# SQL x NoSQL

Main differences, Use Cases, Integration and Example

Fernando Durier
28/12/2017

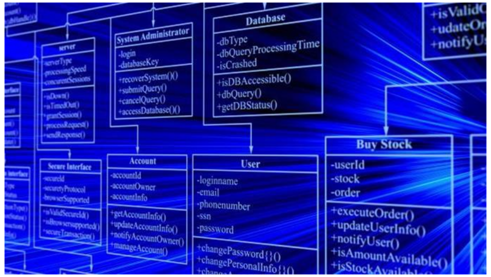
## Motivation:

- Nowadays Data is being constantly generated by different systems, in different formats and in great proportion.
- The RDMS can’t handle this big, fast and flexible kind of data. Nor is being able to scale to this big data proportion.
- New solutions are being created in order to handle such volume of schemaless data.

## Relational Databases

# Relational Databases:

- Features: Databases which stores their data in a “linked” format called tables (formally relations). All operations in a database have closure, in other words they process relations as inputs and generate another set of relations or an empty relation as output.
- Types: Object oriented, column organized, row organized (traditional), lite ones, etc …
- Properties: They operate under the ACID properties which are:
  - Atomicity: Every transaction shall be fully executed, or else be discarded.
  - Consistency: Can be seen as a result of closure, as before and after all transactions the database schema and properties will not be violated.
  - Isolation: transactions may occur in parallel and they will not affect one another.
  - Durability: data present in the database will be immutable until a new transaction happens.



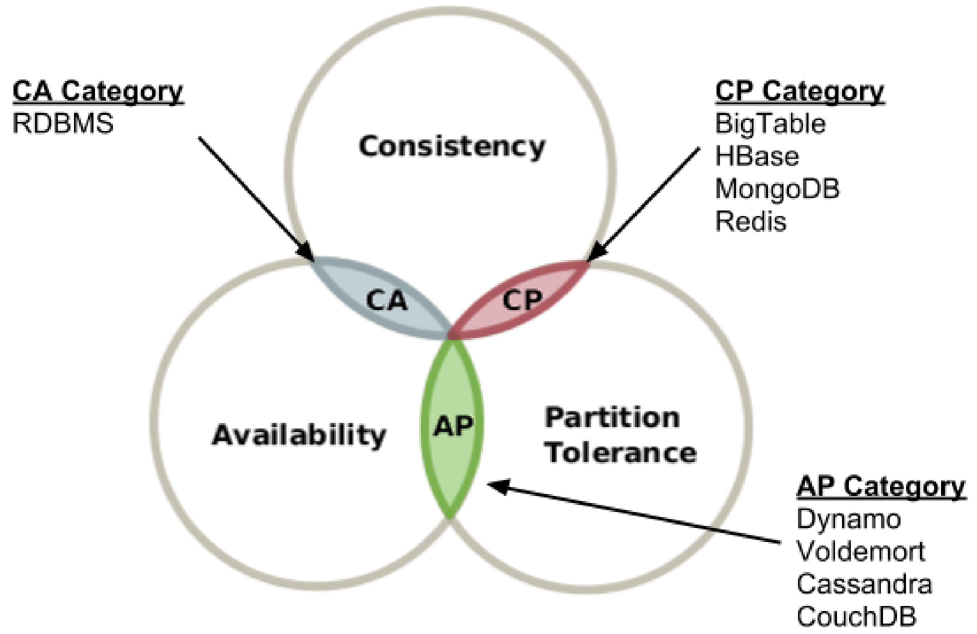
# Relational Databases:

- Pros:
  - Rigid Data Correctness
  - Easy pre-built analytics basic operations
  - Query Language for CRUD operations
  - Bulky
  - Relational Storage Operations
  - Many Layers of Security and Data Access
- Cons:
  - Can’t Scale due to its bulkiness
  - Need an adequate heavily monitored, secure and available environment
  - Inflexible / Schema oriented
- Use Cases:
  - Scientific Papers Repositories
  - Sales Systems
  - Bank Systems
  - ERP

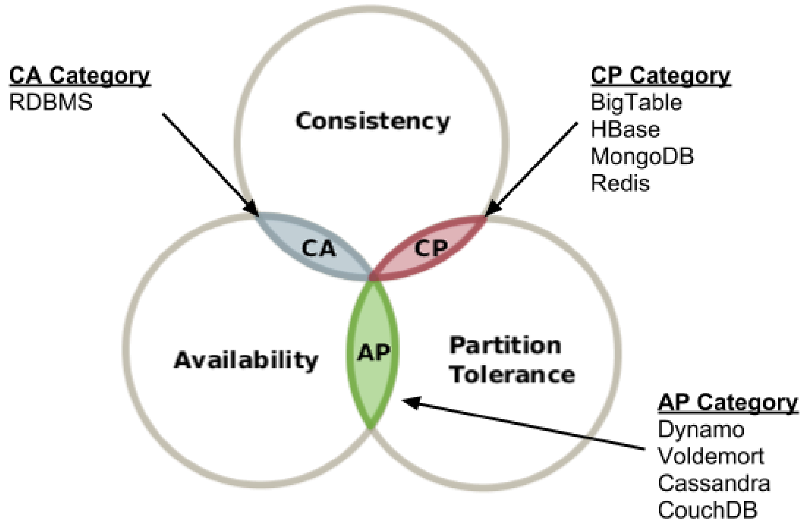
## CAP Theorem

# CAP Theorem:

- Distributed Systems' theorem
- For classic write/read systems
- Heavily used to determine the adequate database to an IT project

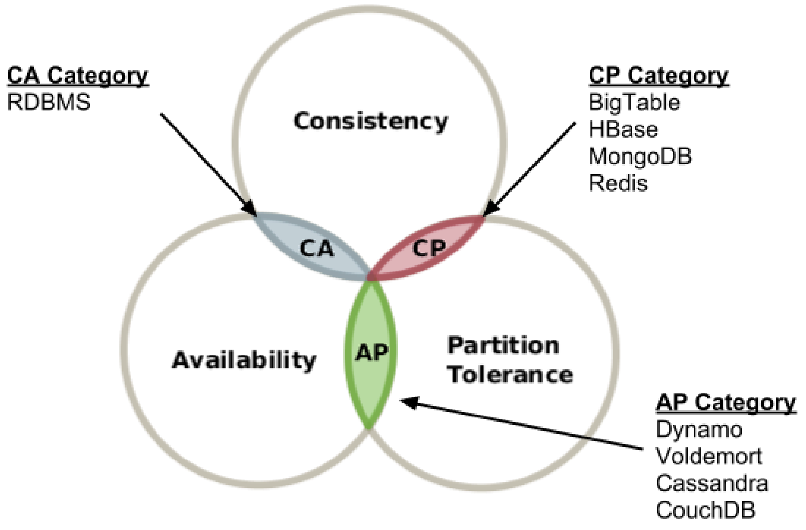


# CAP Theorem:



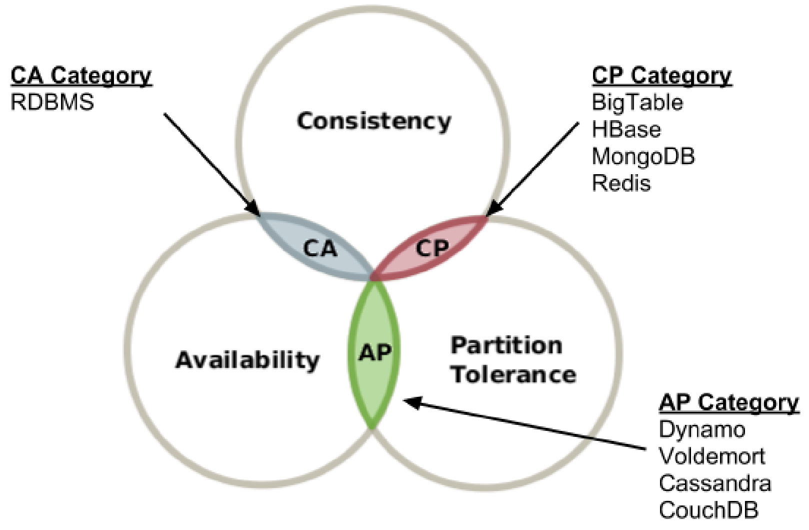
- Consistency: Guarantees that all data inside the database is consistent and synchronized between the database’s partitions before and after transactions.
- Availability: Guarantees that all partitions/clusters/nodes are “queryable” at all time, and will return a response to any request.
- Partition Tolerance: Guarantees that the database will be functional even if one of its partitions is down.

## Why use this ?



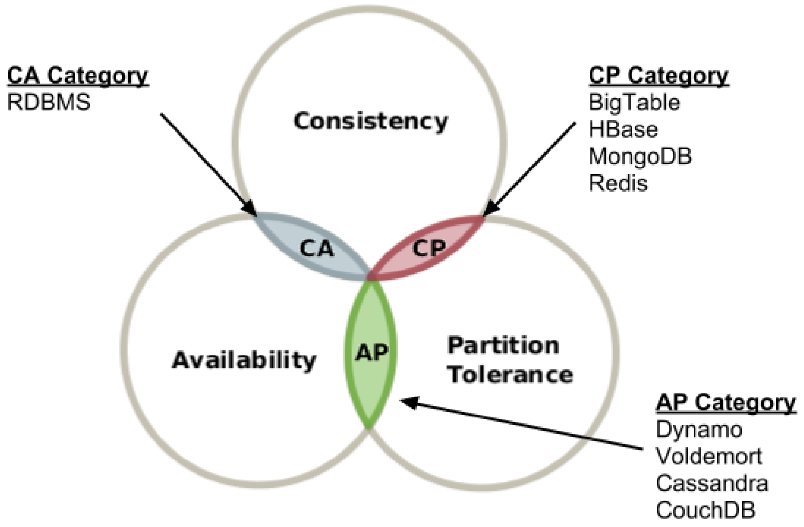
- Normally, the database systems can attend to just two of those subsets at a time.
- Attending to the three of them would be extremely computationally costly, or even impossible for the technology we have today.

## Why just 2 ?



- Partition tolerance implies in database sharding through a cluster of servers.
- Consistency implies in update the data present in all nodes and partitions before a new transaction happens. Comparing with the ACID properties would be to guarantee durability and consistency altogether, as the schema and data from this database would be synchronized and equal to every user before they perform any transaction.

# How to get 3 ?



- In order to attend the 3 requirements, it would be necessary to have a structure capable of manage N different partitions and also it would need to be able to guarantee that all those requests to each of those nodes would be answered, but, before each of those “answers” the system would be synchronized, consistent and available, no matter how many operations are being played.

## NoSQL

# Non Relational Databases:

- Features: A new proposal to store data. They are not substitutes of relational databases, they are just alternatives. They store data in non conventional/relational way, in a new flexible and scalable format.
- Types: Document-Oriented, Big Table, Graph-Oriented, Column-Oriented, Key Value, Hyperdimensional, etc …
- Properties:
  - High Partition Tolerance;
  - Consistency and Durability questionable;
  - Very Practical for the needs imposed by data generation nowadays;
  - Don’t have great server requirements;
  - Cloud Friendly;
  - Schemaless;
  - Non Resource Blocking;
  - Developer Friendly;

# Non Relational Databases:

- Pros:
  - Flexible Schema, in fact, schemaless
  - High Scalability
  - Easy connection between any electronic component
  - Very good for Machine Learning Scenarios
- Cons:
  - Frail consistency
  - Lack of a query language
  - Questionable Durability
- Use Cases:
  - Social Networks
  - Conversational Systems
  - IoT
  - Big Data Analytics with those semi-structured data

## NoSQL Databases

## Cassandra:

- Type: key-value and column oriented hybrid.
- Description: It is free, very scalable and has a structure very similar to relational databases. There are rows and within those there are collections in which data is stored, where each column is a key with its own value and timestamp and the primary key is the partition index.
- Pros: Best Scalability of all NoSQLs, has a query language called CQL and very good partition management;
- Cons: Proportional operation latency to the nodes it is partitioned, due to its lack of row level consistency, parallel alterations on attributes from the same registry may result in inconsistencies.

# Neo4J:

- Type: graph-oriented.
- Description: ACID compliant transactional database with native graph storage and processing. Every data is a graph component, either an edge, a node or an attribute of a node.
- Pros:
  - Has a structure and logic very similar to the relational databases as what is beign modeled and stored here are essencially relationships, the main difference for the RDBMS is that here the data structure is a graph;
  - Optimized well-known algorithms to “walk” through the database
  - Interactive UI.
- Cons: Although it uses a well-known data structure, most of the developers are not familiarized with graphs, and prefer instead a table or other data structures.

## MongoDB:

- Type: document-oriented.
- Description: The most used NoSQL database in the development community. MongoDB as all document oriented databases are based upon IBM Lotus Notes and bascially store data as inidividual documents with key-value pairs inisde them. Uses JSON as document format.
- Pros:
  - High Scalability
  - Big Data Friendly
  - Great Community to help out
  - Perfect Documentation and Modules/Connectors/Libraries
  - Has it is own framework for data mapping, reduce and processing
  - Best Aggregation Framework that I ever worked with
- Cons:
  - Have to implement some few functionalities to manage data.
  - Developers are in charge of managing its system, data, availability, etc

## Cloudant:

- Type: document-oriented .
- Description: IBM NoSQL database based upon couchDB. Operates using the same structures as its father and got a few extra analytics functionalities like warehousing, easy connection to IBM DB2, etc.
- Pros:
  - High Scalability
  - Has a vast set of APIs
  - Very easy to work with
  - Good to beginners in NoSQL area
  - Has good integration with IBM's data analytics tools and systems
  - GeoJSON query optimization
- Cons:
  - Documentation is hard to find and consume
  - Plan Limitations

## Integrating Relational DBs and Non-Relational Ones

## Relational and Non-Relational Integration:

- Topics:
  - Schema
  - Data Migration
  - Warehousing

## Schemas:

- Relational
  - Defined prior to database usage
  - DDL
  - Can be altered after creation, but may implie in data structure changes

- Non-Relational
  - Can be omitted in database creation
  - Normally defined at application level
  - Some NoSQL management Systems have functions of schema discovery

# Data Migration:

- Althoug the databases are completely different the data migration is not a very complicated process.
- It goes as follows, and other operational optimization is done through the tools offered by each kind of management system.

- Relational > Non-Relational

1. Denormalize data
2. Retrieve the denormalized data
3. Convert it into an array of non relational data
4. Insert data from 3 into a specific collection of the non relational database

- Non-Relational > Relational

1. Flatten the data structure
2. Remove configuration atributes
3. Retrieve all docs from the Non-Relational
4. Convert the set of non-relational data (normally an array) into a csv or any tabular structure
5. Load data from 4 into the RDB

## Warehousing:

– A Data Warehouse is a structure that aggregates different kinds of data from different sources in order to perform aggregations, analysis, predictions, etc.
– The warehousing process is not much different from relational to non-relational, the main difference is that the non-realtional data will need to be migrated to the relational format before entering the data marts in the warehouses we know nowadays.

## Cloudant Backend Example in Node.js

## Example:

- Creating a cloudant database:
  - Enter IBM Cloud -> [https://console.bluemix.net](https://console.bluemix.net)
  - Create a new resource.
  - Type Cloudant in the catalog’s search bar.
  - Click on Cloudant Icon and choose the most adequate plan.
  - Enter the resource’s page and look for the credentials section on the side bar. There create the credentials of the service to be used in backend instances.
  - Return to the manage section, and launch the browser UI.
  - There, click on the databases section.
  - In the high-left corner click on create new collection
  - Ready, now you started your first databe and collection.

## Example:

- Basic CRUD operations backend in node.js, to show how to use and manage cloudant database.
- Link -> https://github.com/FernandoDurier/cloudant-basics-handson

## References

# References:

- CATTELL, Rick. Scalable SQL and NoSQL data stores. **Acm Sigmod Record**, v. 39, n. 4, p. 12-27, 2011.
- PARKER, Zachary; POE, Scott; VRBSKY, Susan V. Comparing nosql mongodb to an sql db. In: **Proceedings of the 51st ACM Southeast Conference**. ACM, 2013. p. 5.
- ÖZCAN, Fatma et al. Are we experiencing a big data bubble?. In: **Proceedings of the 2014 ACM SIGMOD international conference on Management of data**. ACM, 2014. p. 1407-1408.
- CHEBOTKO, Artem; KASHLEV, Andrey; LU, Shiyong. A big data modeling methodology for Apache Cassandra. In: **Big Data (BigData Congress), 2015 IEEE International Congress on**. IEEE, 2015. p. 238-245.
- WANG, Guoxi; TANG, Jianfeng. The nosql principles and basic application of cassandra model. In: **Computer Science & Service System (CSSS), 2012 International Conference on**. IEEE, 2012. p. 1332-1335.
- CHODOROW, Kristina. **MongoDB: The Definitive Guide: Powerful and Scalable Data Storage**. " O'Reilly Media, Inc.", 2013.

- BIENKO, Christopher D. et al. **IBM Cloudant: Database as a Service Advanced Topics**. IBM Redbooks, 2015.