CPSC 436C
Cloud Computing for Data Science

Data store - Part2

Maryam R.Aliabadi

mraiyata@cs.ubc.ca

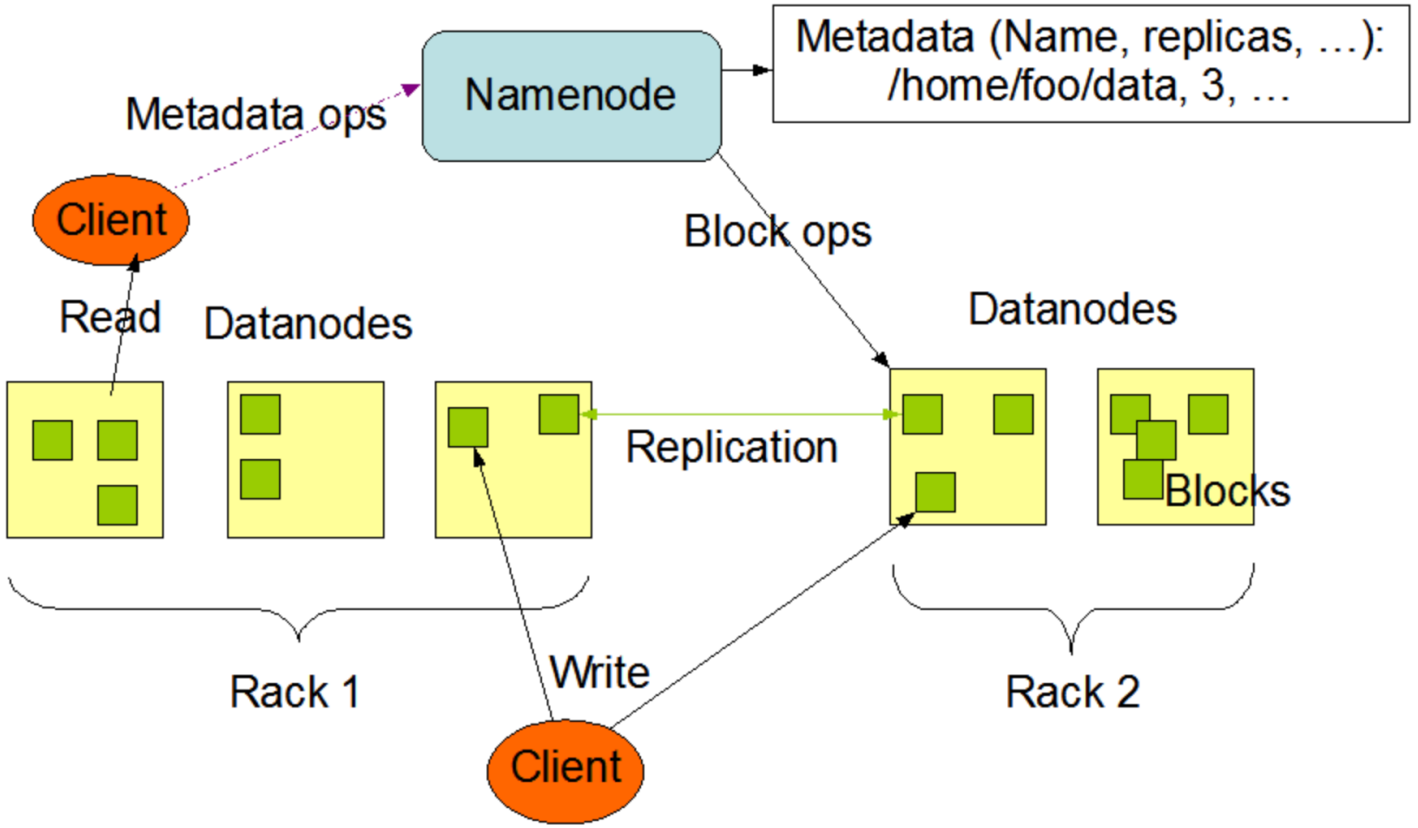
Spring 2024

Last Class‘ Review

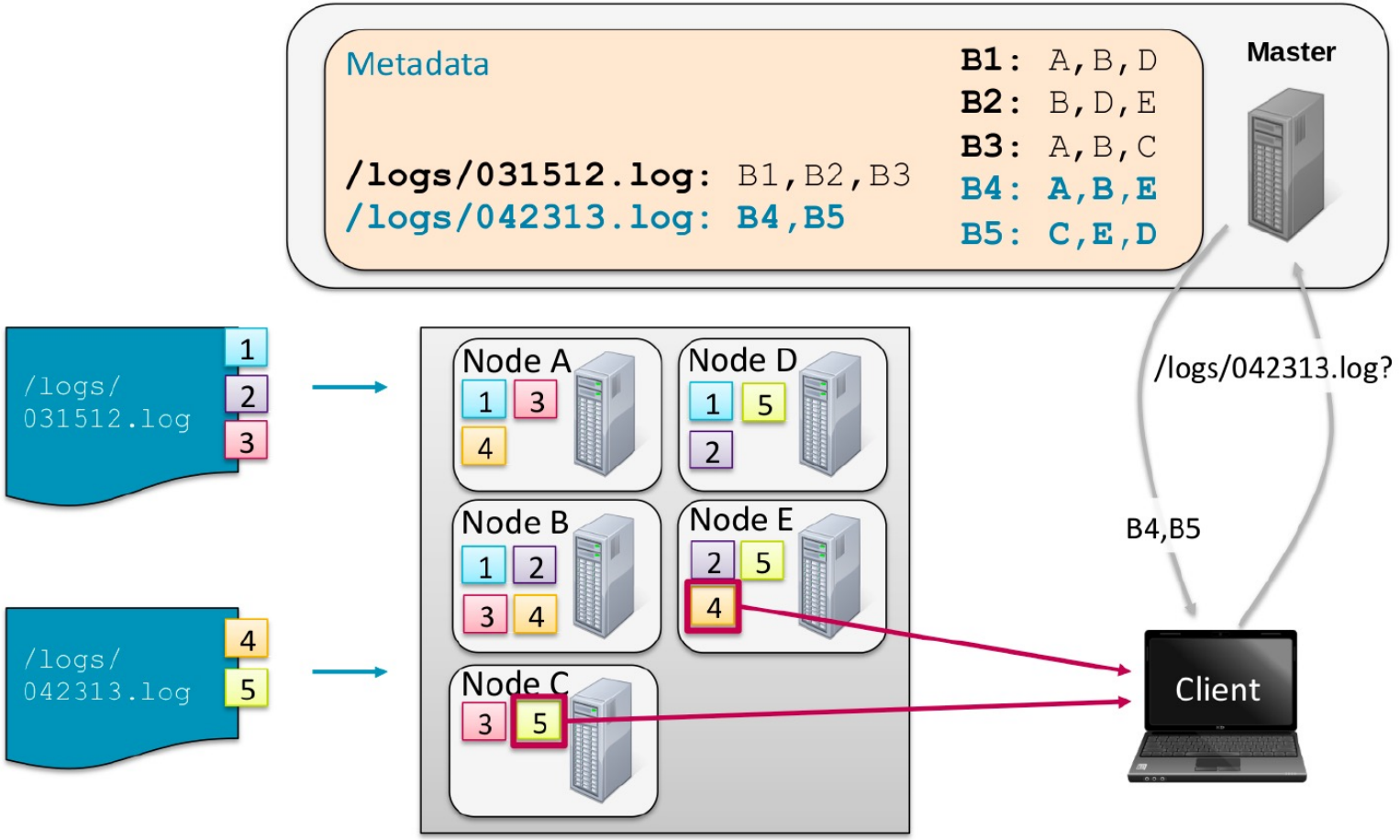
- HDFS Architecture
- HDFS Components
- Name node operations
- HDFS API
- HDFS Fault tolerance

HDFS Architecture

- Main components:
  - HDFS Name node
  - HDFS Data node
  - HDFS Client

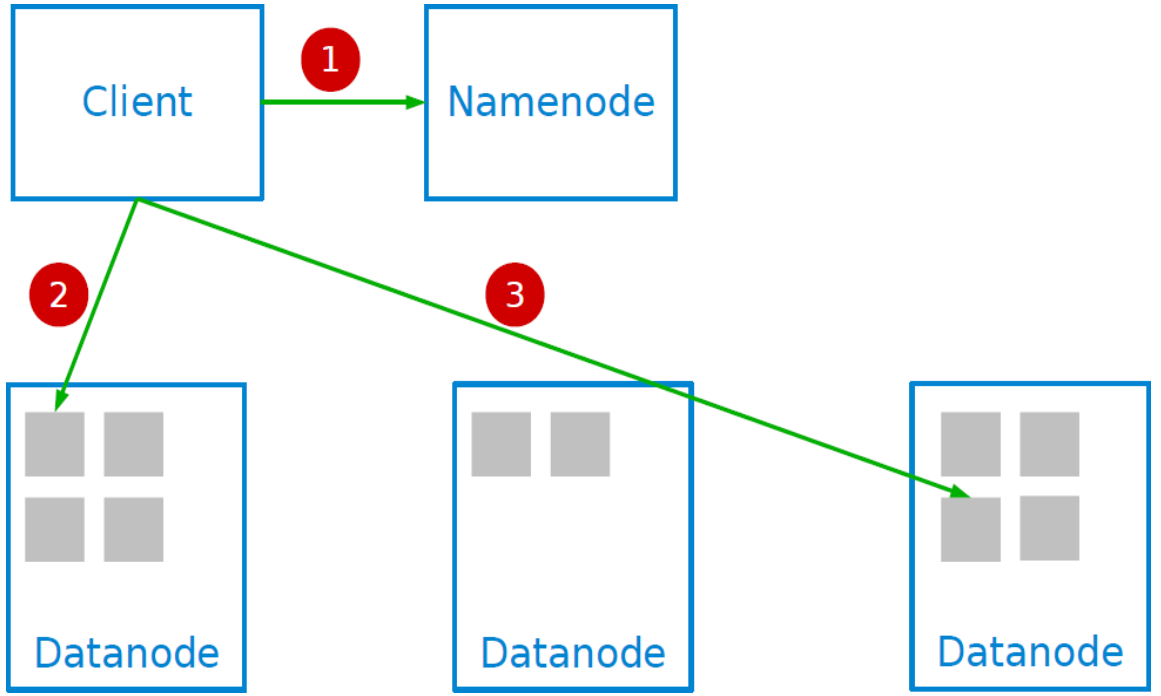


How to store and retrieve files?



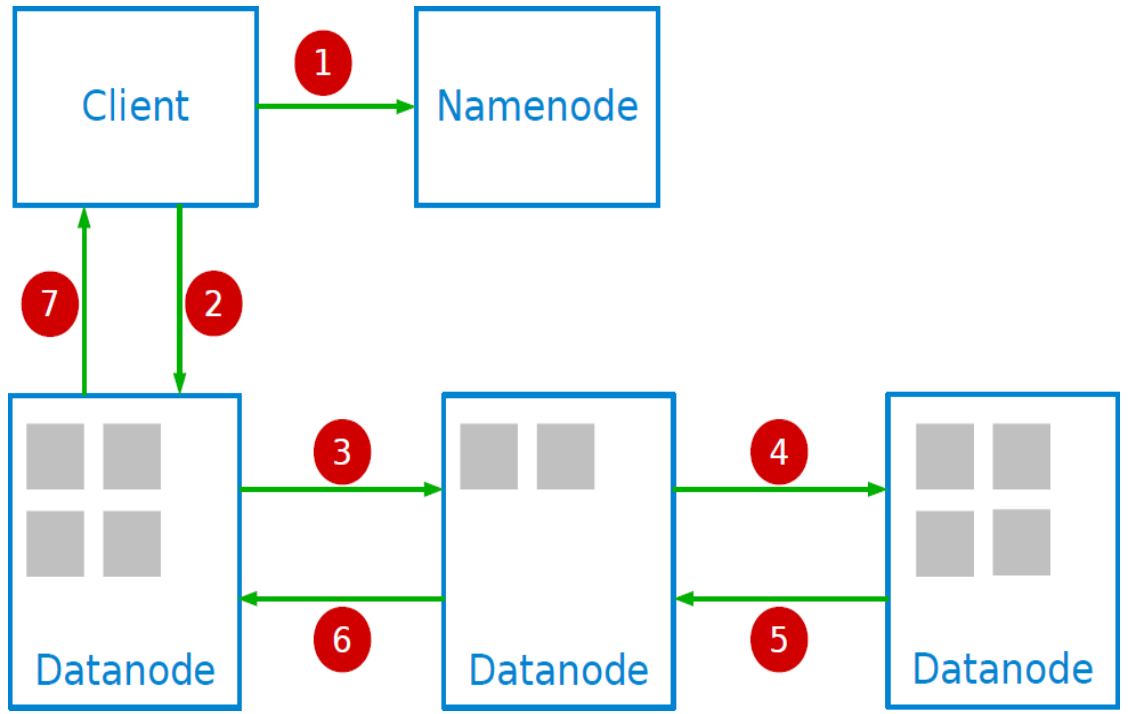
HDFS Read

- 1. Retrieve block locations.
- 2, 3. Read blocks to re-assemble the file.



HDFS Write

- 1. Create a new file in the Namenode's Namespace; calculate block topology.
- 2, 3, 4. Stream data to the first, second and third node.
- 5, 6, 7. Success/failure acknowledgment.



HDFS Advantages

- Scalability
- Fault tolerance
- High Throughput
- Data Locality

Quiz

join.iClicker.com
YFMA

1. How does HDFS ensure fault tolerance?

A. By compressing data
B. By replicating data across multiple nodes
C. By partitioning data into smaller blocks
D. By encrypting data at rest

Quiz

join.iClicker.com
YFMA

2. In HDFS, what is NOT the role of the NameNode?

A. Stores data blocks
B. Manages metadata and namespace
C. Performs garbage collection
D. Handles data replication

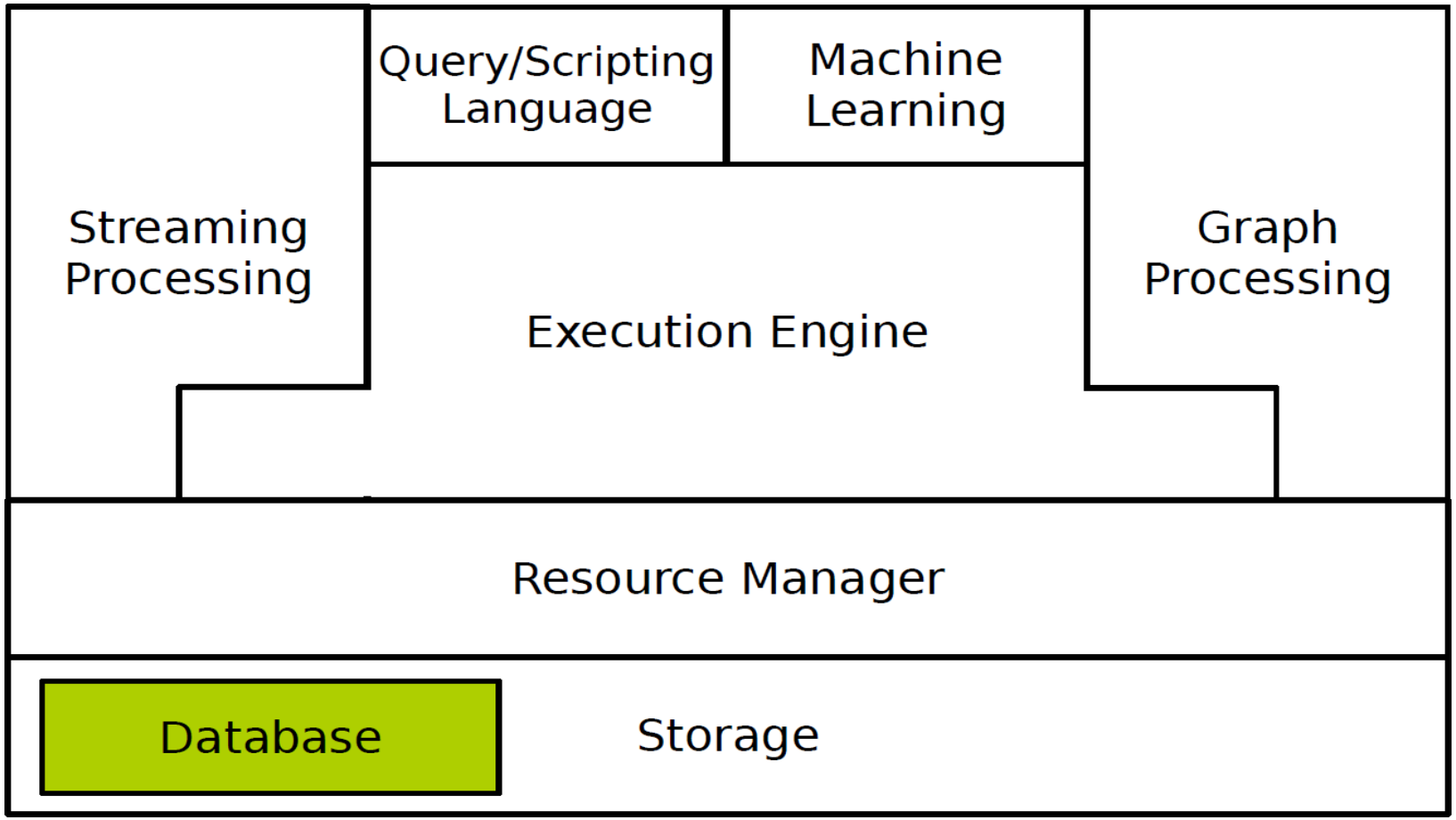
Quiz

join.iClicker.com
YFMA

- **What is the purpose of the Secondary NameNode in HDFS?**

  - A. To act as a backup for the NameNode
  - B. To manage metadata and namespace
  - C. To handle data replication
  - D. To act as a checkpoint for the Namenode

Today's Topic

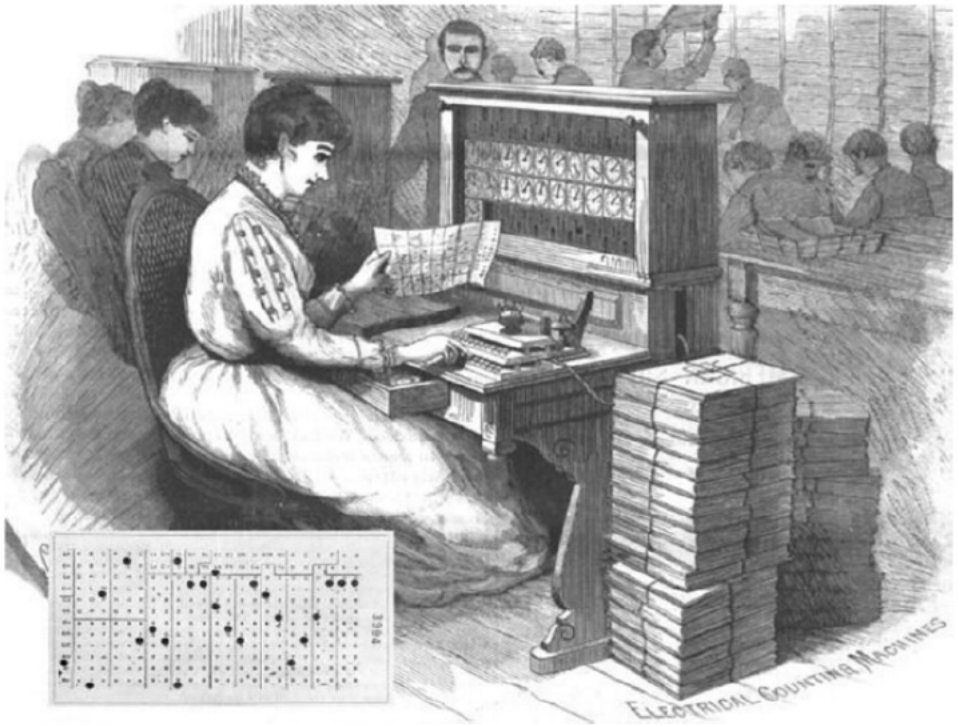


Database and Database Management System

- Database: an organized collection of data.
- Database Management System (DBMS): a software that interacts with users, other applications, and the database itself to capture and analyze data.

Early database systems

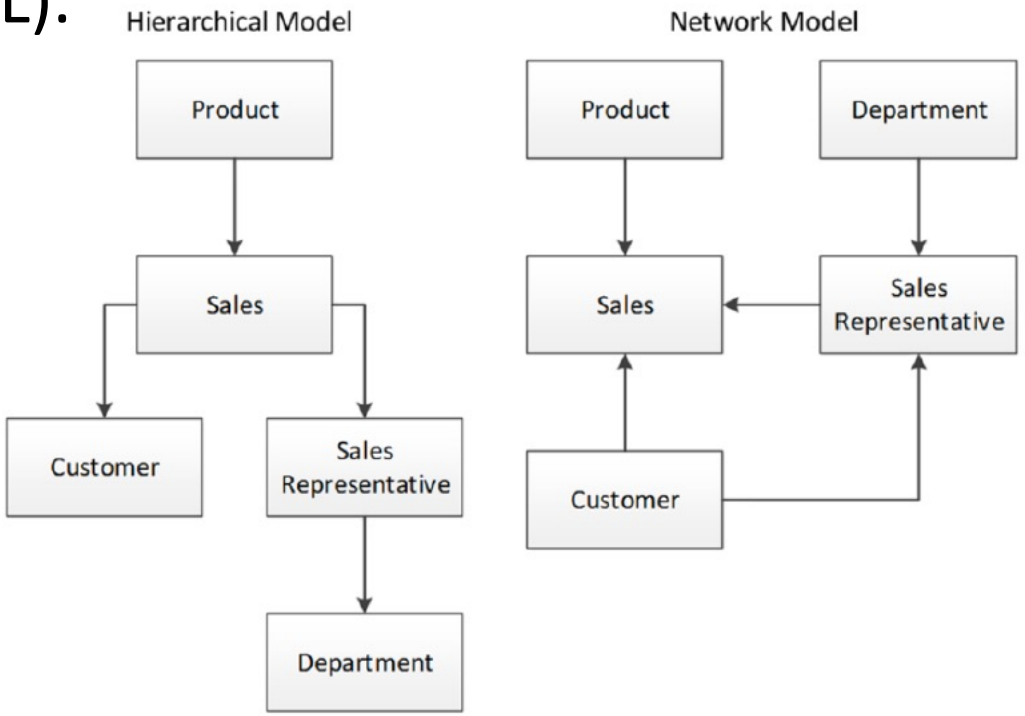
- There were databases but no Database Management Systems (DBMS).



[Guy Harrison, Next Generation Databases: NoSQLand Big Data, 2015]

The First Database Revolution

- Navigational data model: hierarchical model (IMS) and network model (CODASYL).
- **Disk-aware**



[Guy Harrison, Next Generation Databases: NoSQLand Big Data, 2015]

The Second Database Revolution

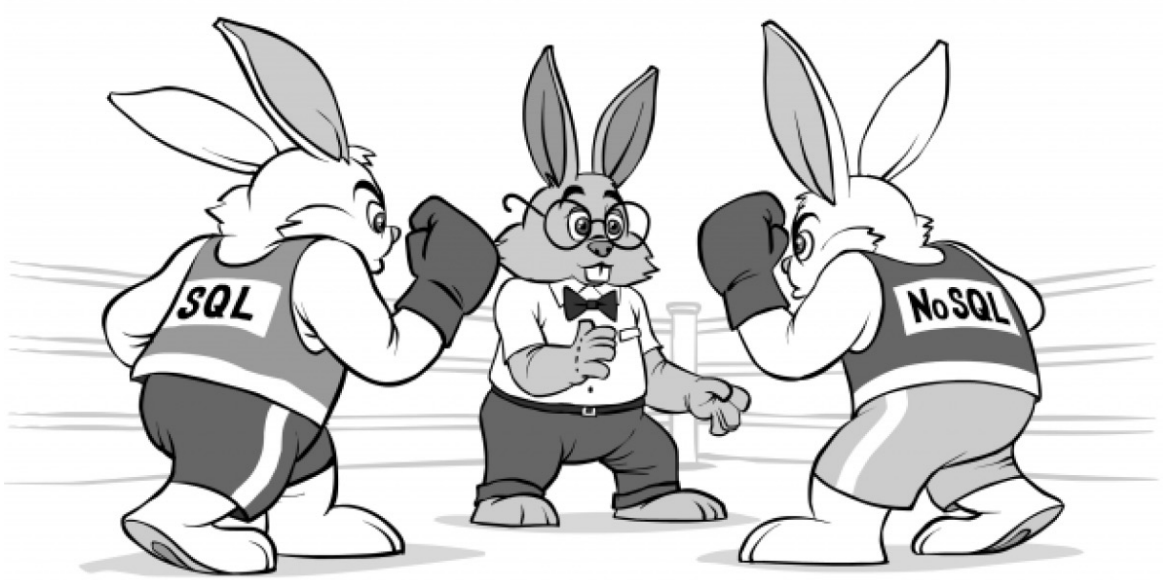
- **Relational** data model:
  - Logical data is disconnected from physical information storage
- **ACID** transactions
  - Atomic, Consistent, Isolated, Durable
- **SQL** language
- **Object** databases
  - Information is represented in the form of objects

ACID Properties

- Atomicity
- Consistency
- Isolation
- Durability

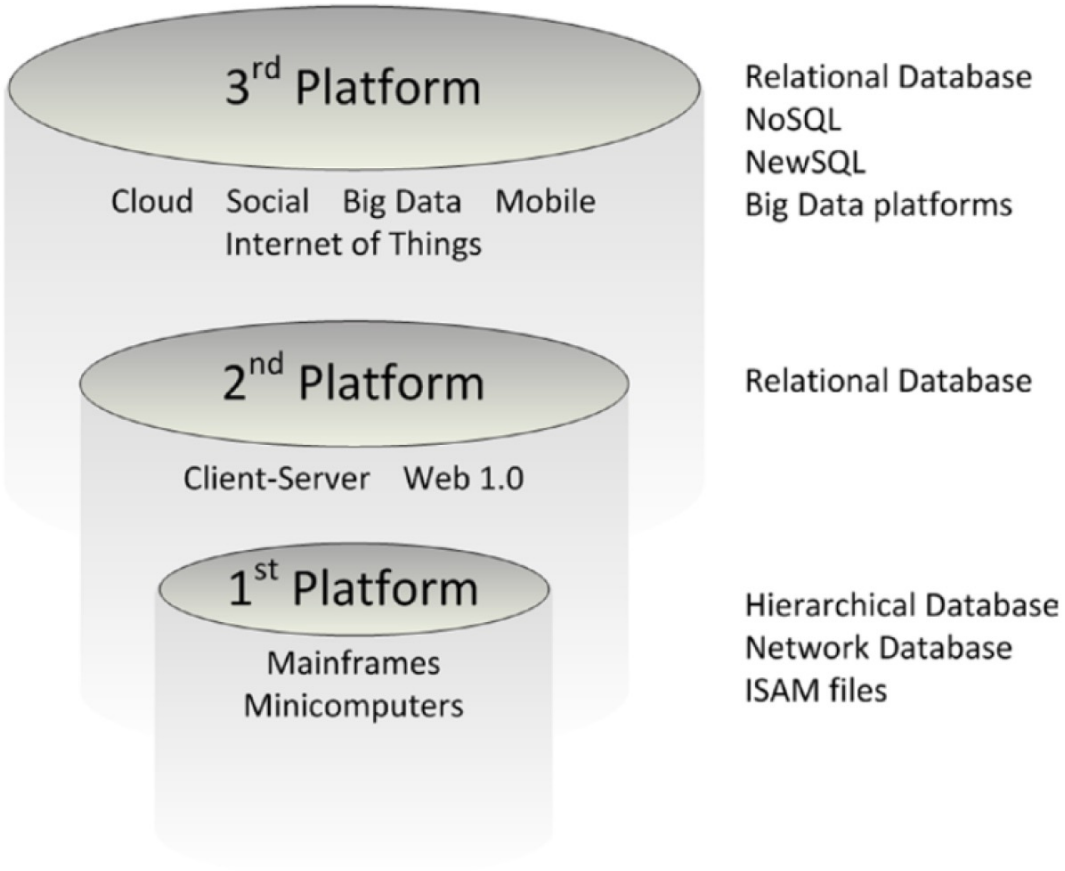
The Third Database Revolution

- NoSQL databases : BASE instead of ACID
- NewSQL databases : scalable performance of NoSQL + ACID



[http://ithare.com/nosql-vs-sql-for-mogs]

Three Waves of Database Technology



[Guy Harrison, Next Generation Databases: NoSQLand Big Data, 2015]

SQL Databases

Relational SQL Database

- RDBMSs: the dominant technology for storing structured data in web and business applications.

- SQL is good
  - Rich language and toolset
  - Easy to use and integrate
  - Many vendors

- They promise: **ACID**

ACID Properties

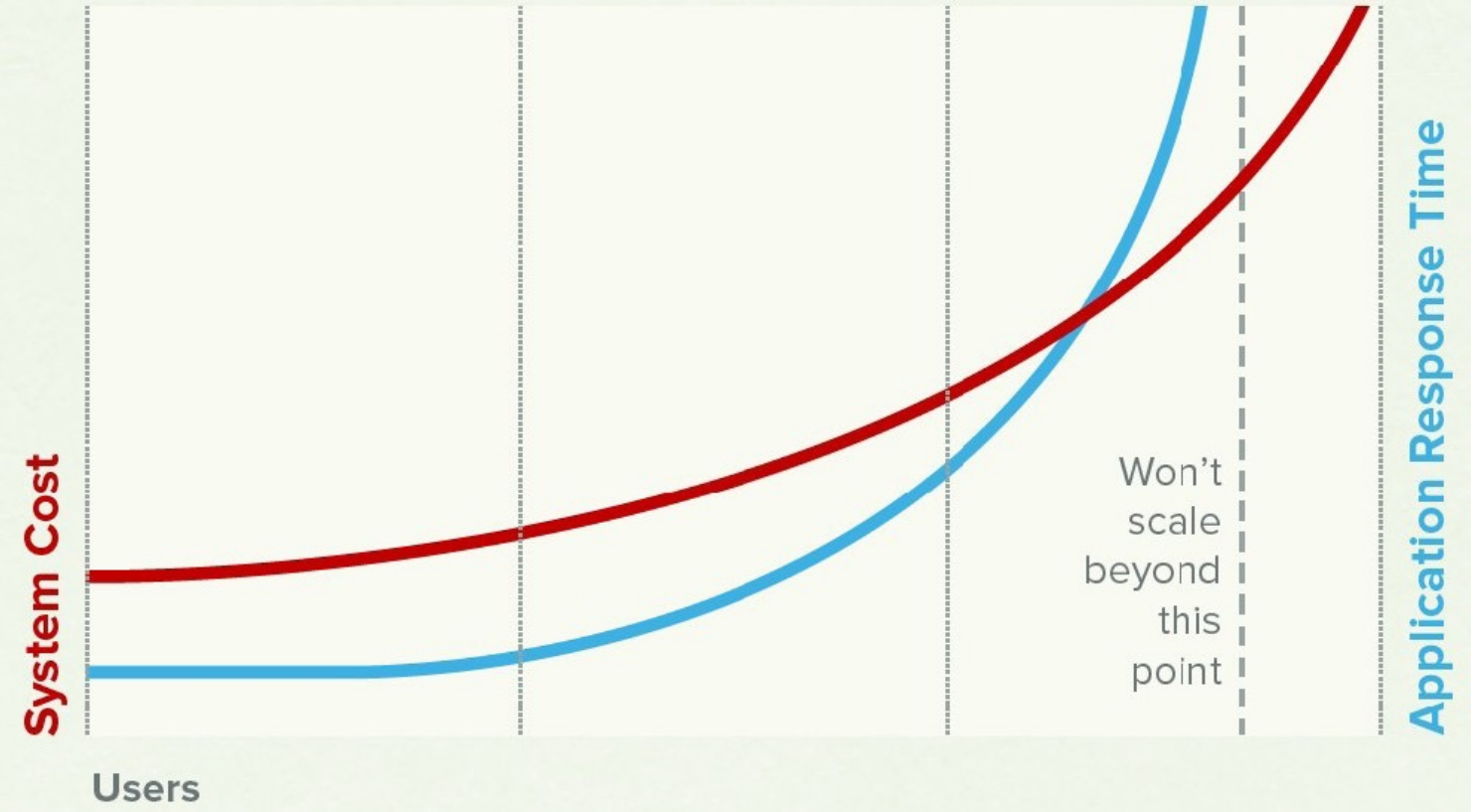
- **A**tomicity
  - All included statements in a transaction are either executed or the whole transaction is aborted without affecting the database.
- **C**onsistency
  - A database is in a consistent state before and after a transaction.
- **I**solation
  - Transactions can not see uncommitted changes in the database.
- **D**urability
  - Changes are written to a disk before a database commits a transaction so that committed data cannot be lost through a power failure.

SQL Database Challenges

- Web-based applications caused spikes.
  - Internet-scale data size
  - High read-write rates
  - Frequent schema changes

- RDBMS were not designed to be distributed.

Scaling RDBMS is Expensive and Inefficient



[http://www.couchbase.com/sites/default/files/uploads/all/whitepapers/NoSQLWhitepaper.pdf]

Large Scale Application Requirement

- The large-scale applications have to be reliable: availability + redundancy

- These properties are difficult to achieve with ACID properties.

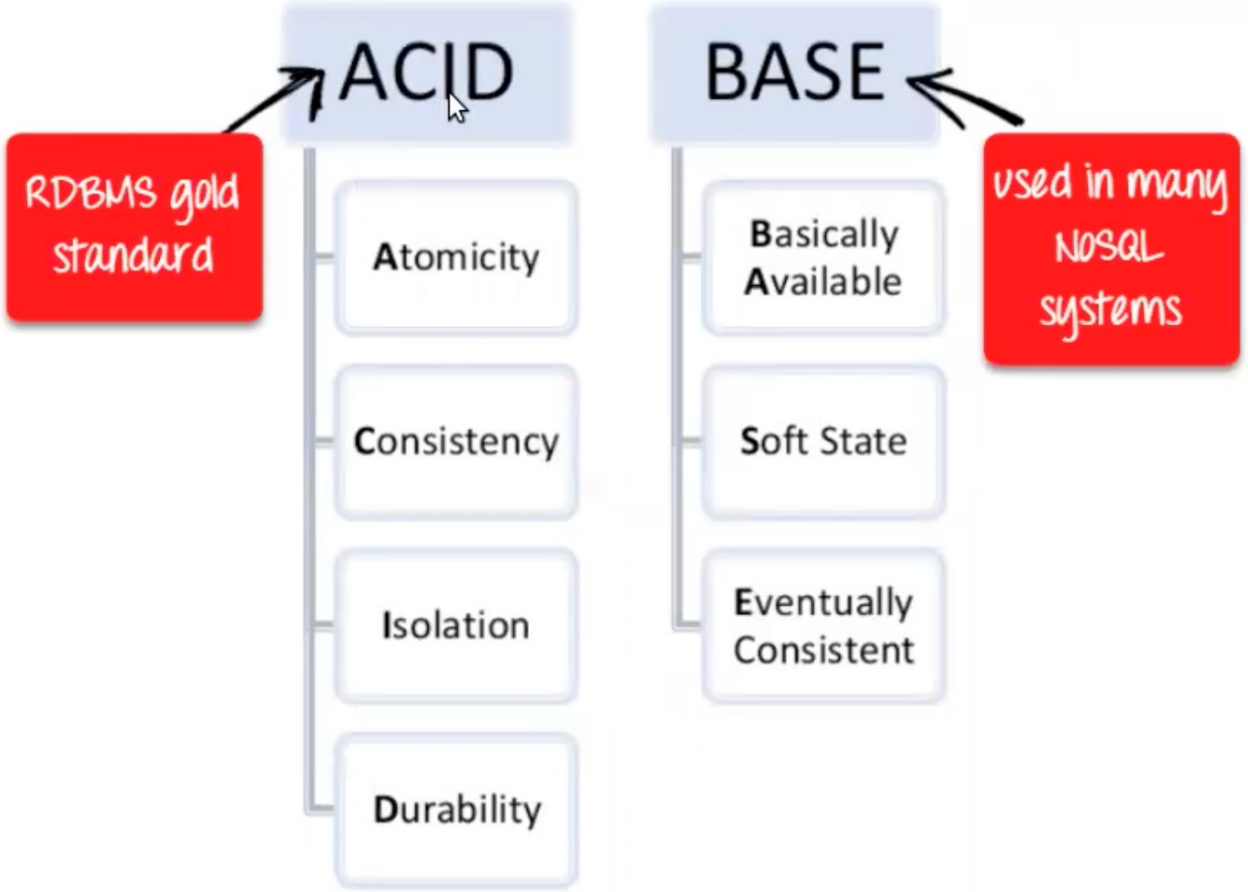
- The BASE approach forfeits the ACID properties of consistency and isolation in favor of availability, graceful degradation, and performance.

BASE Properties

- Basic Availability
  - Possibilities of faults but not a fault of the whole system.
- Soft-state
  - Copies of a data item may be inconsistent
- Eventually consistent
  - Copies becomes consistent at some later time if there are no more updates to that data item

Solution

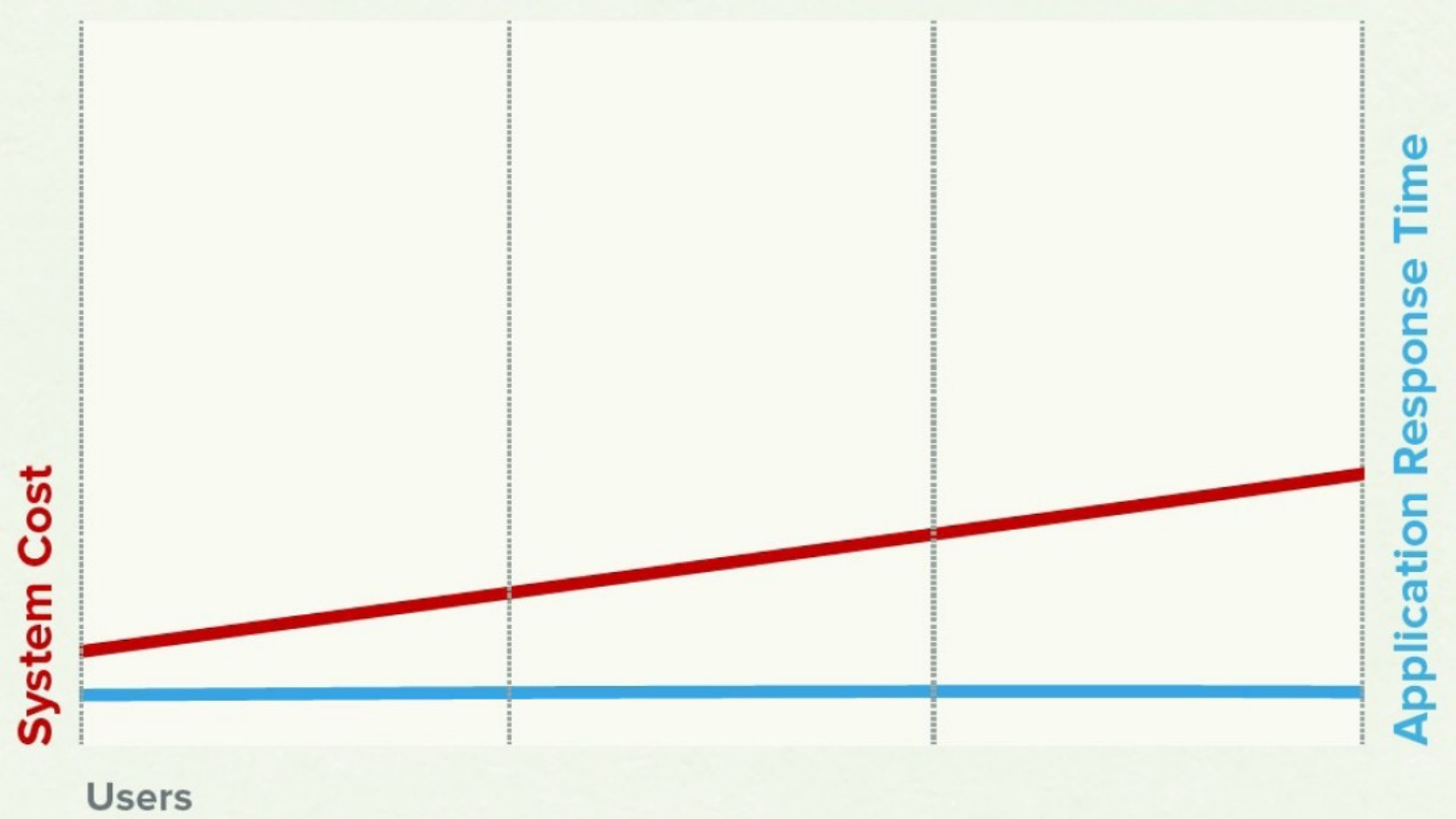
NoSQL is based on BASE



NoSQL

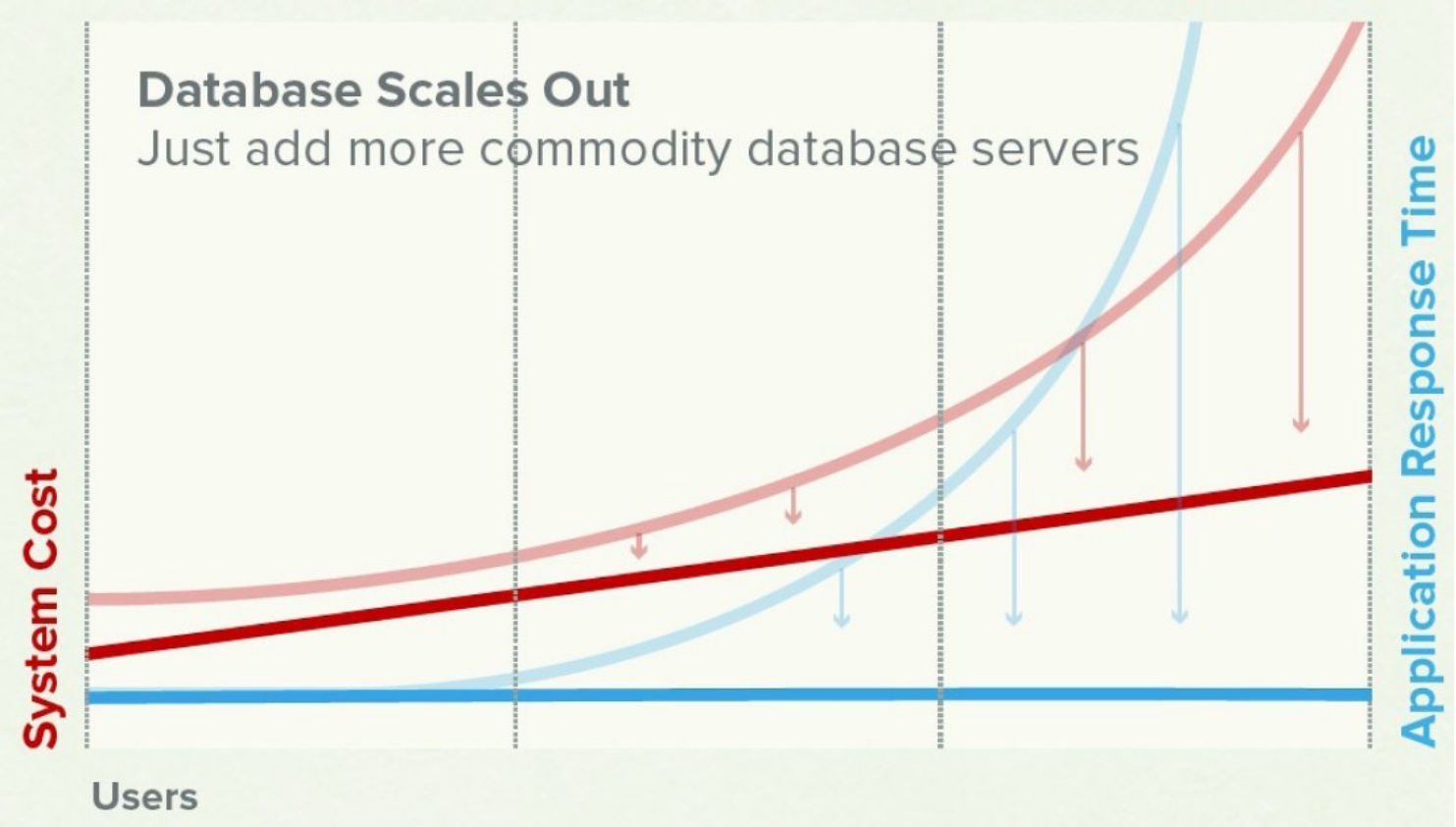
- Avoids
  - Overhead of ACID properties
  - Complexity of SQL query
- Provides
  - Scalability
  - Easy and frequent changes to DB
  - Large data volumes

NoSQL Cost and Performance



[http://www.couchbase.com/sites/default/files/uploads/all/whitepapers/NoSQLWhitepaper.pdf]

RDBMS Vs. NoSQL



[http://www.couchbase.com/sites/default/files/uploads/all/whitepapers/NoSQLWhitepaper.pdf]

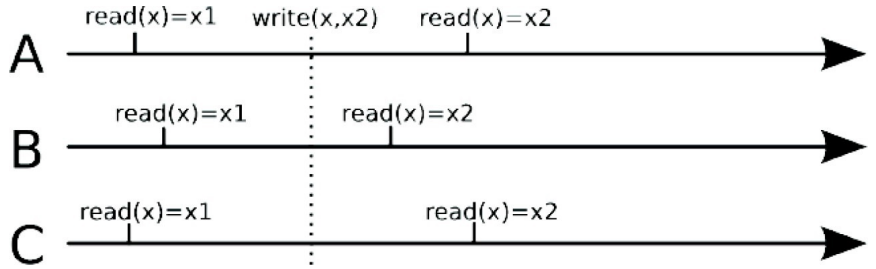
CAP Theorem

Availability

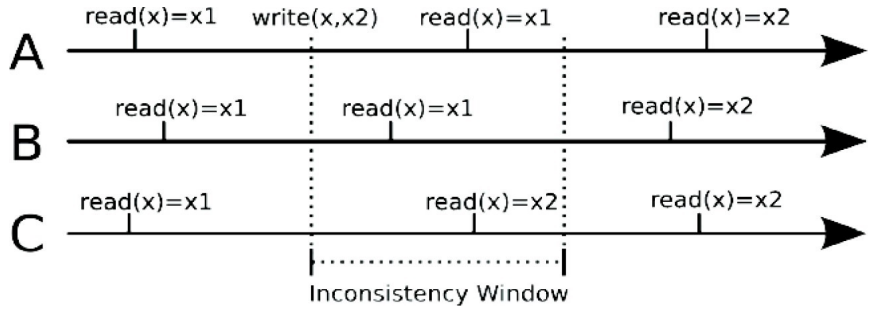
- Replicating data to improve the availability of data.
- Data replication
  - Storing data in more than one site or node

Consistency

- Strong consistency
  - After an update completes, any subsequent access will return the updated value.



- Eventual consistency
  - Does not guarantee that subsequent accesses will return the updated value.
  - Inconsistency window.
  - If no new updates are made to the object, eventually all accesses will return the last updated value.

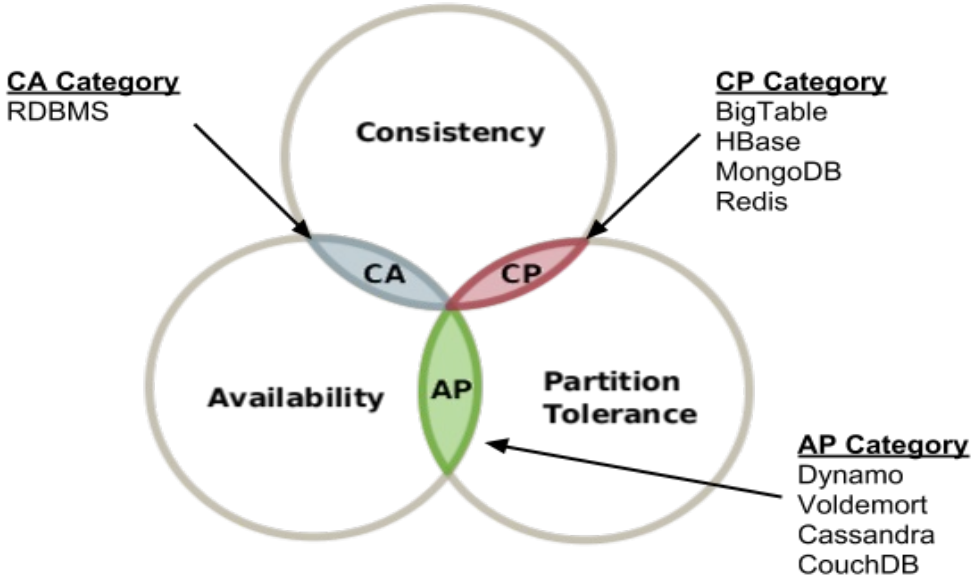


Partition Tolerance

- Partition tolerance, in the context of databases and distributed systems, refers to a system's ability to continue functioning correctly and reliably even when network partitions or communication failures occur.

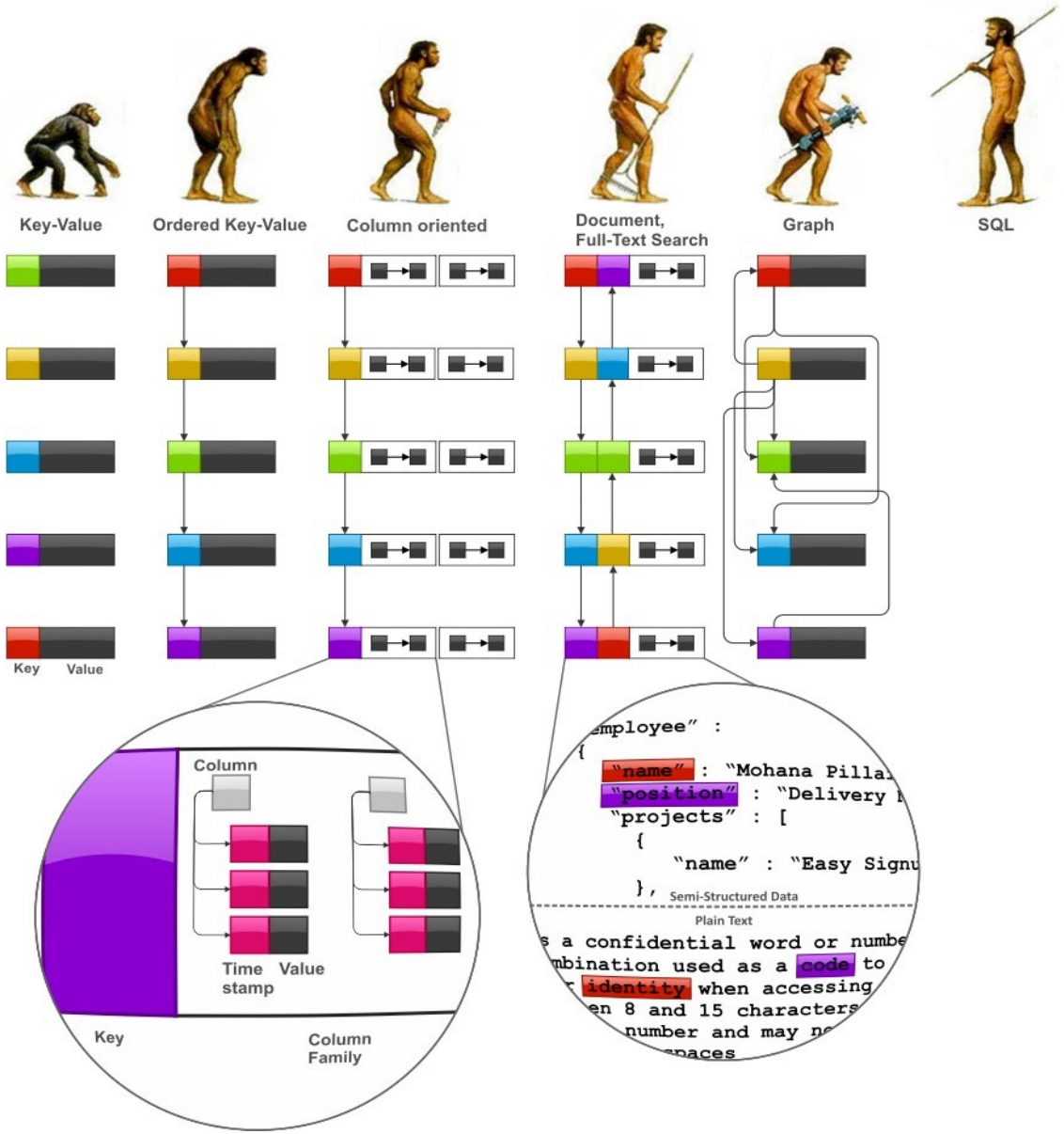
CAP Theorem

- Consistency : Consistent state of data after the execution of an operation.
- Availability : Clients can always read and write data.
- Partition Tolerance :Continue the operation in the presence of network partitions.
- You can choose only two!



NoSQL Data Models

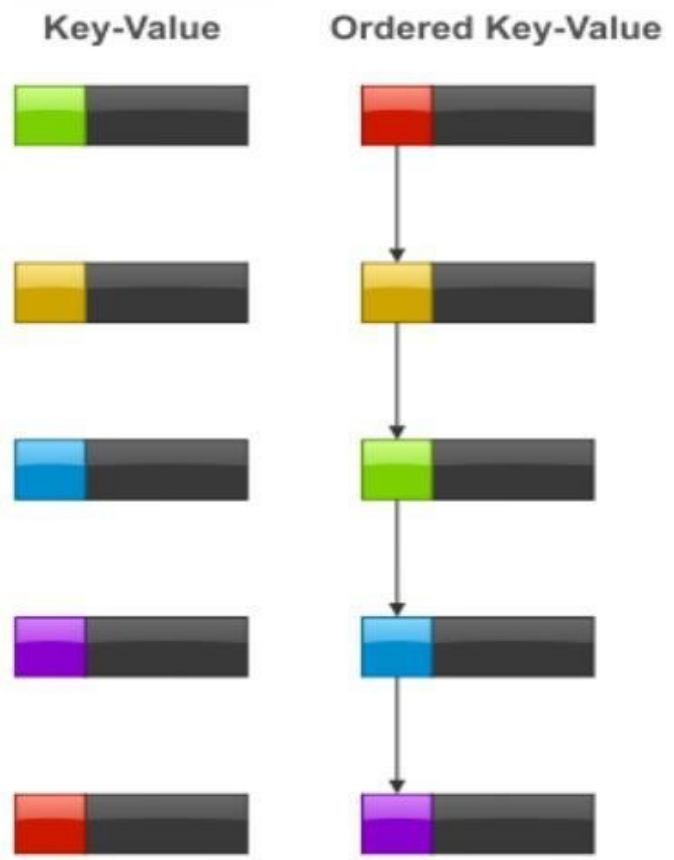
NoSQL Data Models



[http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques]

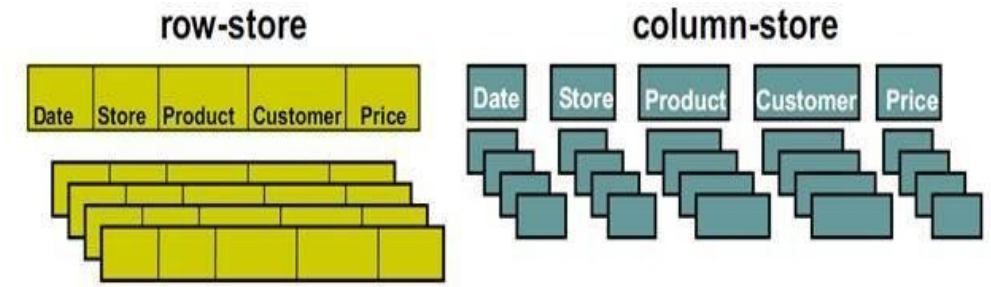
NoSQL Data Models: Key-Value

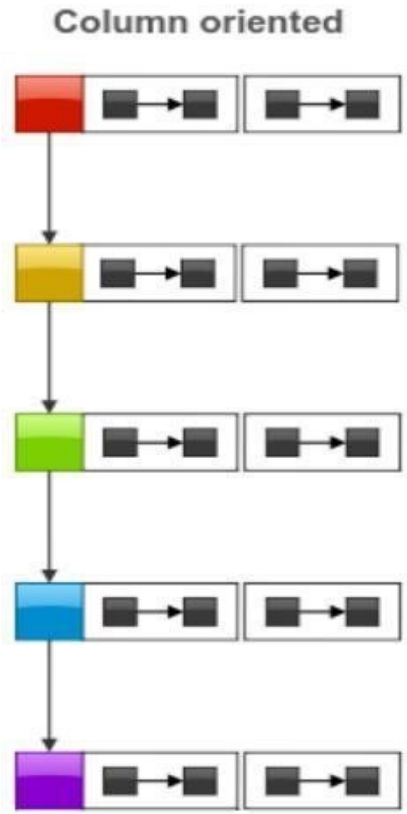
- Collection of key/value pairs.
- Ordered Key-Value: processing over key ranges.
- Dynamo, Scalaris, Voldemort, Riak, …
  - E.g., {Lamborghini : Gallardo, Aventador, Murciélago, Reventón, Diablo, Huracán, Veneno, Centenario}.



NoSQL Data Models: Column-Oriented

- Similar to a key/value store, but the value can have multiple attributes (Columns).
- Column: a set of data values of a particular type.
- Store and process data by column instead of row.
- BigTable, Hbase, Cassandra, …





Row-Oriented Vs. Column-Oriented

rowid	id	first_name	last_name	ssn	salary	dob	title	joined
1001	1	John	Smith	111	101,000	1/1/1991	eng	1/1/2011
1002	2	Kary	White	222	102,000	2/2/1992	mgr	2/1/2012
1003	3	Norman	Freeman	333	103,000	3/3/1993	mkt	3/1/2013
1004	4	Nole	Smith	444	104,000	4/4/1994	adm	4/1/2014
1005	5	Dar	Sol	555	105,000	5/5/1995	adm	5/1/2015
1006	6	Yan	Thee	666	106,000	6/6/1996	mkt	6/1/2016
1007	7	Hasan	Ali	777	107,000	7/7/1997	acc	7/1/2017
1008	8	Ali	Bilal	888	108,000	8/8/1998	acc	8/1/2018

Row-Oriented Database

- Tables are stored as rows in disk
- A single block to read to the table fetches multiple rows with all columns
- More IOS are required to find a particular row in a table scan, but once you find the row you get all columns for that row.

Row-Oriented Database

1001, 1, John, Smith, 111, 101,000, 1/1/1991, eng, 1/1/2011|||
1002,2,Kary,White,222,102,000,2/2/1992,mgr,2/1/2012

1003,3,Norman,Freeman,333,103,000,3/3/1993,mkt,3/1/2013|||
1004,4,Nole,Smith,444,104,000,4/4/1994,adm,4/1/2014

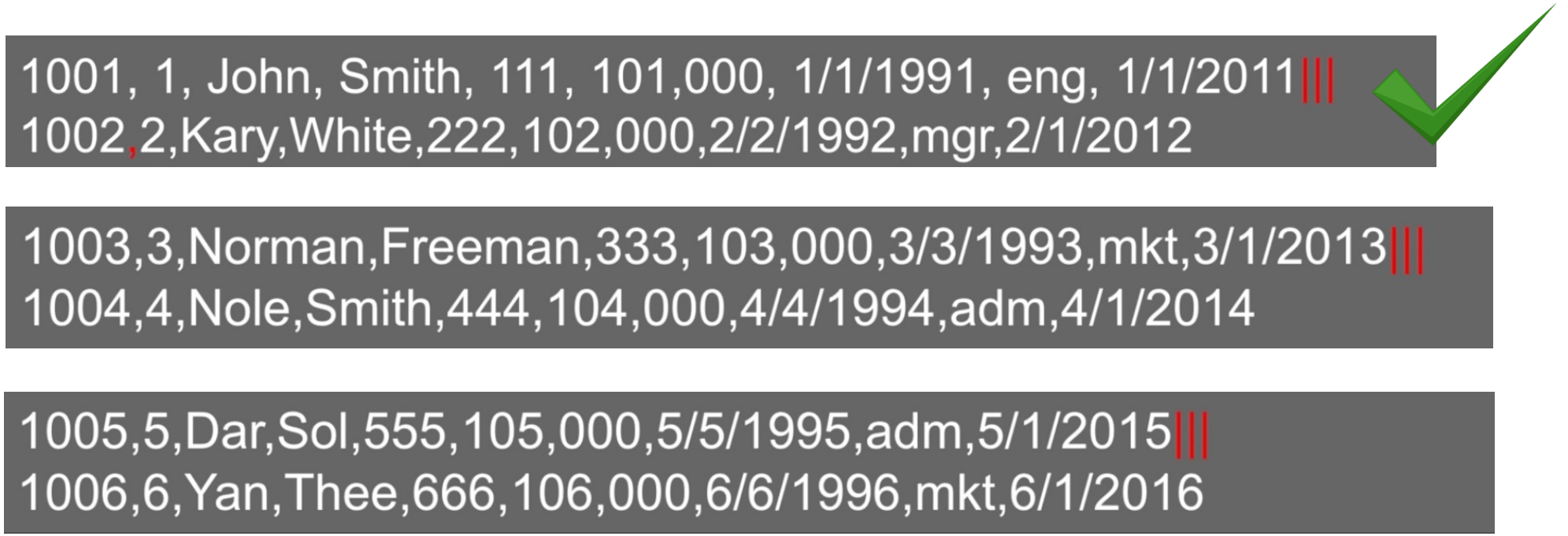
1005,5,Dar,Sol,555,105,000,5/5/1995,adm,5/1/2015|||
1006,6,Yan,Thee,666,106,000,6/6/1996,mkt,6/1/2016

Row-Oriented Database

- Select first_name from emp where ssn=666

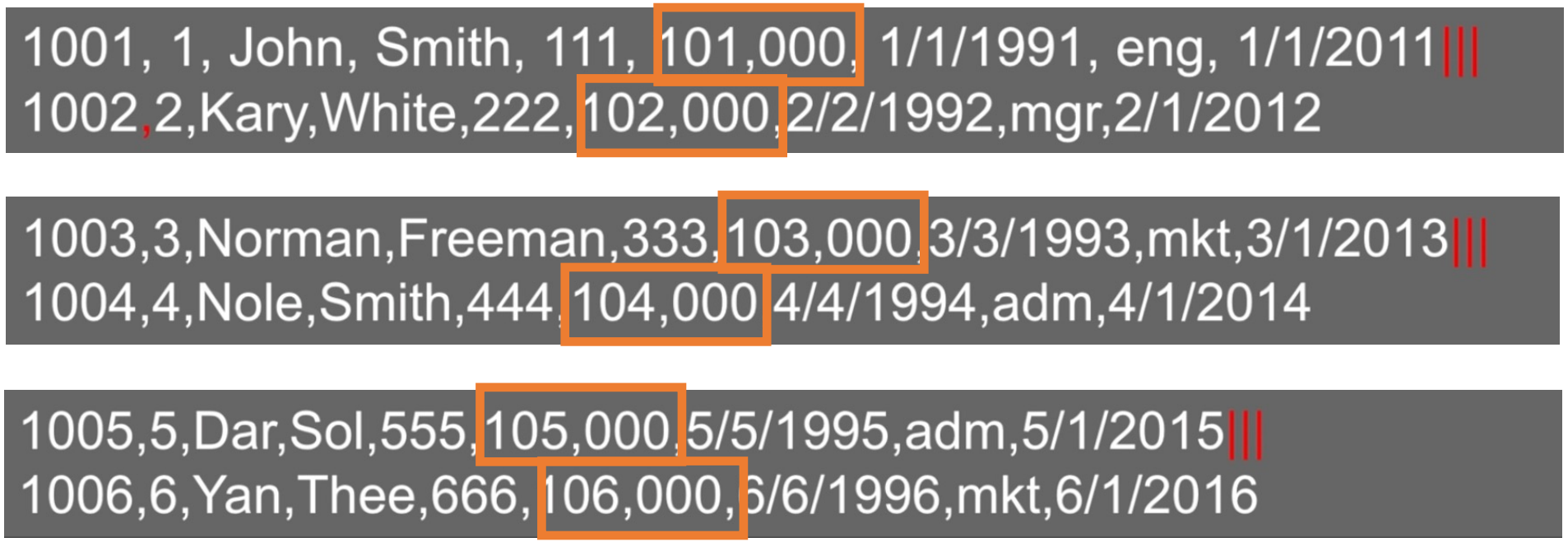
Row-Oriented Database

- Select from emp where id=1



Row-Oriented Database

- Select sum(salary) from emp



Column-Oriented Databases

- Tables are stored as columns first in the disk.
- A single block io read to the table fetches multiple columns with all marching rows.
- Less IOs are required to get more values of a given column, but working with more columns need more IOs.

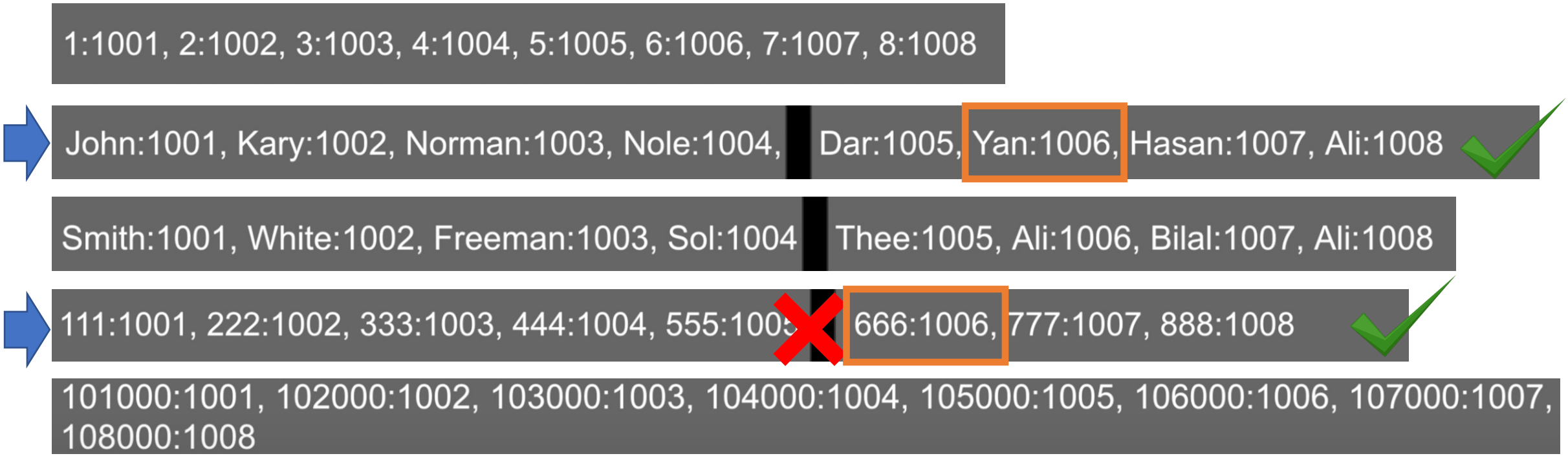
Row-Oriented Vs. Column-Oriented

rowid	id	first_name	last_name	ssn	salary	dob	title	joined
1001	1	John	Smith	111	101,000	1/1/1991	eng	1/1/2011
1002	2	Kary	White	222	102,000	2/2/1992	mgr	2/1/2012
1003	3	Norman	Freeman	333	103,000	3/3/1993	mkt	3/1/2013
1004	4	Nole	Smith	444	104,000	4/4/1994	adm	4/1/2014
1005	5	Dar	Sol	555	105,000	5/5/1995	adm	5/1/2015
1006	6	Yan	Thee	666	106,000	6/6/1996	mkt	6/1/2016
1007	7	Hasan	Ali	777	107,000	7/7/1997	acc	7/1/2017
1008	8	Ali	Bilal	888	108,000	8/8/1998	acc	8/1/2018

Column-Oriented Database

Column-Oriented Database

- Select first_name from emp where ssn=666



Column-Oriented Database

- Select from emp where id=1

Column-Oriented Database

- Select sum(salary) from emp

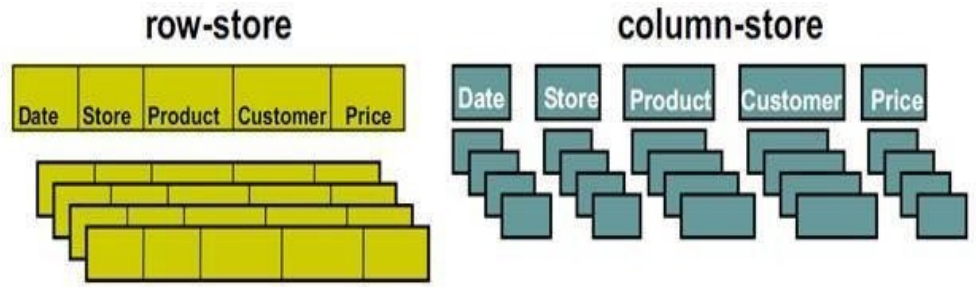
Row-Oriented Vs. Column-Oriented Database

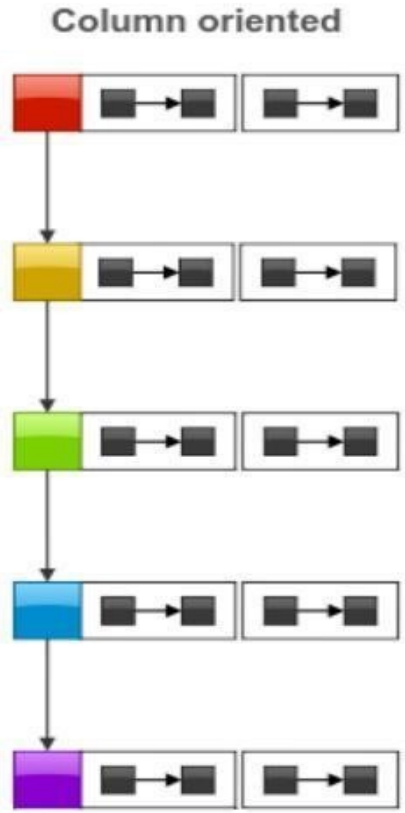
- Row-based
- Optimal for read/write
- OLTP (Online Analytical Processing)
- Compression isn't efficient
- Aggregation isn't efficient
- Efficient queries with multiple columns

- Column-based
- Writes are slower
- OLAP (Online Transaction Processing)
- Compress greatly
- Amazing for aggregation
- Inefficient queries with multiple columns

NoSQL Data Models: Column-Oriented

- Similar to a key/value store, but the value can have multiple attributes (Columns).
- Column: a set of data values of a particular type.
- Store and process data by column instead of row.
- BigTable, Hbase, Cassandra, …

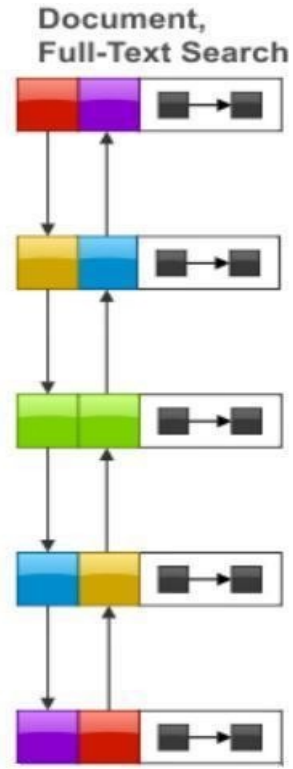




NoSQL Data Models: Document-based

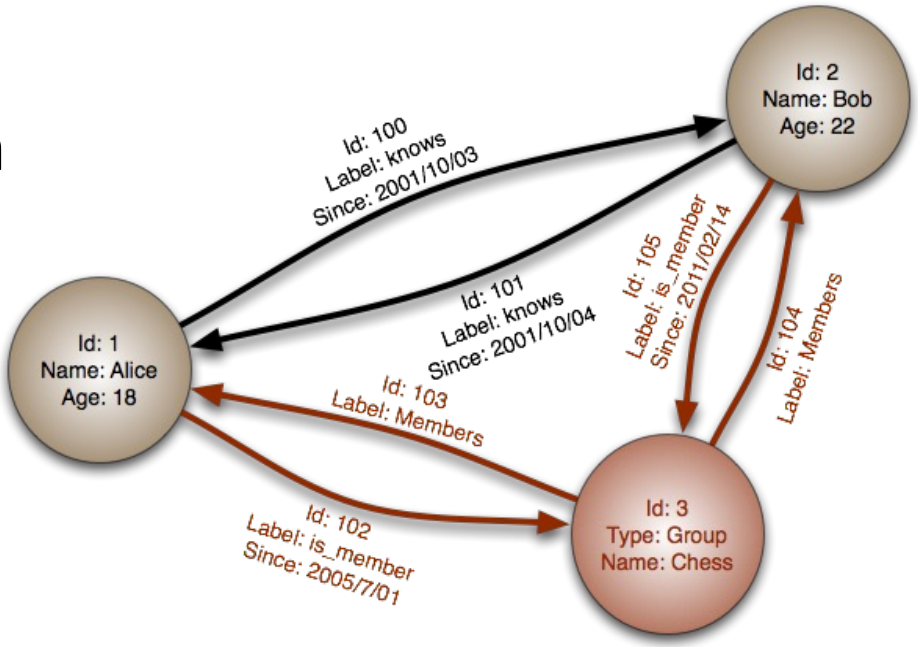
- Similar to a column-oriented store, but values can have complex documents, e.g., XML, YAML, JSON, and BSON.
- CouchDB, MongoDB, …

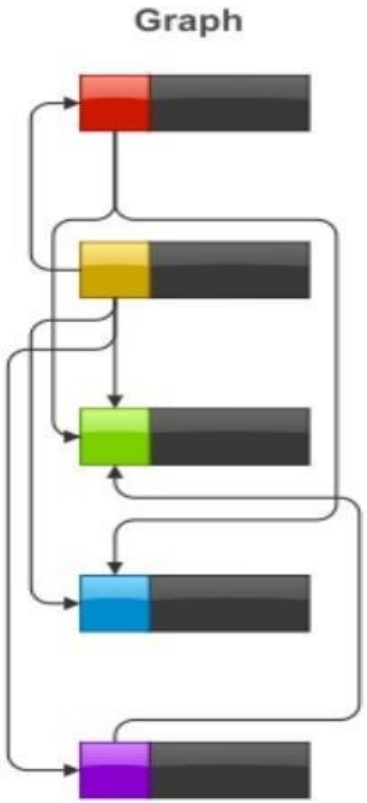
```
{   FirstName: "Bob",
    Address: "5 Oak St.",
    Hobby: "sailing"
 }

 {
   FirstName: "Jonathan",
   Address: "15 Wanamassa Point Road",
   Children: [
        {Name: "Michael", Age: 10},
        {Name: "Jennifer", Age: 8},
   ]
  }
```



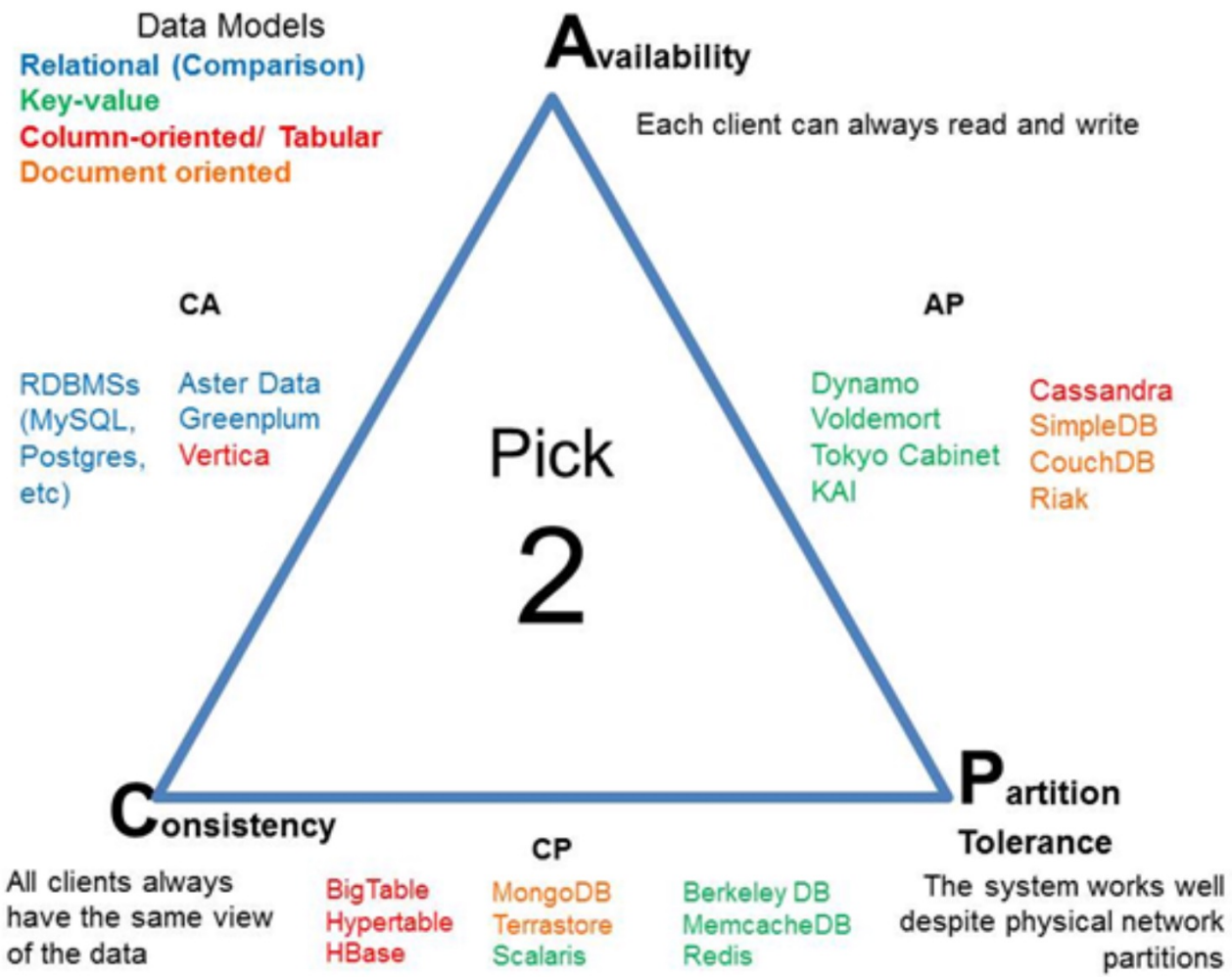
NoSQL Data Models: Graph-based

- Uses graph structures with nodes, edges, and properties to represent and store data.
- Neo4J, InfoGrid, …
- E.g., Fraud detection





Recap


Data Models
Relational (Comparison)
Key-value
Column-oriented/ Tabular
Document oriented
Availability
Each client can always read and write
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick 2
Consistency
All clients always have the same view of the data
CP
BigTable
Hypertable
HBase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance
The system works well despite physical network partitions

Next Topic:

Key-Value Store