Big Data Technologies

NoSQL

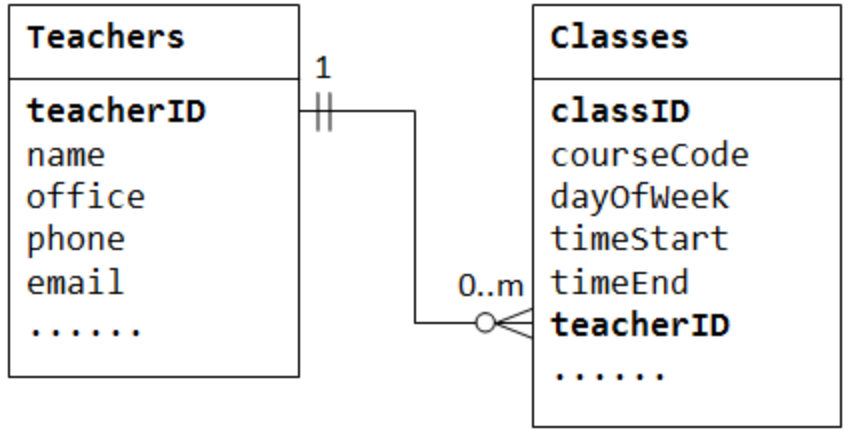
Lionel Fillatre
Polytech Nice Sophia
lionel.fillatre@univ-cotedazur.fr

Outlines

- Why NoSQL?
- CAP Theorem
- NoSQL Solutions
  - Key Value Pair System
  - Document-based System
  - Column-based System
  - Graph Database System
- Conclusion

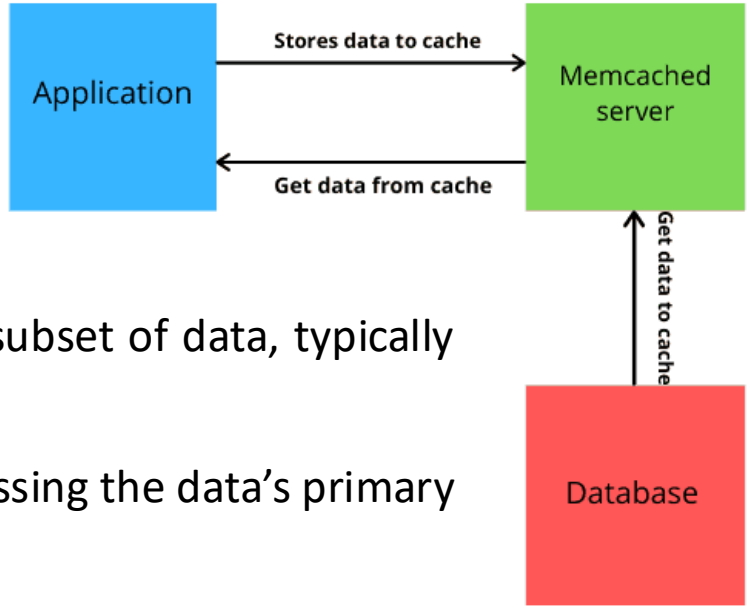
Why NoSQL?

History



- Relational Databases – mainstay of business
- Web-based applications caused spikes
  - Especially true for public-facing e-commerce sites
- Developers are interested in RDBMS with Memcache or integrate other caching mechanisms within the application (ie. Ehcache)
  - Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API call
  - Ehcache is an open source, standards-based cache that boosts performance, offloads your database, and simplifies scalability. It's robust and integrates with other popular libraries and frameworks.

Memory Caching

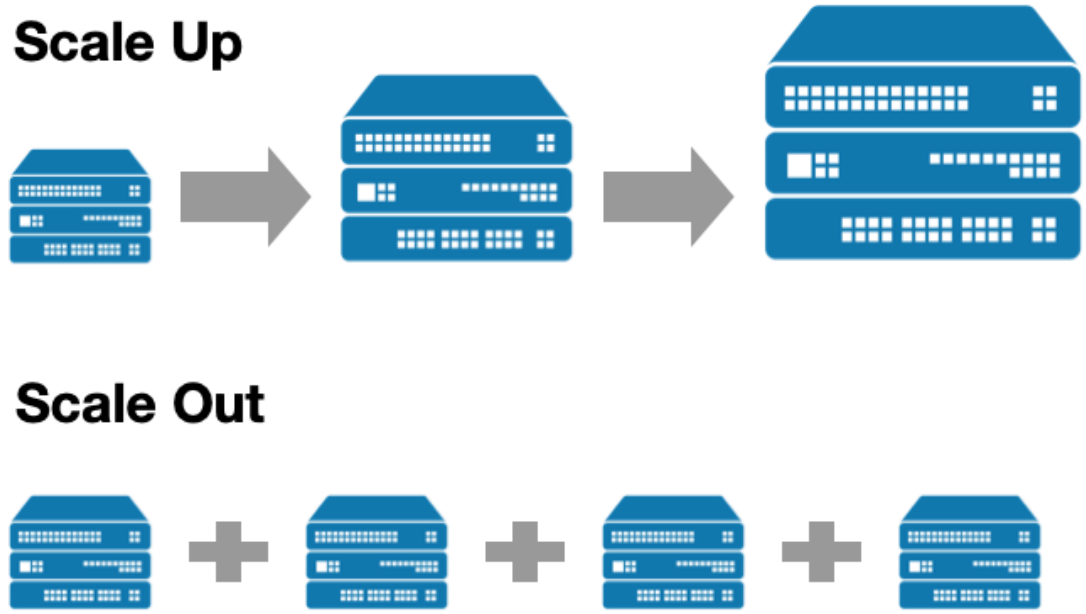


- What is Caching?
  - In computing, a cache is a high-speed data storage layer which stores a subset of data, typically transient in nature
  - Future requests for that data are served up faster than is possible by accessing the data's primary storage location.
  - Caching allows you to efficiently reuse previously retrieved or computed data.

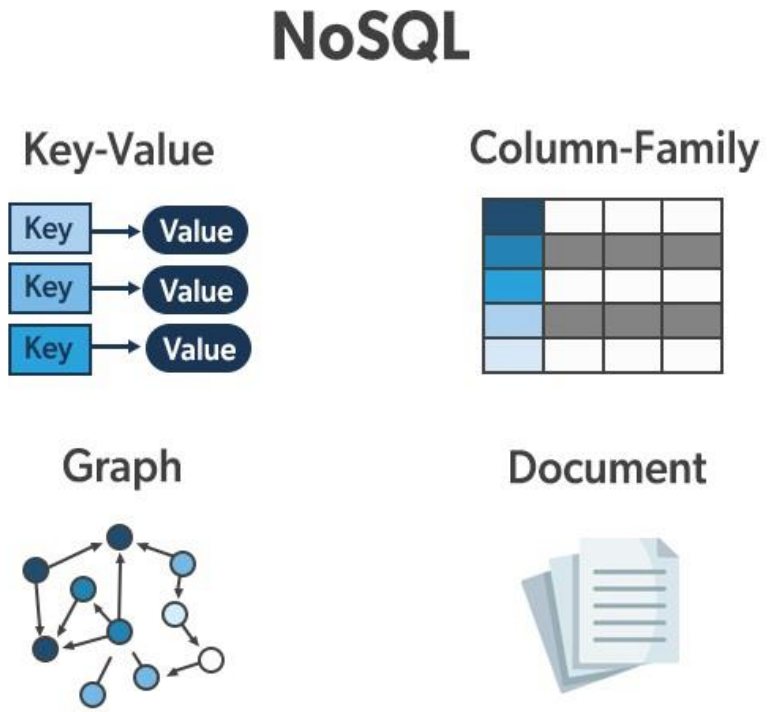
- How does Caching work?
  - The data in a cache is generally stored in fast access hardware such as RAM (Random-access memory) and may also be used in correlation with a software component.
  - A cache's primary purpose is to increase data retrieval performance by reducing the need to access the underlying slower storage layer.
  - Trading off capacity for speed, a cache typically stores a subset of data transiently, in contrast to databases whose data is usually complete and durable.

Scaling Up/Scaling Out

- Issues with scaling up when the dataset is just too big
- RDBMS were not designed to be distributed
- For data storage, an RDBMS cannot be the be-all/end-all
- Began to look at multi-node database solutions
- Known as ‘scaling out’ or ‘horizontal scaling’
- Main approaches include:
  - Master-slave
  - Partitions



What is NoSQL?



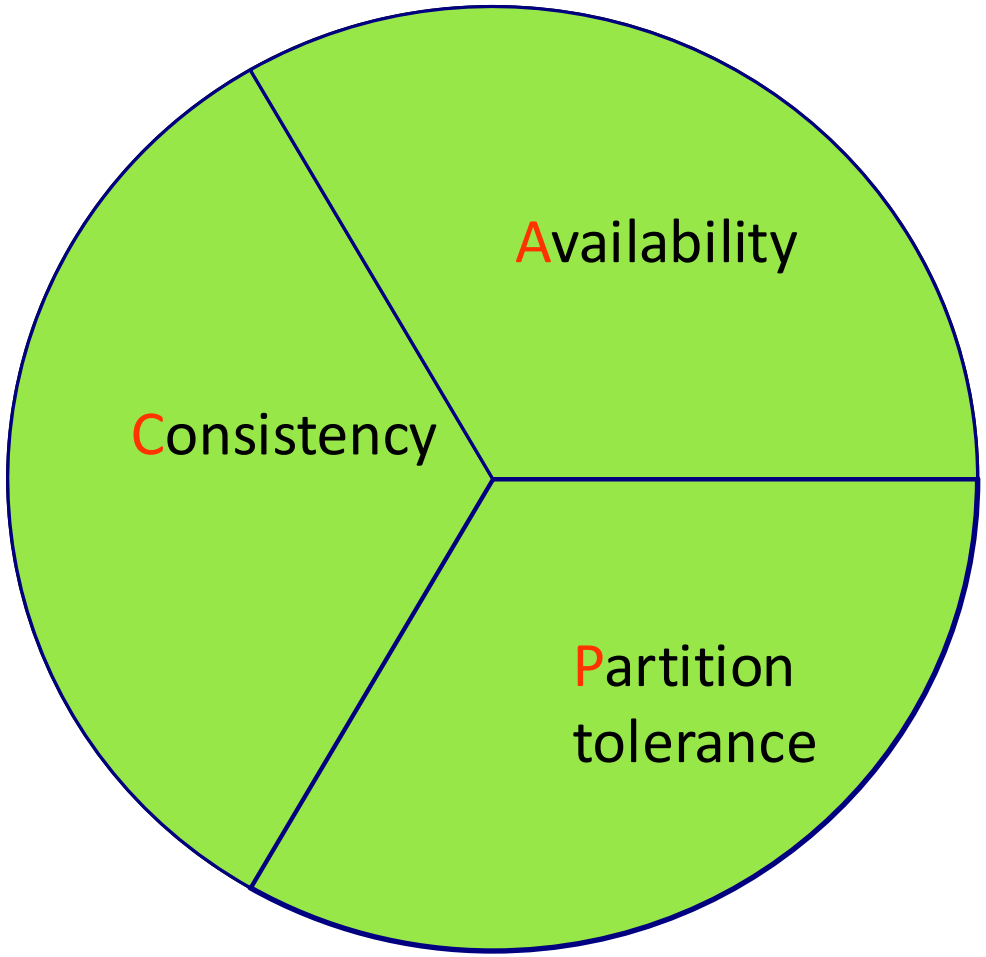
- Stands for **N**ot **O**nly **SQL**
- Class of non-relational data storage systems
- You would use NoSQL for two main reasons: scalability and flexible data model.
- Usually do not require a fixed table schema nor do they use the concept of joins
- All NoSQL offerings relax one or more of the ACID properties ⇒ **CAP theorem**
- Just as there are different programming languages, need to have other data storage tools in the toolbox

The perfect matching

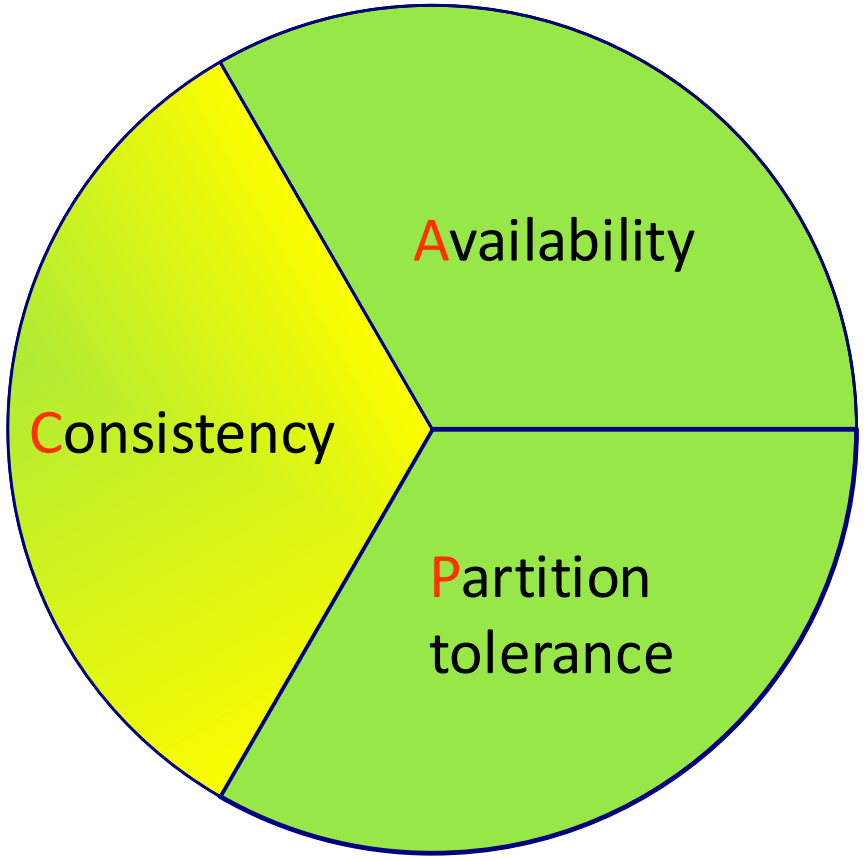
- Three major papers were the seeds of the NoSQL movement
  - BigTable (Google)
  - Dynamo (Amazon)
    - Gossip protocol (discovery and error detection)
    - Distributed key-value data store
    - Eventual consistency
  - CAP Theorem
- Large datasets, acceptance of alternatives, and dynamically-typed data has come together in a perfect storm
- Not a backlash/rebellion against RDBMS
- SQL is a rich query language that cannot be rivaled by the current list of NoSQL offerings
- Relational databases are very mature
  - 40+ years versus 10+ years

CAP Theorem

The CAP Theorem



Consistency



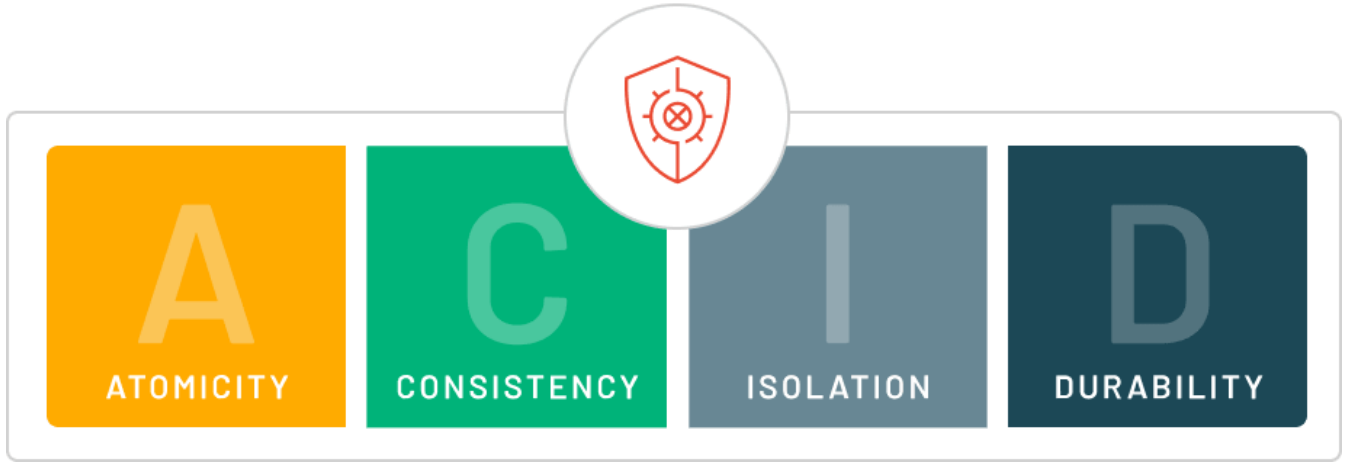
- Once a writer has written, all readers will see that write
- All nodes see the same data at the same time
- Every read receives the most recent write or an error

Consistency

- Two kinds of consistency:
  - strong consistency – ACID (Atomicity Consistency Isolation Durability)
  - weak consistency – BASE (Basically Available Soft-state Eventual consistency)

ACID Transactions

- A DBMS is expected to support “ACID transactions,” process that are:
  - Atomic : Either the whole process is done or none is.
  - Consistent : Database constraints are preserved.
  - Isolated : It appears to the user as if only one process executes at a time.
  - Durable : Effects of a process do not get lost if the system crashes.



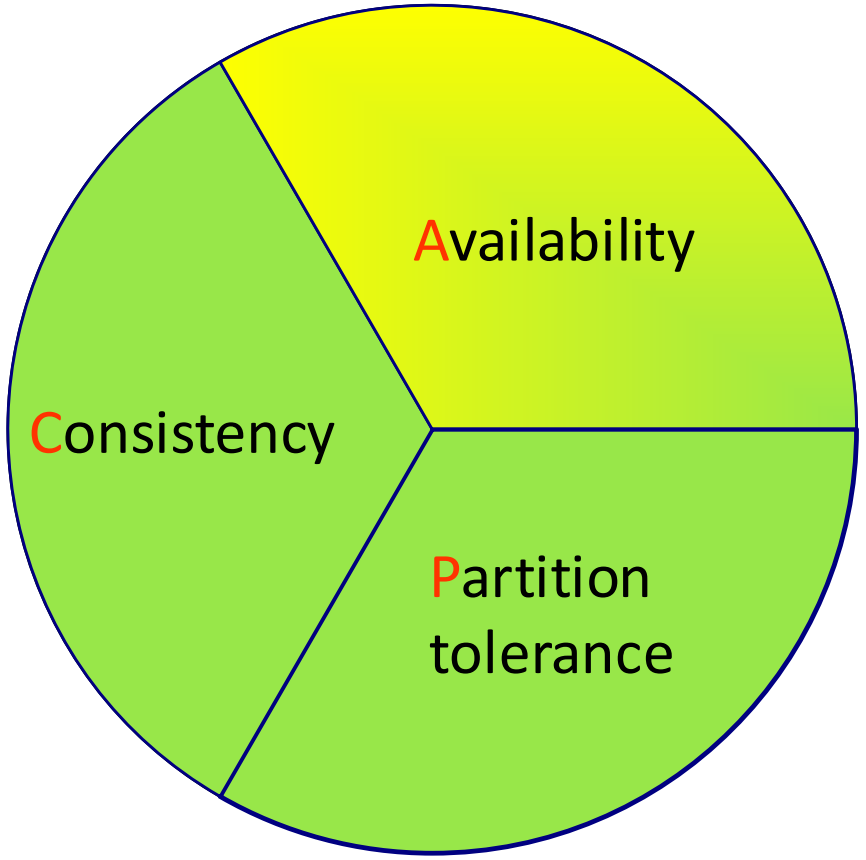
NoSQL databases relaxed the ACID properties ⇒ BASE

- Basically Available: there will be a response to any request. But, that response could still be 'failure' to obtain the full requested data or the data may be in an inconsistent or changing state.

- Soft state: Usually in computer science, the term soft state means data will expire if it is not refreshed. In NoSQL operations, it refers to the fact that data may eventually be overwritten with more recent data. This property overlaps with the eventual consistency. The state of the system could change over time, so even during times without input there may be changes going on due to 'eventual consistency'.

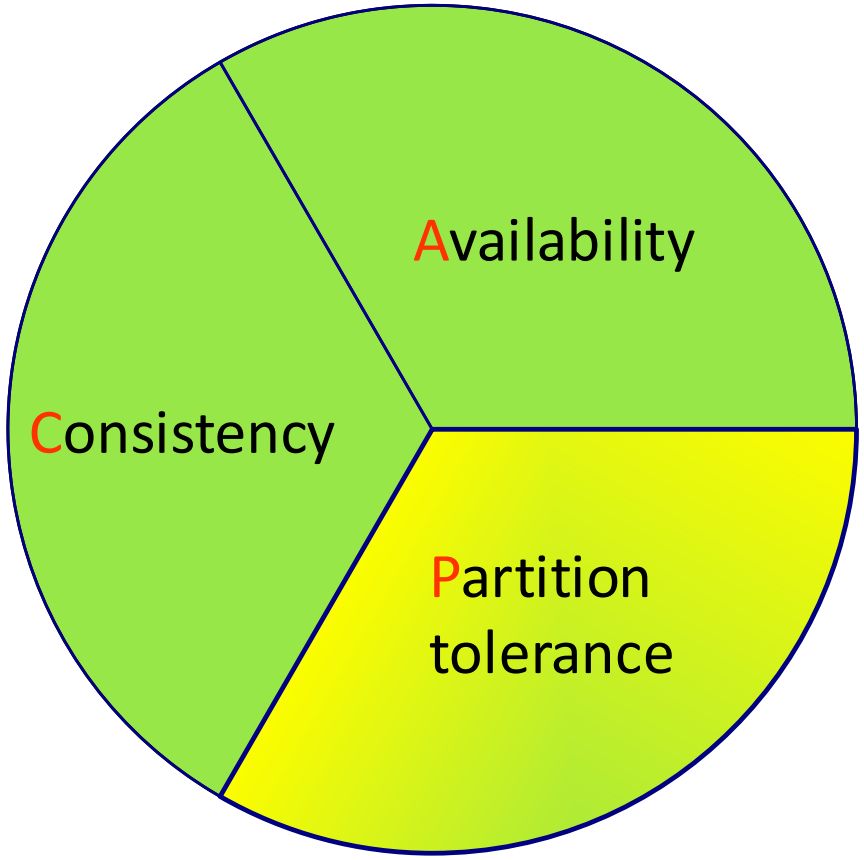
- Eventual consistency: The system will eventually become consistent once it stops receiving input. The data will propagate to everywhere it should sooner or later, but the system will continue to receive input and is not checking the consistency of every transaction before it moves onto the next one.

Availability

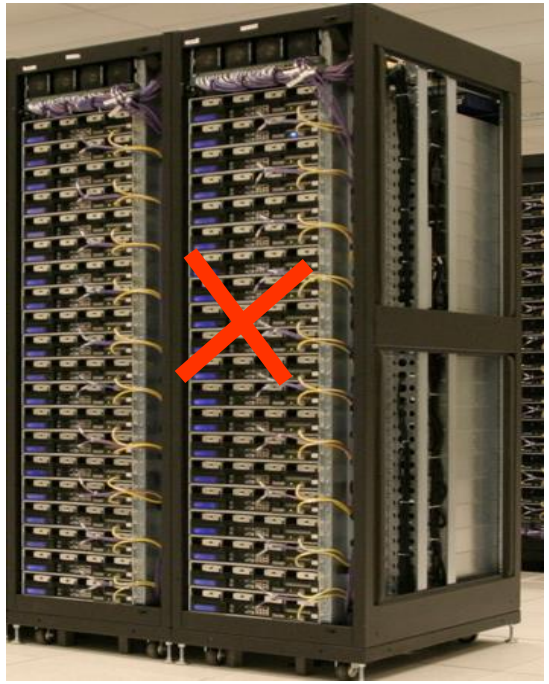


- Every request receives a (non error) response – without guarantee that it contains the most recent write

Partition tolerance



- The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.

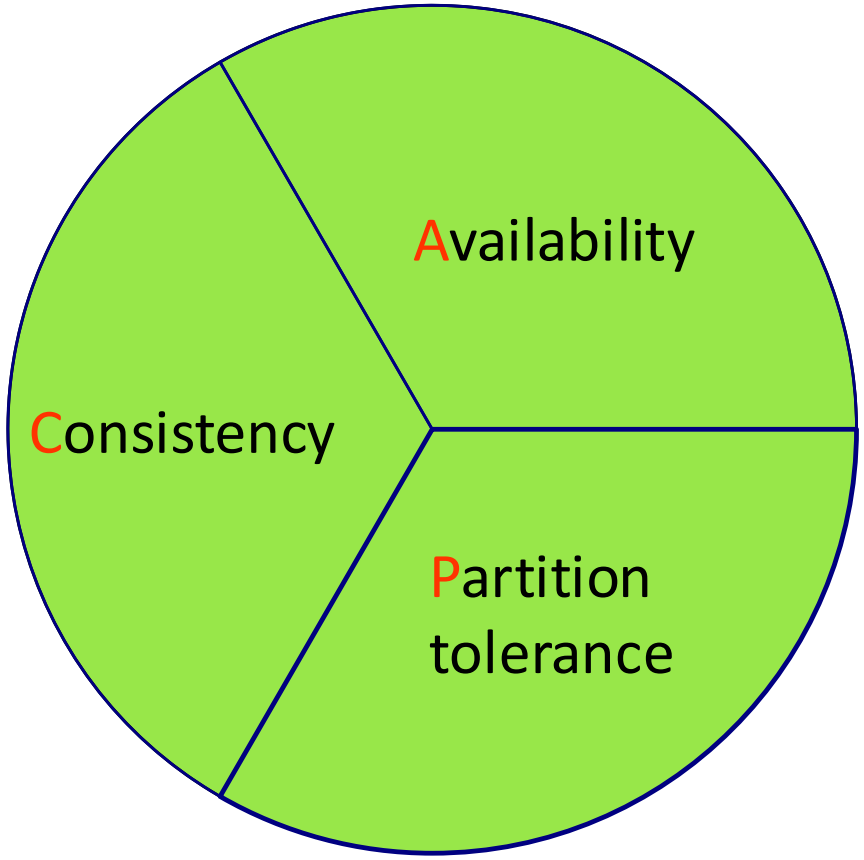


Failure is the rule

- Amazon:
  - Several thousands of disks fail over per year (25 disks per day)

- Sources of failures are numerous
  - Hardware (disk)
  - Network
  - Power
  - Software
  - Software and OS updates.

The CAP Theorem



Theorem: You can have at most **two** of these properties for any shared-data system

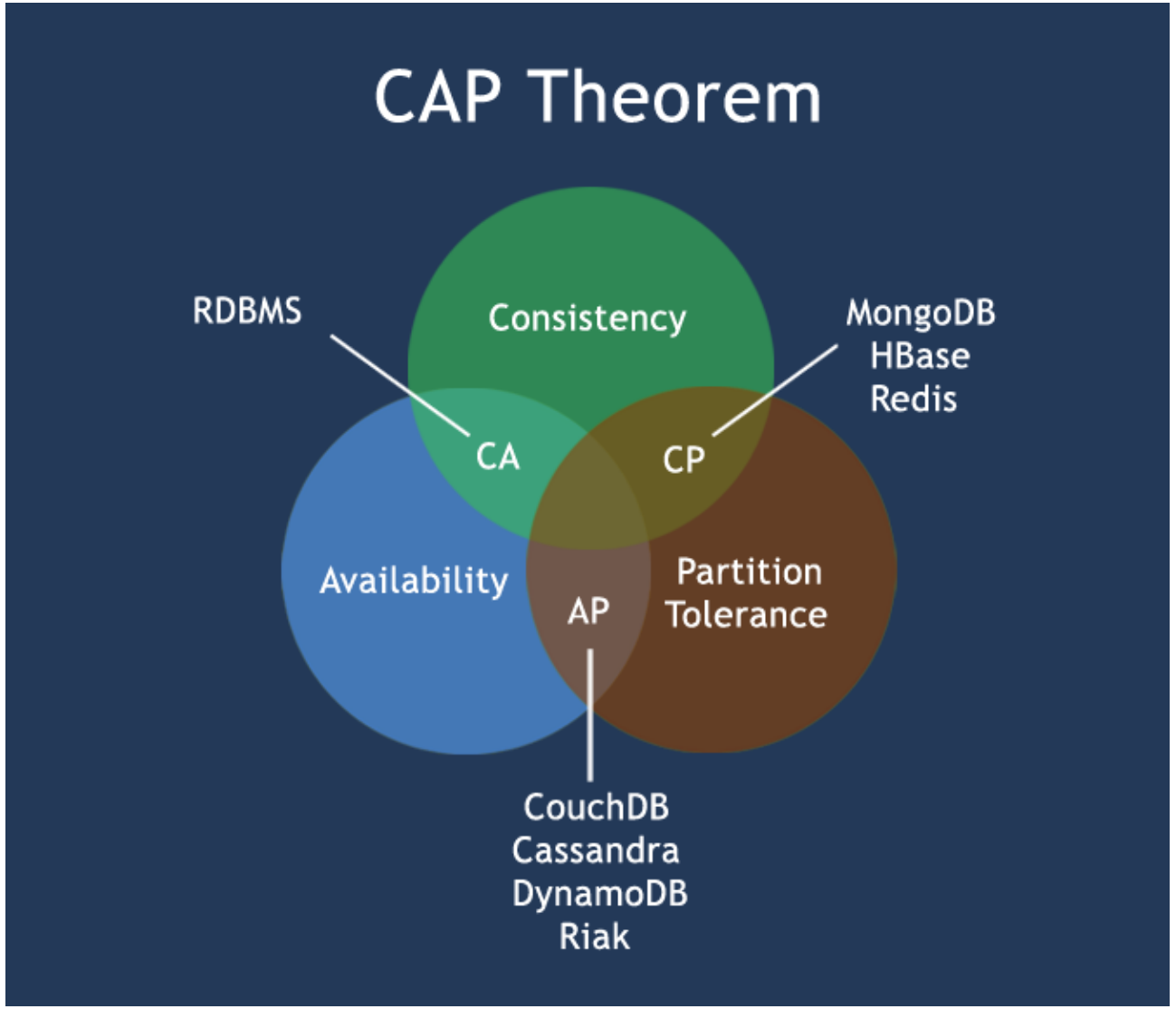
The CAP Theorem

- Distributed systems must be partition tolerant (P), so we have to choose between Consistency and Availability.

- When choosing consistency over availability, the system will return an error or a time-out if particular information cannot be guaranteed to be up to date due to network partitioning.

- When choosing availability over consistency, the system will always process the query and try to return the most recent available version of the information, even if it cannot guarantee it is up to date due to network partitioning.

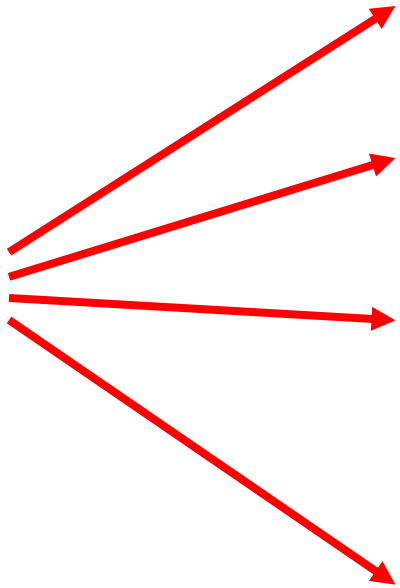
The CAP Theorem



NoSQL Solutions

7 Popular NoSQL Databases

Our lab



	Type	Primary Model	Query Language	Transactions	Scalability
MongoDB	Document	Document Store	MongoDB Query Language	Yes (ACID for single documents)	Horizontal
Apache Cassandra	Wide-column	Wide Column Store	CQL (Cassandra Query Language)	Limited ACID	Horizontal
Redis	Key-value / Data structure store	In-memory Data Store	Redis commands	Transactions with optimistic locking	Master-slave replication
Couchbase	Document	Document / Key-Value	N1QL, SQL++	ACID (on document level)	Horizontal
Neo4j	Graph	Graph Database	Cypher	ACID Transactions	Horizontal
Amazon DynamoDB	Key-value / Document	Key-Value and Document Store	AWS proprietary	ACID with limitations	Managed, Horizontal
ArangoDB	Multi-model	Document, Graph, Key-value	AQL (ArangoDB Query Language)	ACID	Horizontal

https://www.bairesdev.com/blog/nosql-databases/

Six key features of NoSQL

1. Scale horizontally “simple operations”
   - key lookups, reads and writes of one record or a small number of records, simple selections
2. Replicate/distribute data over many servers
3. Simple call-level interface (contrast with SQL)
4. Weaker concurrency model than ACID
5. Efficient use of distributed indexes and RAM
6. Flexible schema

Different Types of NoSQL Systems

- Distributed Key-Value Systems - Lookup a single value for a key
  - A few examples: Amazon's DynamoDB, Oracle NoSQL Database, Redis
- Document-based Systems - Access data by key or by search of "document" data.
  - A few examples: CouchDB, MongoDB
- Column-based Systems
  - A few examples: Google's BigTable, HBase, Facebook's Cassandra
- Graph-based Systems - Use a graph structure
  - A few examples: Google's Pregel, Neo4j

⇒ Use different types for different types of applications

Key Value Pair System

Key-Value Pair (KVP) Stores

- Access data (values) by strings called keys.
- Data has no required format – data may have any format
- Extremely simple interface
  - Data model: (key, value) pairs
  - Basic Operations: Insert(key,value), Fetch(key),Update(key), Delete(key)
- Implementation: efficiency, scalability, fault-tolerance
  - Records distributed to nodes based on key
  - Replication
  - Single-record transactions, “eventual consistency”
- Example systems
  - Amazon Dynamo, Oracle NoSQL Database, …

Storage

- "Value" is stored as a "blob"
  - Without caring or knowing what is inside
  - Application is responsible for understanding the data
- In simple terms, a NoSQL Key-Value store is a single table with two columns: one being the (Primary) Key, and the other being the Value.

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Example of unstructured data for user records

Key: 1	ID: sj	First Name: Sam	
Key: 2	Email: jb@gmail.com	Location: London	Age: 37
Key: 3	Facebook ID: jkirk	Password: xxx	Name: James

Each record may have a different schema

Document-based Systems

Document Stores

- Like Key-Value Stores, except Value is a “Document”
  - Data model: (key, “document”) pairs
  - Document format: JSON, XML, other semi-structured formats
  - Basic operations: Insert(key,document),
    - Fetch(key), Update(key),
    - Delete(key)
  - Also Fetch() based on document contents
- Example systems
  - CouchDB, MongoDB, SimpleDB, …
- Document stores
  - Store arbitrary/extensible structures as a “value”

Example record

- Records within a single table can have different structures.
- An example record from Mongo, using JSON format, might look like

```
{
"_id" : ObjectId("4fccbf281168a6aa3c215443"),
"first_name" : "Thomas",
"last_name" : "Jefferson",
"address" : {
        "street" : "1600 Pennsylvania Ave NW",
        "city" : "Washington",
        "state" : "DC"
}
}
```

Embedded object

- Though records are called documents, they are not documents in the sense of a word processing document, although you can store binary data (using BSON format) in any of the fields in the document.
- You can also modify the structure of any document on the fly by adding and removing members from the document, either by reading the document into your program, modifying it and re-saving it, or by using various update commands.

Example of relational scheme in MongoDB

Relational scheme

ID	first_name	last_name	cell	city	year_of_birth	location_x	location_y
1	'Mary'	'Jones'	'516-555-2048'	'Long Island'	1986	'-73.9876'	'40.7574'

ID	user_id	profession
10	1	'Developer'
11	1	'Engineer'

ID	user_id	name	version
20	1	'MyApp'	1.0.4
21	1	'DocFinder'	2.5.7

ID	user_id	make	year
30	1	'Bentley'	1973
31	1	'Rolls Royce'	1965

MongoDB document

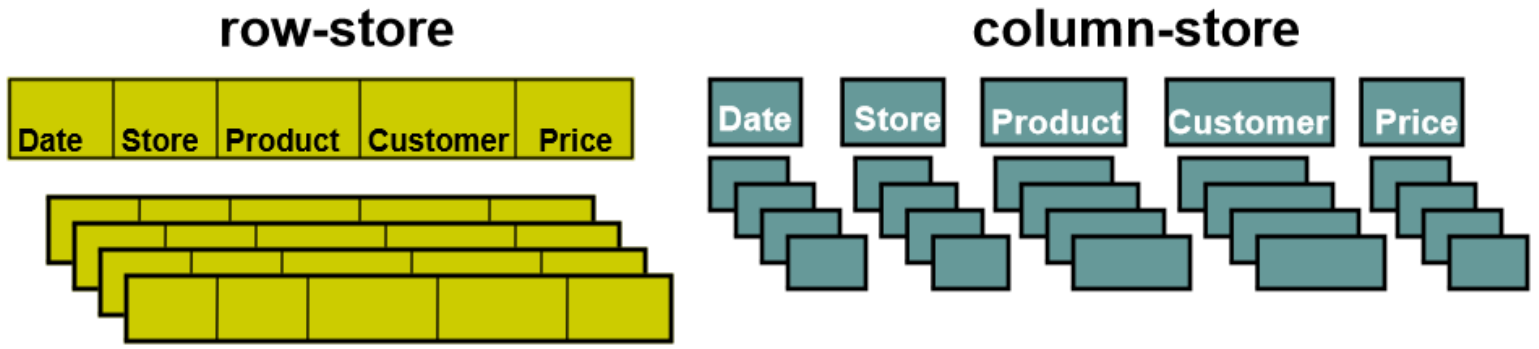
```
{
first_name: "Mary",
last_name: "Jones",
cell: "516-555-2048",
city: "Long Island",
year_of_birth: 1986,
location: {
        type: "Point",
        coordinates: [-73.9876, 40.7574]
},
profession: ["Developer", "Engineer"],
apps: [
 { name: "MyApp",
   version: 1.0.4 },
 { name: "DocFinder",
   version: 2.5.7 }
],
cars: [
   { make: "Bentley",
     year: 1973 },
   { make: "Rolls Royce",
     year: 1965 }
 ]
}
```

https://programmerprodigy.code.blog/2021/06/28/introduction-to-document-value-data-store-along-with-use-case/

Column-based System

Column-based Stores (aka Wide-Column Stores)

- Based on Google's BigTable store.
- What is a column-based store? - Data tables are stored as sections of columns of data, rather than as rows of data.



+ easy to add/modify a record

- might read in unnecessary data

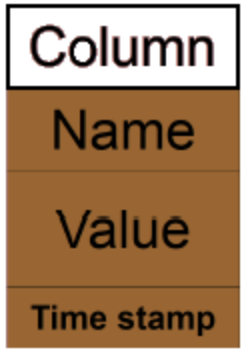
+ only need to read in relevant data

- tuple writes require multiple accesses

=> suitable for read-mostly, read-intensive, large data repositories

Data Model: Column

- Columns
  - The column is lowest/smallest instance of data.
  - It is a tuple that contains a name, a value and a timestamp.



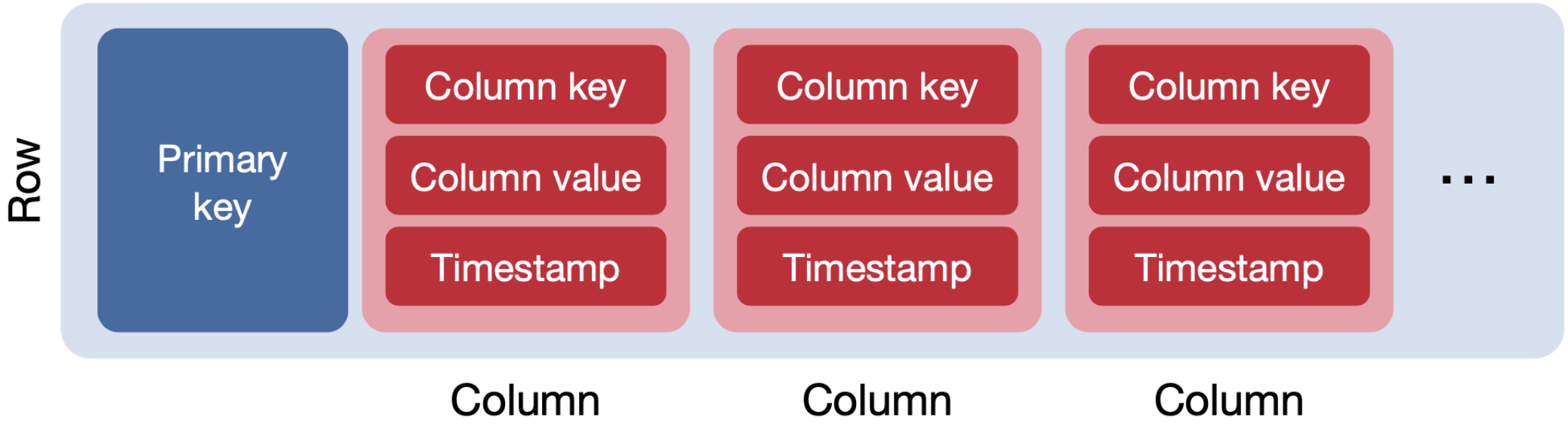
- The column is used as a store for the value, and has a timestamp that is used to differentiate the valid content from stale ones.

- According to the CAP theorem, distributed data stores cannot guarantee consistency - availability is a more important issue.

- Therefore, the data store or the application will use the timestamp to find out which of the stored values in the backup nodes are up-to-date.

Data Hierarchy: Row

- Row: Ordered collection of columns identified by a row key
- Each row can have a different – and almost unlimited – number of columns
- Columns are often sparse: two different rows may share just a few columns among a very large family of columns



Graph Database System

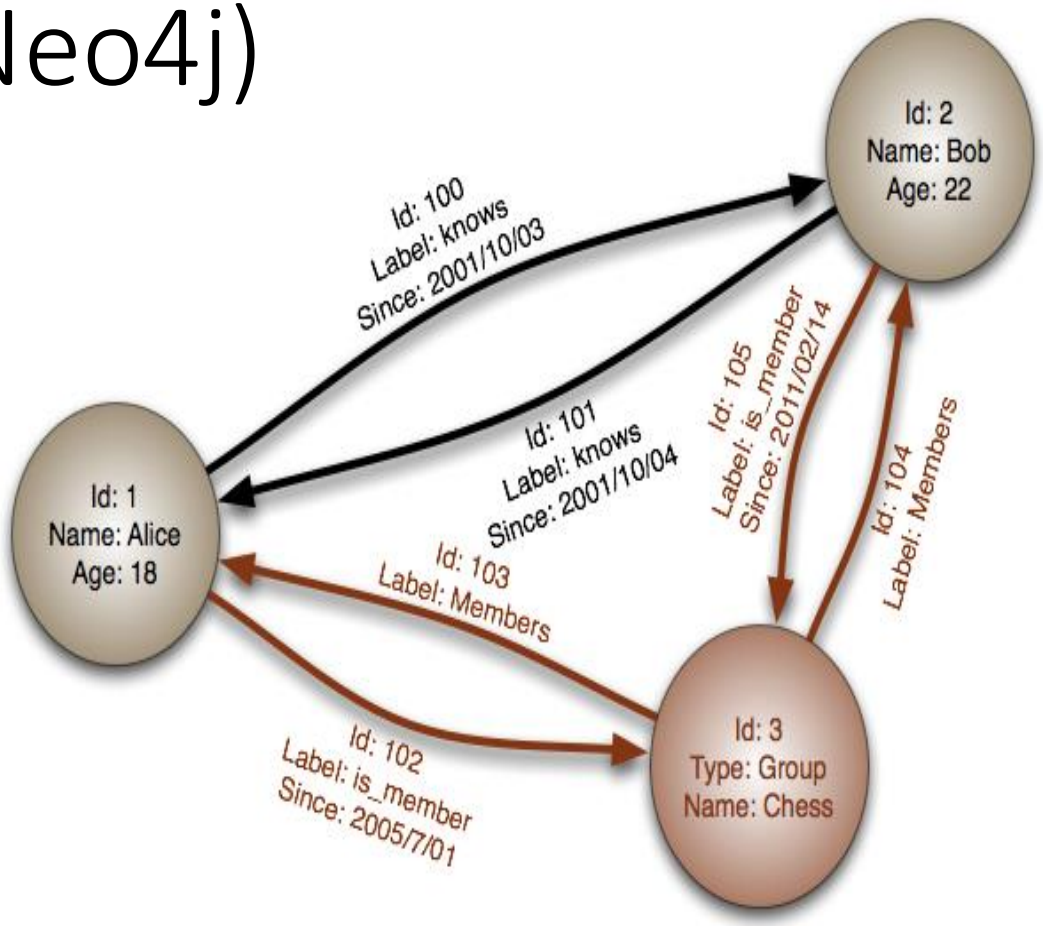
Why Graph Database?

- Most NoSQL databases store sets of disconnected aggregates. This makes it difficult to use them for connected data and graphs.
- One well-known strategy for adding relationships to such stores is to embed an aggregate's identifier inside the field belonging to another aggregate - effectively introducing foreign keys.
- But this requires joining aggregates at the application level, which quickly becomes prohibitively expensive.
- In general, graph databases are useful when you are more interested in relationships between data than in the data itself:
    - For example, in representing and traversing social networks, generating recommendations, or conducting forensic investigations (e.g. pattern detection).

Graph Database Systems

- Graph databases are based on graph theory, and employ nodes, edges, and properties:
  - **Nodes** represent entities such as people, businesses, accounts, or any other item to be tracked.
    - They are roughly the equivalent of the record, relation, or row in a relational database, or the document in a document database.
  - **Edges**, also termed graphs or relationships, are the lines that connect nodes to other nodes; they represent the relationship between them.
    - Edges are the key concept in graph databases, representing an abstraction that is not directly implemented in other systems.
  - **Properties** are germane information that relate to nodes.
    - For example, if Wikipedia were one of the nodes, it might be tied to properties such as website, reference material, or word that starts with the letter w, depending on which aspects of Wikipedia are germane to a given database.

Examples (Neo4j)



CREATE

(a {Id: 1, Name: 'Alice', Age: 18})

-[:IS_MEMBER {Id: 102, Label: 'is_member', Since: '2005/7/01'}]->

(c {Id: 3, Type: ' Group', Name: 'Chess'})

Conclusion

Conclusion

- NoSQL: an alternative to RDMS for Big Data processing

- Four main kinds of NoSQL systems
  - Key-Value Pair
  - Document-based
  - Column-based
  - Graph database

- Several types of NoSQL solutions exist: choose the most relevant one for you!