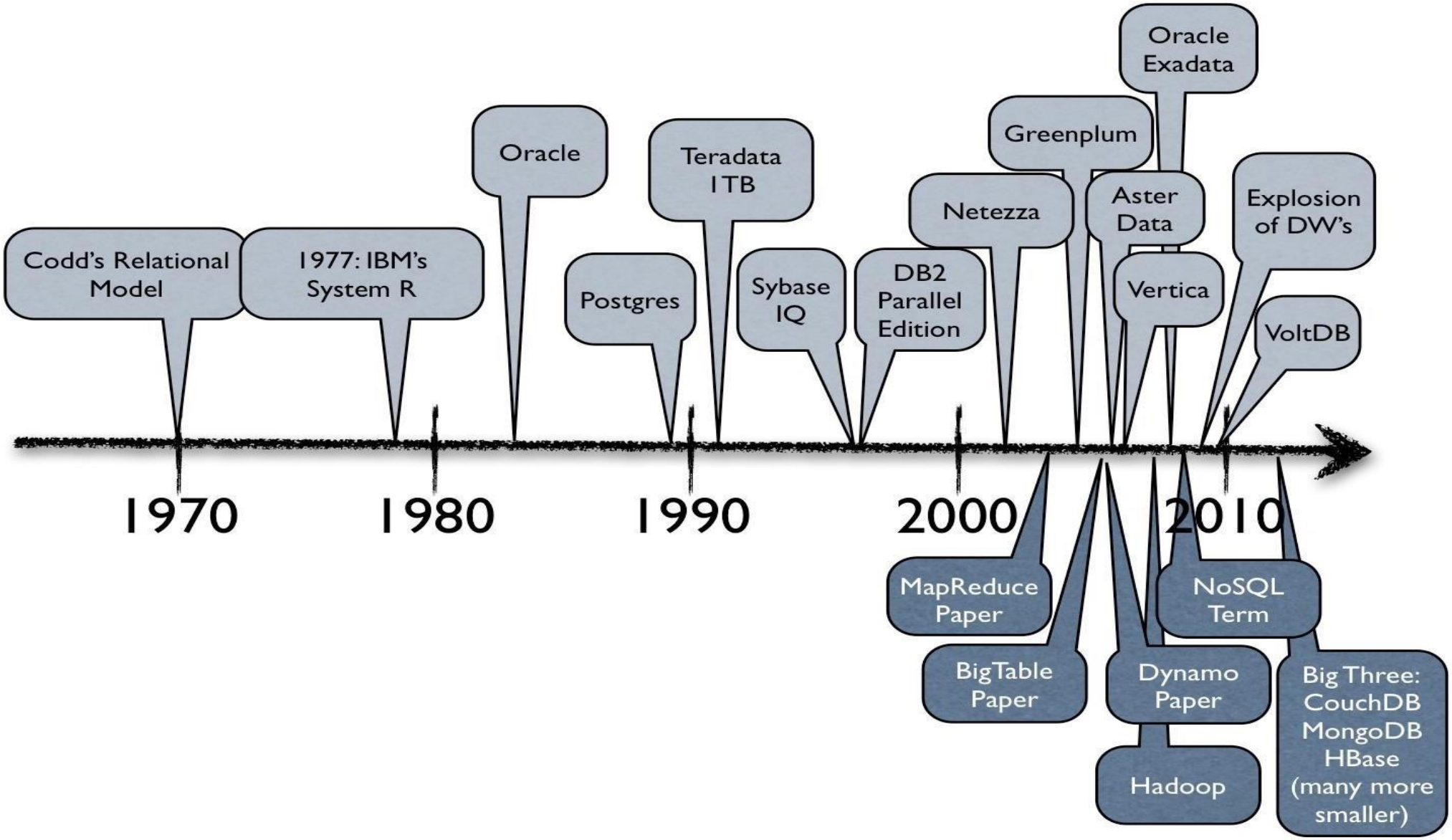
Introduction

01

Introduction to NOSQL

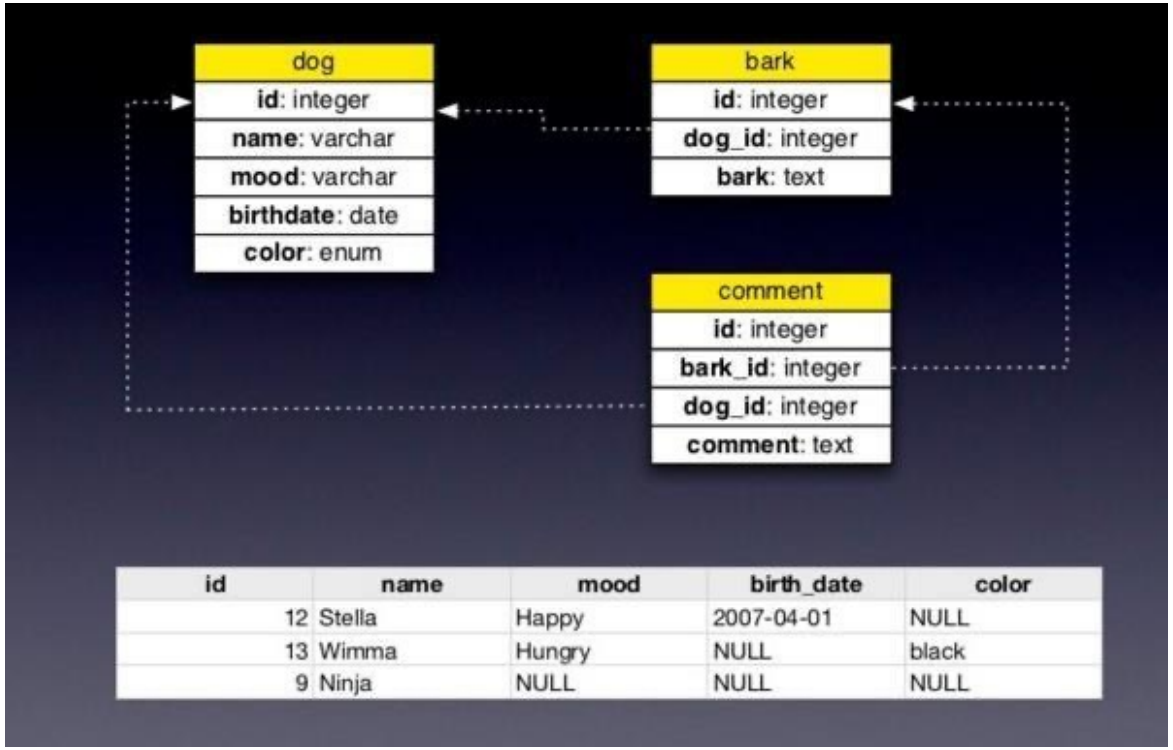
- Database:
  - Organized collection of data
- DBMS: Database Management System:
  - Software package with computer programs that controls the creation, maintenance and use of a database
- Databases are created to operate large quantities of information by inputting, storing, retrieving, and managing that information

History


Codd's Relational Model
1977: IBM's System R
Oracle
Postgres
Teradata 1TB
Sybase IQ
DB2 Parallel Edition
Netezza
Greenplum
Oracle Exadata
Aster Data
Vertica
Explosion of DW's
VoltDB
1970
1980
1990
2000
2010
MapReduce Paper
BigTable Paper
Dynamo Paper
Hadoop
NoSQL Term
Big Three: CouchDB MongoDB HBase (many more smaller)

Benefits of Relational databases:

- Designed for all purposes
- Strong consistency, concurrency, recovery
- Mathematical background
- Standard Query language (SQL)
- Lots of tools to use with i.e: Reporting services, entity frameworks, …

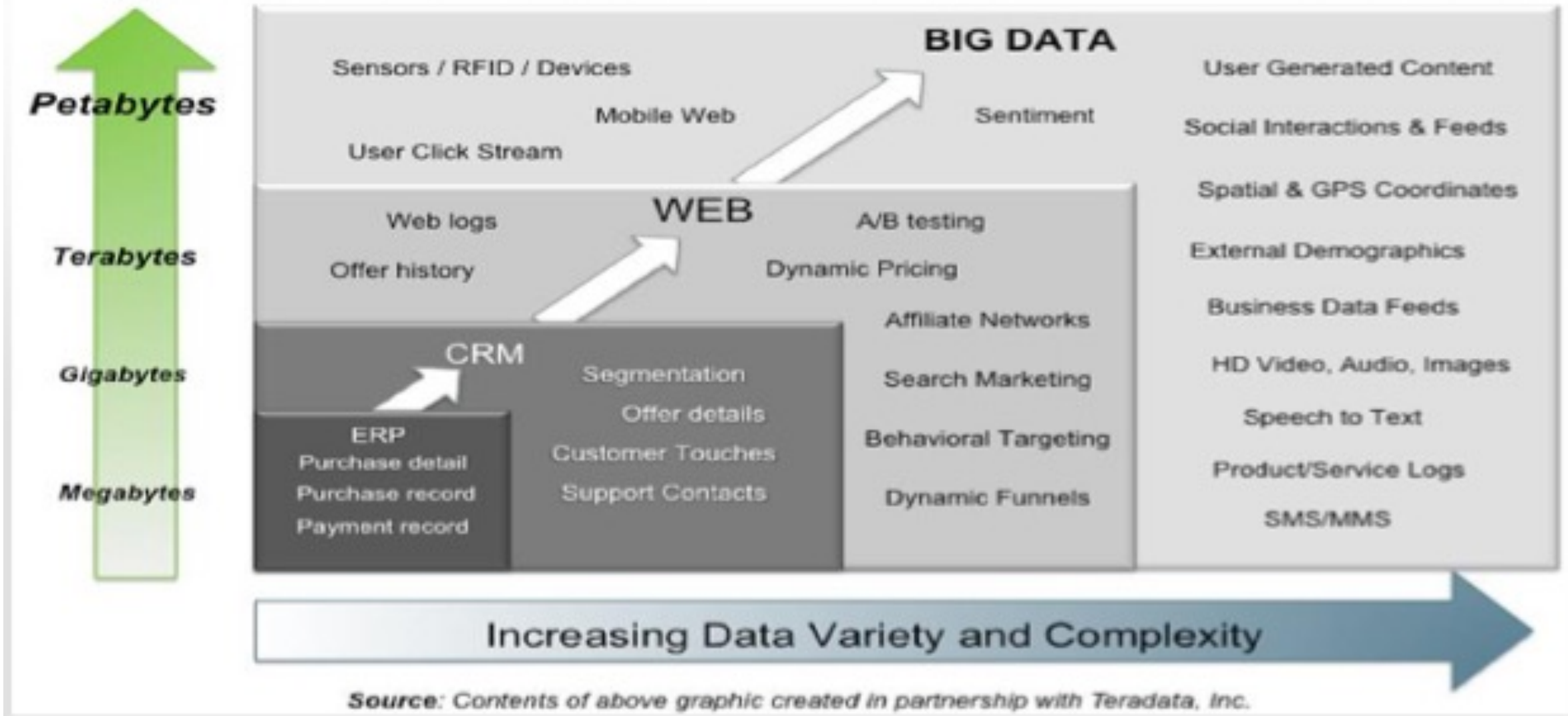


id	name	mood	birth_date	color
12	Stella	Happy	2007-04-01	NULL
13	Wimma	Hungry	NULL	black
9	Ninja	NULL	NULL	NULL

Applications and Web Platforms

- Exponential growth of the amount of Data (x2 /2 years)
- Unprecedent management of this volume
  - Need to distribute both computation and data
  - Huge number of servers
  - Heterogeneous data, maybe complex and oftenlinked
- Examples :
  - Google, Amazon, Facebook
  - Google DataCenter: 5000 servers/data center, ~1M de servers
  - Facebook : 1 PetaBytes ofdata

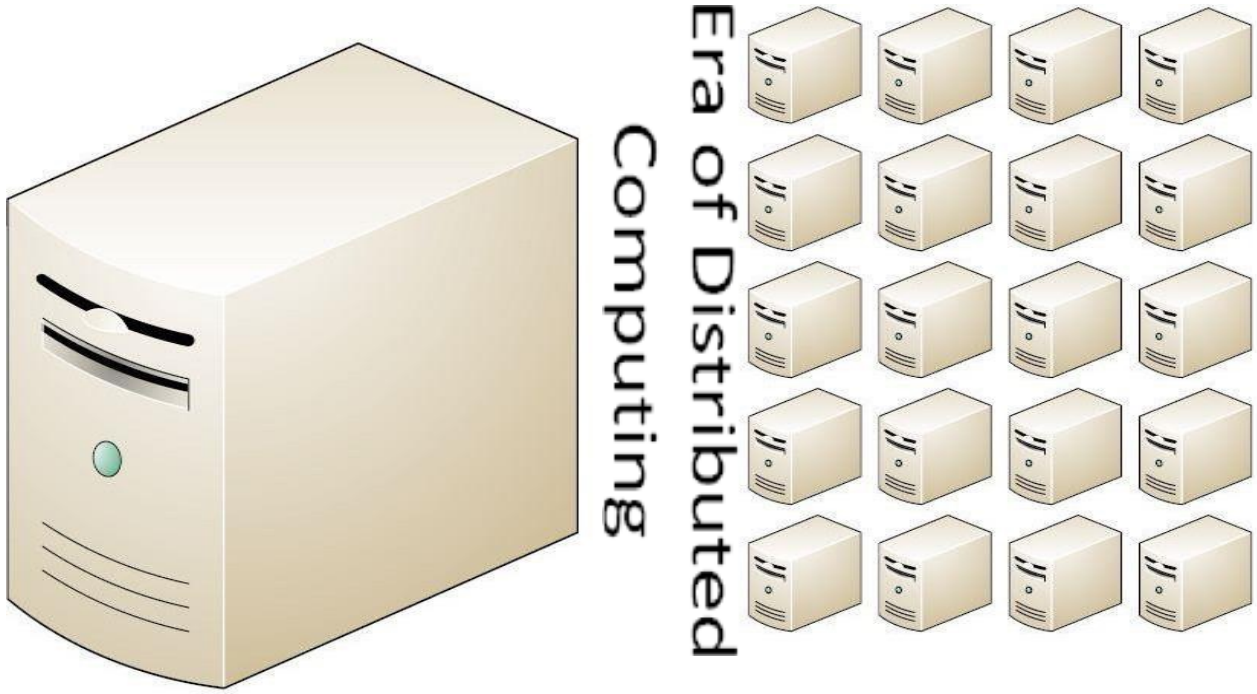
Big Data = Transactions + Interactions + Observations



***Source**: Contents of above graphic created in partnership with Teradata, Inc.*

NOSQL: Why, What and When?

- But ..
  - Relational databases were not built for distributed applications.
- Because ..
  - Joins are expensive
  - Hard to scale horizontally
  - Impedance mismatch occurs
  - Expensive (product cost, hardware, Maintenance)
- And ..
  - When distributed, it's weak in:
    - Speed (performance)
    - High availability
    - Partition tolerance



What is NOSQL?

- A NOSQL database provides a mechanism for storage and retrieval of data that employs less constrained consistency models than traditional relational database

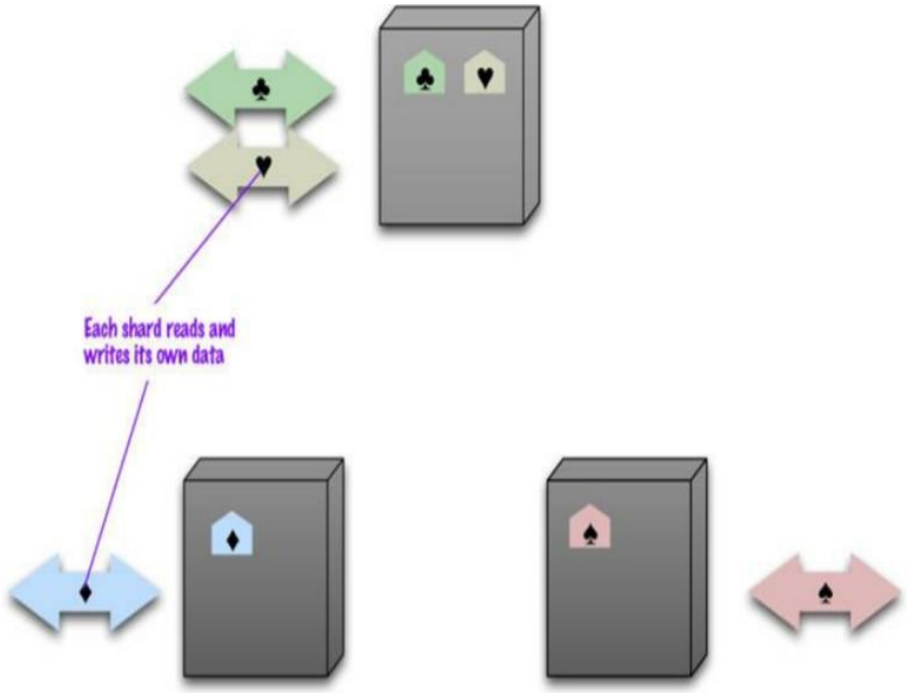
- NOSQL stands for "Not Only SQL“ to emphasize that languages other than SQL-like ones are used to query databases.
  - A more accurate name would be “Non Relational Databases”

Data Distribution

- NoSQL systems: data distributed over large clusters
- Data distribution models:
  - Single server (is an option for some applications)
  - Multiple servers
- Orthogonal aspects of data distribution:
  - Sharding: different data on different nodes
  - Replication: the same data copied over multiple nodes

Sharding of Data

- Different parts of the data onto different servers
  - Horizontal scalability
  - Ideal case: different users all talking to different server nodes
  - Data accessed together on the same node – aggregate unit!



(+) it can improve both reads and writes

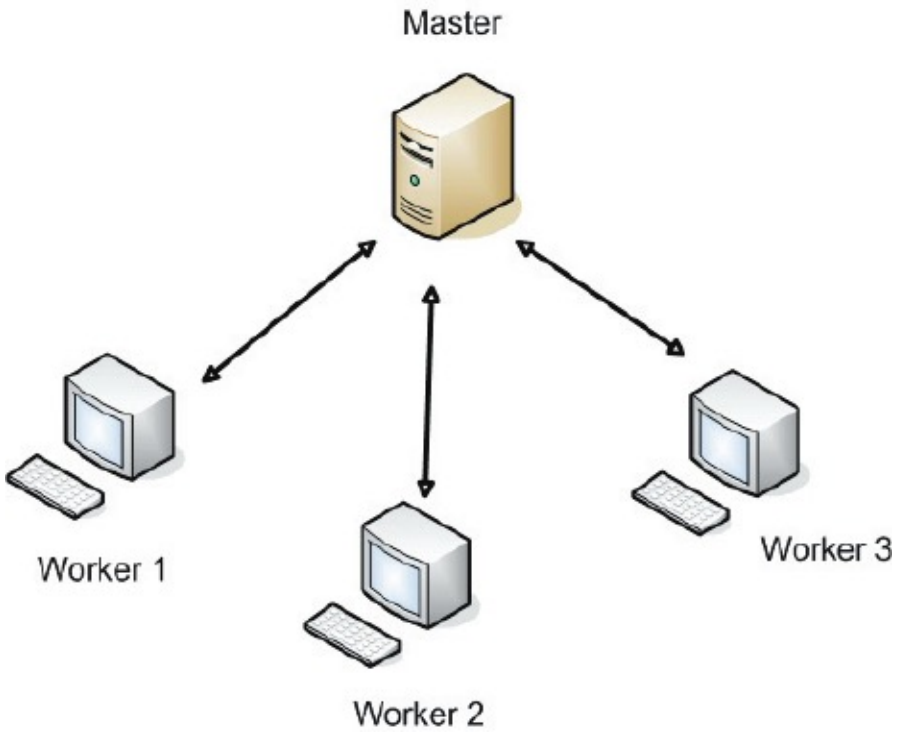
(–) Clusters use many, and sometimes less reliable, machines : resilience decreases

Improving Performance

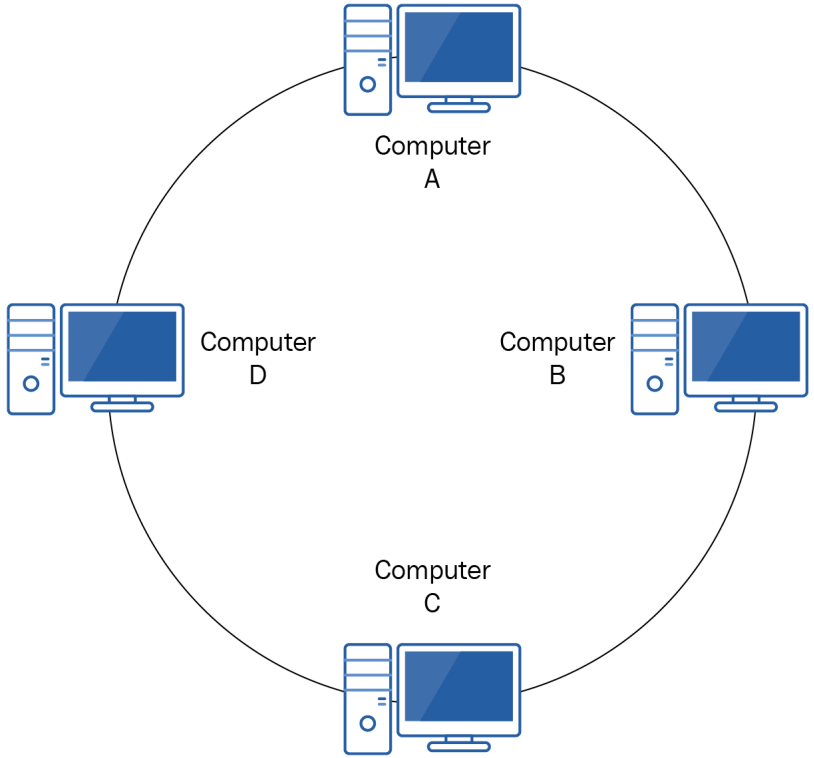
- Main rules of sharding:
  1. Place the data close to where it's accessed
     - Orders for Boston: data in your eastern US data center
  2. Try to keep the load even
     - All nodes should get equivalent amounts of the load
  3. Put together data that may be read in sequence
     - Same order, same node
- Most NoSQL databases offer auto-sharding
  - The database takes on the responsibility of sharding

Data Replication

- Comes in two forms



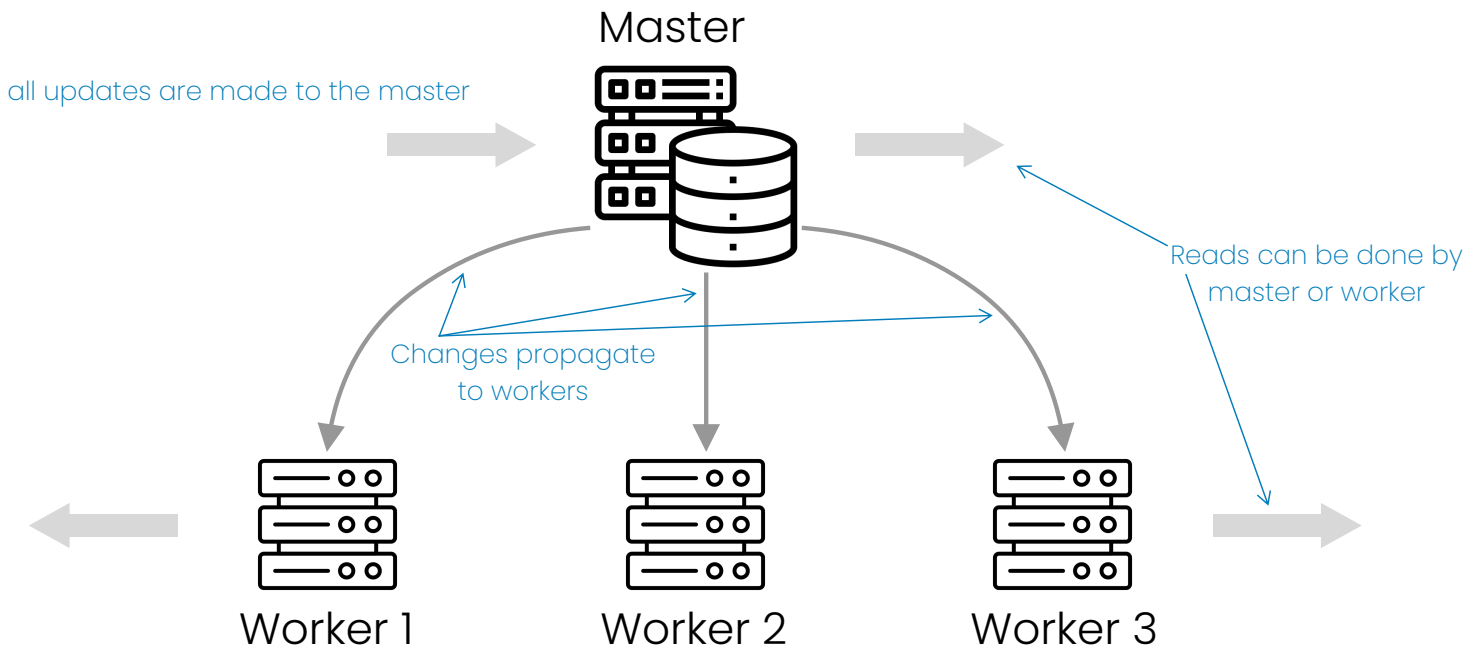
Master-Worker



Peer to Peer (Ring Architecture)

Master-Worker Replication

- Master
  - is the authoritative source for the data
  - is responsible for processing any updates to that data
  - can be appointed manually or Automatically
- Workers
  - A replication process synchronizes the workers with the master
  - After a failure of the master, a worker can be appointed as new master very quickly

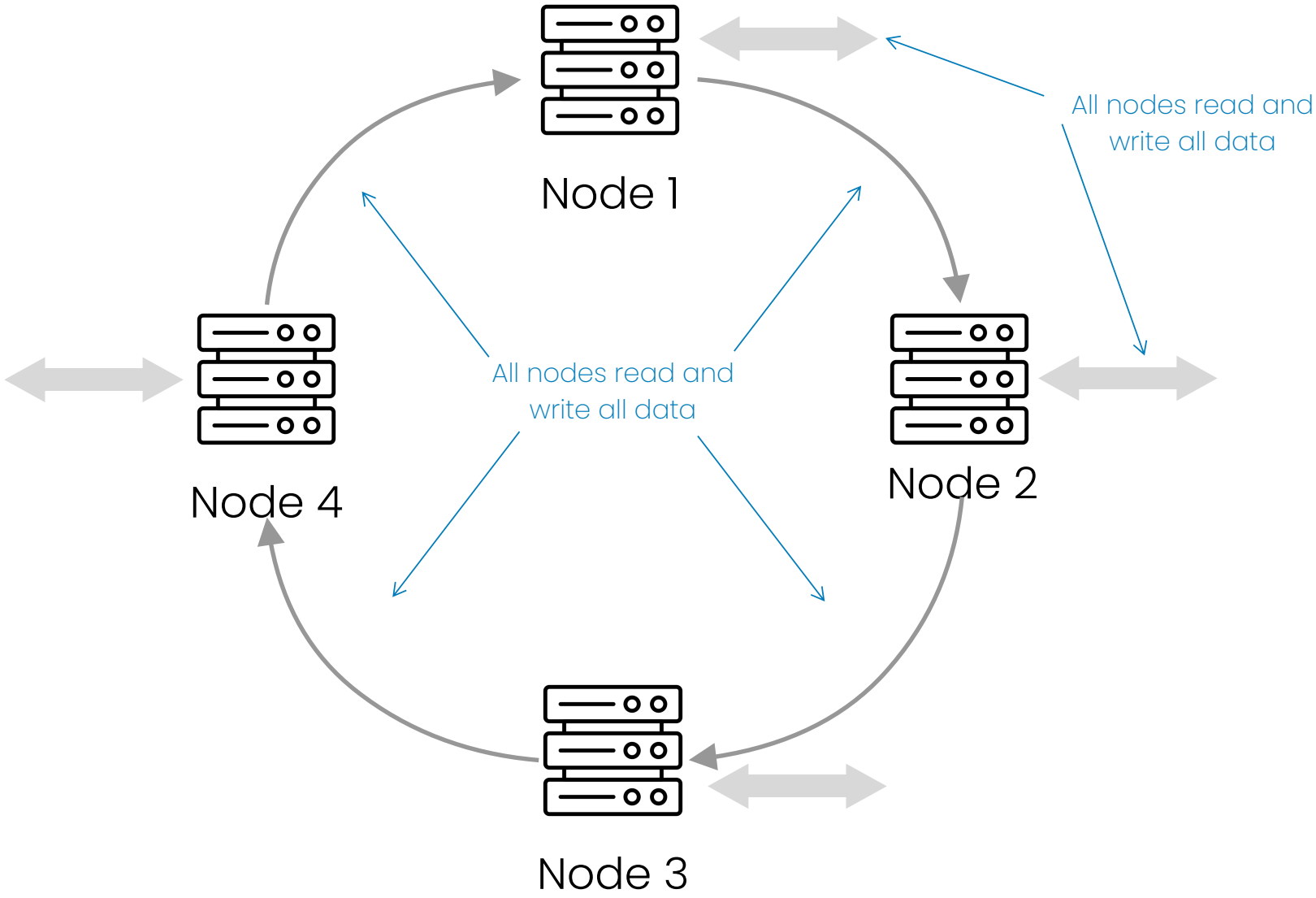


Master-Worker Replication: Pros and Cons

- Pros
  - In case of more read requests, add more worker nodes, and ensure that all read requests are routed to the workers
  - Should the master fail, the workers can still handle read requests
  - Good for datasets with a read-intensive dataset
- Cons
  - The master is a bottleneck
    - Limited by its ability to process updates and to pass those updates on
    - Its failure does eliminate the ability to handle writes until:
      - The master is restored or
      - a new master is appointed
  - Inconsistency due to slow propagation of changes to the workers
  - Bad for datasets with heavy write traffic

Peer to Peer Replication

- All the replicas have equal weight, they can all accept writes
- The loss of any of them doesn't prevent access to the data store



Peer to Peer Replication: Pros and Cons

- Pros
  - You can ride over node failures without losing access to data
  - You can easily add nodes to improve your performance
  - Data is written and read from any node
- Cons
  - Inconsistency!
    - Slow propagation of changes to copies on different nodes
      - Inconsistencies on read lead to problems but are relatively transient
    - Two people can update different copies of the same record stored on different nodes at the same time - a write-write conflict.

How does NoSQL vary from RDBMS?

- Looser schema definition
- Applications written to deal with specific documents/ data
  - Applications aware of the schema definition as opposed to the data
- Designed to handle distributed, large databases
- Trade offs:
  - No strong support for ad hoc queries but designed for speed and growth of database
  - Query language through the API
- Relaxation of the ACID properties

RDB ACID to NoSQL BASE

Atomicity

Consistency

Isolation

Durability

ACID

BASE

Basically Available

Soft-State

Eventual Consistency

Characteristics of NOSQL Databases

- NoSQL avoids:
  - Overhead of ACID transactions
  - Complexity of SQL queries
  - Burden of up-front schema design
  - Transactions (It should be handled at application layer)
- NOSQL Provides:
  - Easy and frequent changes to DB
  - Fast development
  - Large data volumes
  - Schema-less model

What is a schema-less datamodel?

- In relational Databases:
  - You can’t add a record which does not fit the schema
  - You need to add NULLs to unused items in a row
  - We should consider the datatypes. i.e : you can’t add a string to an interger field
  - You can’t add multiple items in a field
    - For this, you should create another table: primary-key, foreign key, join normalization, … !!!
- In NoSQL Databases:
  - There is no schema to consider
  - There is no unused cell
  - There is no datatype (implicit)
  - Most of considerations are done in the application layer
  - We gather all items in an aggregate (document)

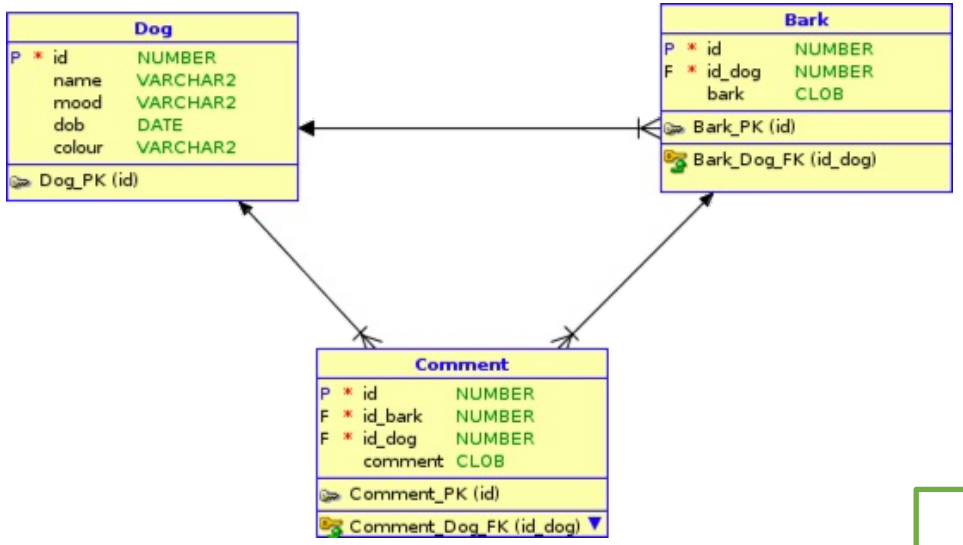
Types of NOSQL Databases

- Key-value
- Graph database
- Document-oriented
- Column-oriented

Key-value data model

- One of the simplest types, a kind of distributed Hashmap
- Designed to save data without defining a schema
- All data is in key/value form
  - The value can be a string, a serialized object, blob…
  - The data is opaque to the system: it is not possible to access it without using the key
- Lack of typing has an impact on querying: all the intelligence carried before by the queries must be carried by the application that queries the DB
- Communications are mostly limited to PUT, GET and DELETE operations
- *Objective*: provide quick access to information, keep the session of a web site…
- *Example*: DynamoDB (Amazon), Azure Table Storage (ATS), Redis, BerkeleyDB, Voldemort (LinkedIn)

Key-value data model



	Name	Mood	DOB	Colour
12	Stella	Happy	2007-04-01	NULL
13	Wimma	Hungry	NULL	Black
9	Ninja	NULL	NULL	NULL

Key

Dog_12

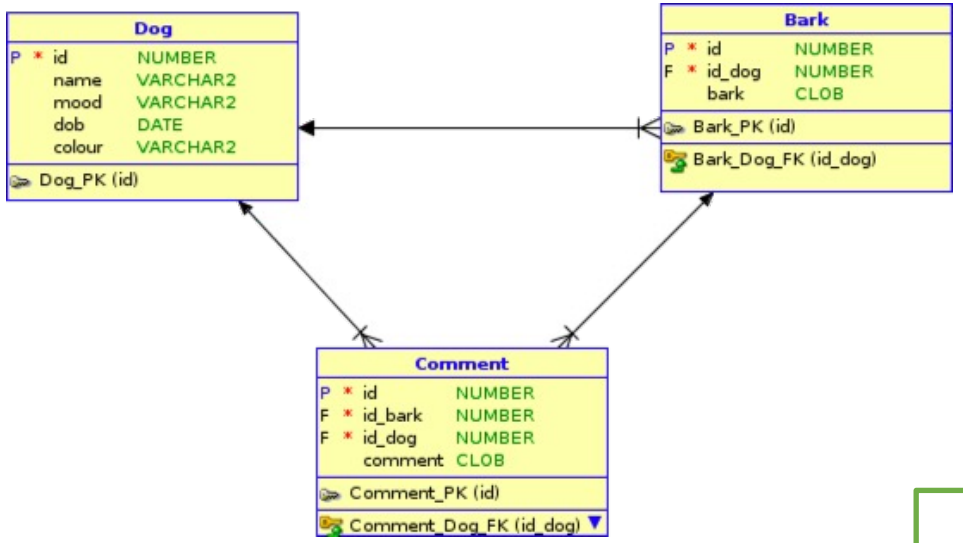
Value

Name_$#_Stella~~Mood_$#_Happy~~DOB_$#_2007-04-01...

Document-oriented Databases

- Extend the key/value paradigm, with more complex "documents" instead of simple data, and a unique key for each one
  - JSON or XML documents
- Each document is an object, contains one or more fields, and each field contains a typed value (string, date, binary or array)
- Allow to store, extract and manage document-oriented information (semi-structured data)
- *Advantage*: be able to retrieve, via a single key, a set of hierarchically structured information
  - In relational databases, this would imply several joins
- *Examples*: Mongo DB (SourceForge), Couch DB (Apache), RavenDB (for .Net/Windows platforms, querying via LINQ)

Document-oriented Databases



	Name	Mood	DOB	Colour
12	Stella	Happy	2007-04-01	NULL
13	Wimma	Hungry	NULL	Black
9	Ninja	NULL	NULL	NULL

Key

Dog_12

Document (V1)

```
{
        type :      « Dog »,
        name :      « Stella »,
        mood :      « Happy »,
        dob : 2007-04-01
}
```

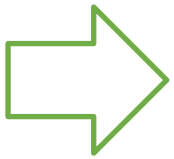
Document (V2)

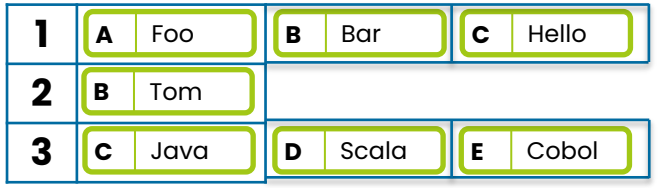
```
{
        type :      « Dog »,
        name :      « Stella »,
        mood :      « Happy »,
        dob : 2007-04-01
        bark : [
                { text : « I'm famished!»
                  comment : [
                        { id_dog : « dog_4 »,
                          text : « Nobody cares! »
                        }
                  ]
        ]
}
```

Column-oriented Databases

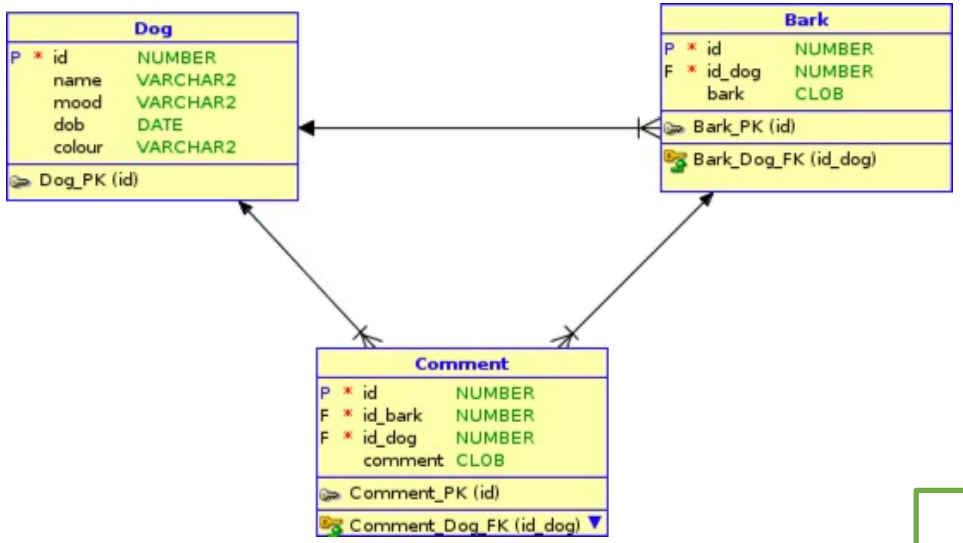
- Evolution of the key/value DB
- Resemble RDBMS, but with a dynamic number of columns, different from one record to another (no columns with NULL values)
- Offer very high performance and a highly scalable architecture
- *Examples*: Hbase (Hadoop), Cassandra (Facebook, Twitter), BigTable (Google)

	A	B	C	D	E
1	Foo	Bar	Hello		
2		Tom			
3			Java	Scala	Cobol



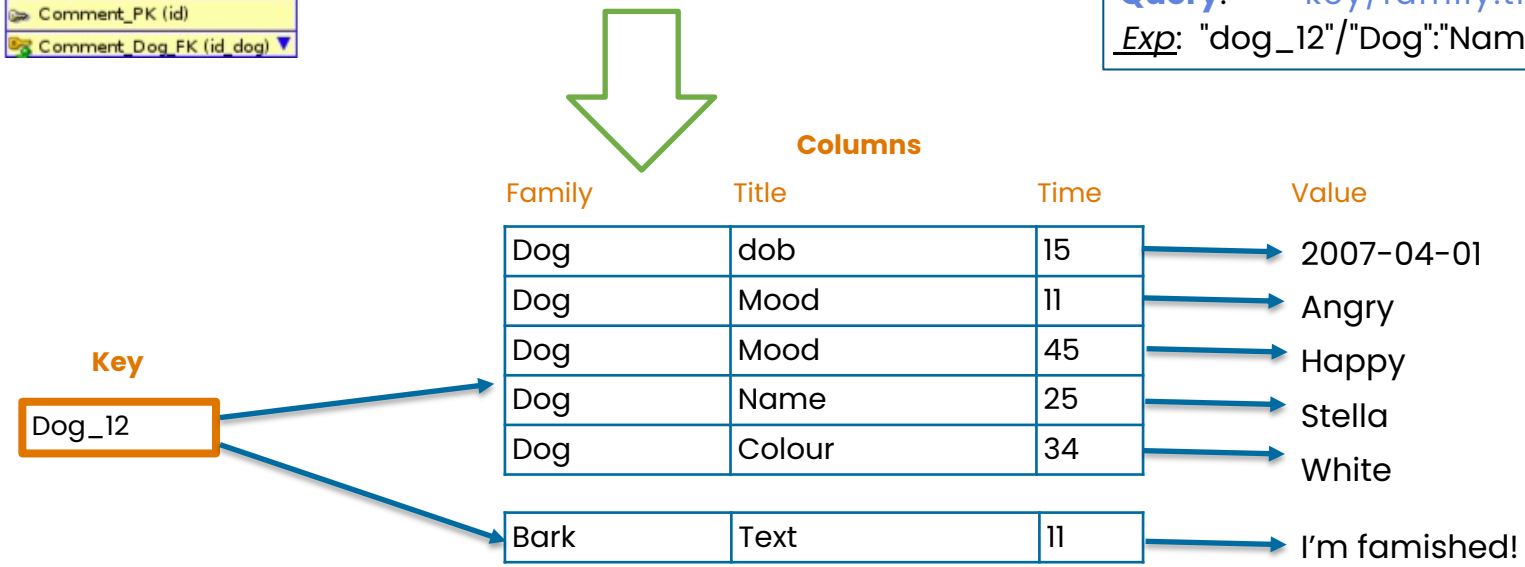


Column-oriented Databases



	Name	Mood	DOB	Colour
12	Stella	Happy	2007-04-01	White
13	Wimma	Hungry	NULL	Black
9	Ninja	NULL	NULL	NULL

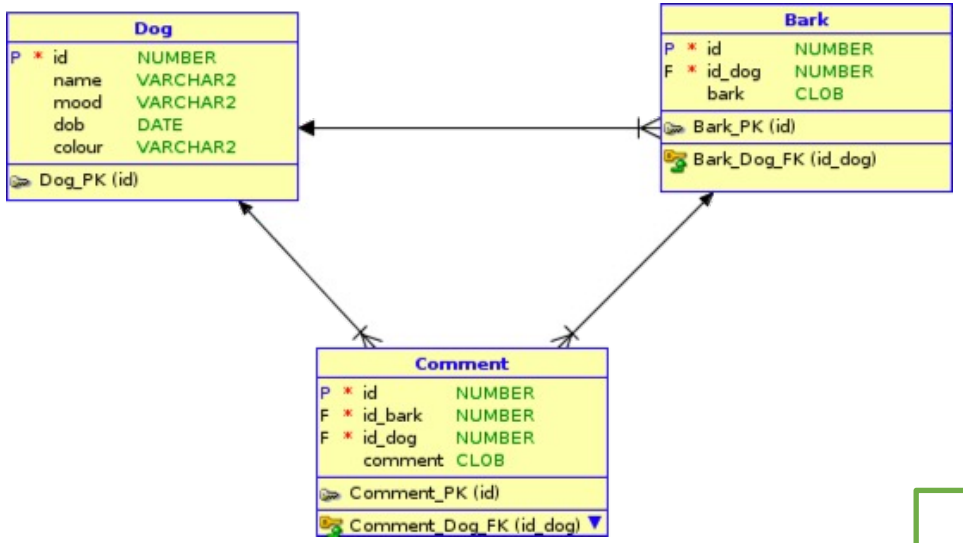
Query: key/family:title[/time]
Exp: "dog_12"/"Dog":"Name" ➔ Stella



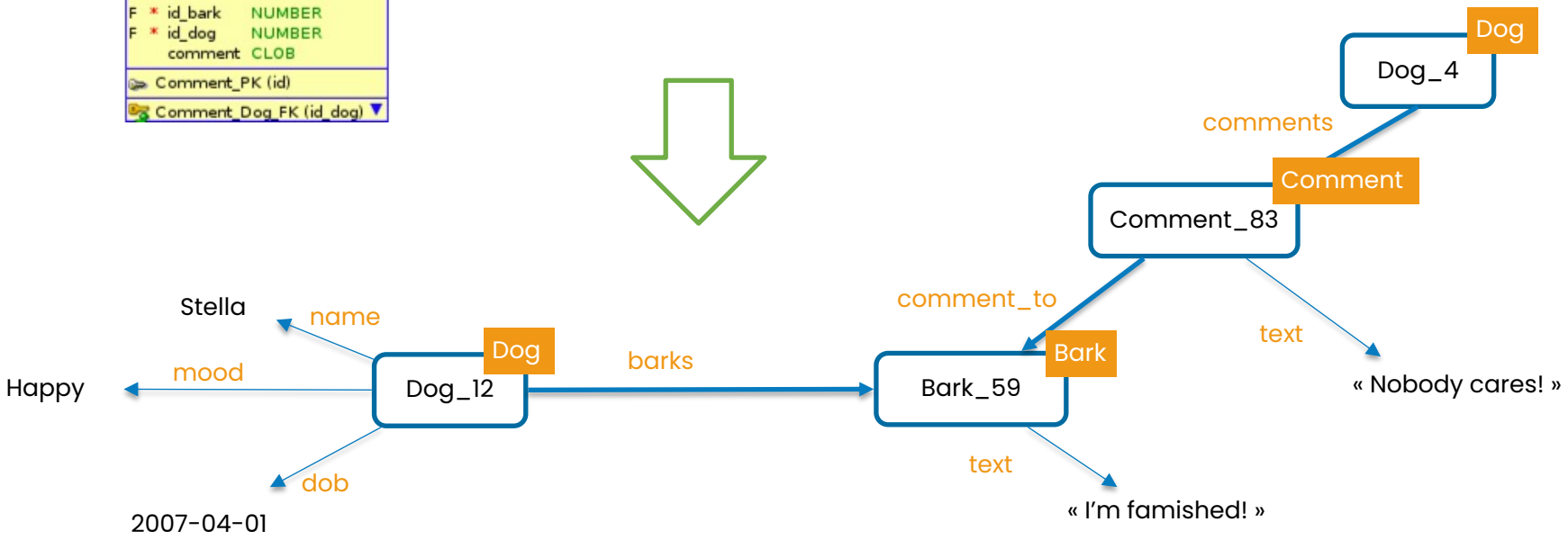
Graph Databases

- Based on graph theory
- Relies on the notions of nodes, relationships and their related properties
- Designed for data whose relationships are represented as graphs, and having interconnected elements, with an indeterminate number of relationships between them
- Adapted to social network data processing
- *Examples*: Neo4J, InfiniteGraph, OrientDB

Graph Databases



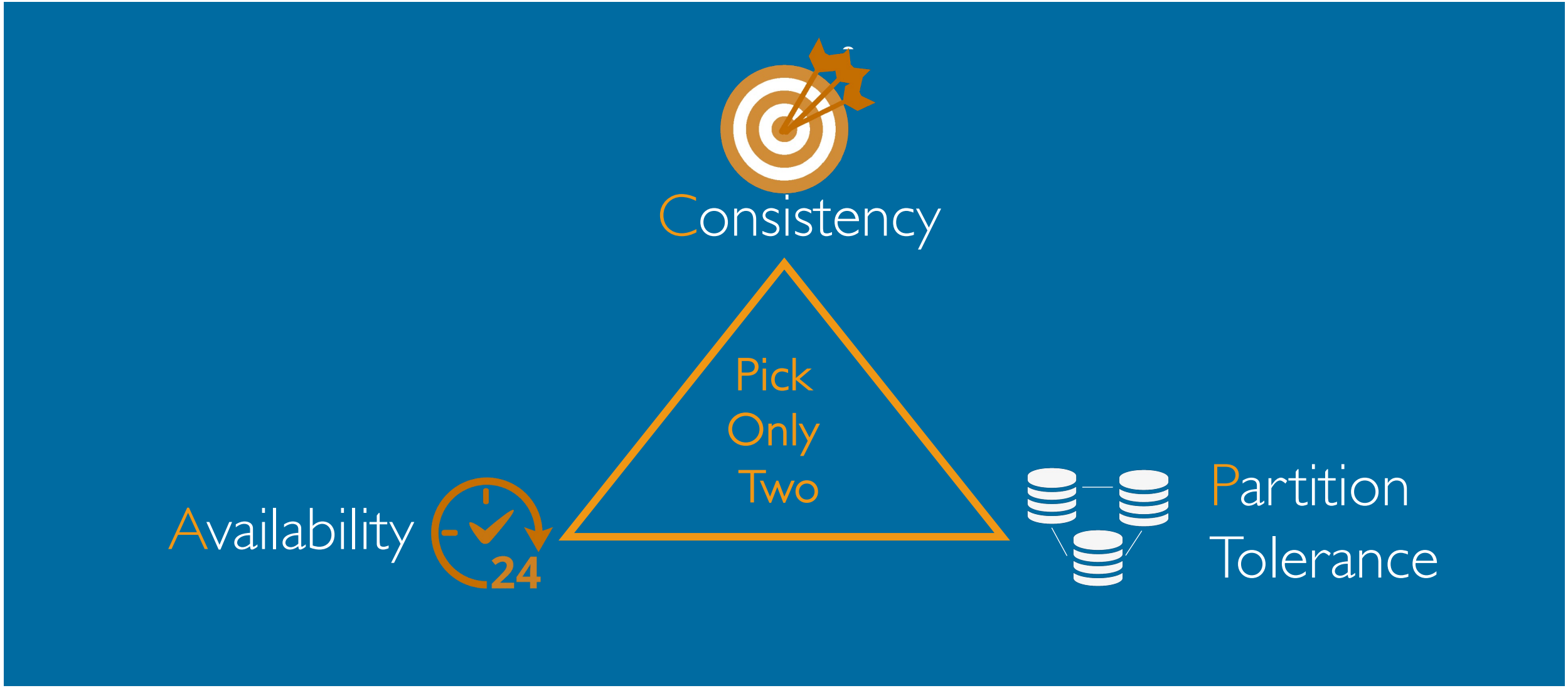
	Name	Mood	DOB	Colour
12	Stella	Happy	2007-04-01	White
13	Wimma	Hungry	NULL	Black
9	Ninja	NULL	NULL	NULL



SQL vs NOSQL: Some considerations

- What does my data look like?
- How is the current data stored?
- How important is the guaranteed accuracy of database transactions?
- How important is the speed of the database?
- What happens when the data is not available?
- How is the database being used?
- Should I split up the data to leverage the advantages of both RDBMS and NoSQL?

CAP Theorem


Consistency
Pick
Only
Two
Availability
24
Partition
Tolerance

CAP Theorem

- What the CAP theorem really says:
  - If you cannot limit the number of faults and requests can be directed to any server and you insist on serving every request you receive, then you cannot possibly be consistent
- How it is interpreted: (wrong!)
  - You must always give something up: consistency, availability or tolerance to failure and reconfiguration

CAP Theorem for NoSQL

- Consistency:
  - All replicas contain the same version of data
  - Client always has the same view of the data (no matter what node)
- Availability:
  - System remains operational on failing nodes
  - All clients can always read and write
- Partition tolerance
  - multiple entry points
  - System remains operational on system split (communication malfunction)
  - System works well across physical
  - network partitions

Benefits of NoSQL

- Elastic Scaling
  - RDBMS scale up – bigger load , bigger server
  - NO SQL scale out – distribute data across multiple hosts seamlessly
- DBA Specialists
  - RDMS require highly trained expert to monitor DB
  - NoSQL require less management, automatic repair and simpler data models
- Big Data
  - Huge increase in data RDMS: capacity and constraints of data volumes at its limits
  - NoSQL designed for big data

Benefits of NoSQL

- Flexible data models
  - Change management to schema for RDMS have to be carefully managed
  - NoSQL databases more relaxed in structure of data
    - Database schema changes do not have to be managed as one complicated change unit
    - Application already written to address an amorphous schema

- Economics
  - RDMS rely on expensive proprietary servers to manage data
  - No SQL: clusters of cheap commodity servers to manage the data and transaction volumes
  - Cost per gigabyte or transaction/second for NoSQL can be lower than the cost for a RDBMS

Drawbacks of NOSQL

- Support
  - RDBMS vendors provide a high level of support to clients
  - Stellar reputation
  - NoSQL – are open source projects with startups supporting them
    - Reputation not yet established

- Maturity
  - RDMS are mature products: means stable and dependable
    - Also means old, no longer cutting edge nor interesting
  - NoSQL are still implementing their basic feature set

Drawbacks of NOSQL

- Administration
  - RDMS administrator is a well defined role
  - NOSQL's goal: no administrator necessary, however NO SQL still requires effort to maintain
- Lack of Expertise
  - Whole workforce of trained and seasoned RDMS developers
  - Still recruiting developers to the NoSQL camp
- Analytics and Business Intelligence
  - RDMS designed to address this niche
  - NoSQL designed to meet the needs of an Web 2.0 application - not designed for ad hoc query of the data
    - Tools are being developed to address this need