Data Bases

Introduction to NoSQL

Sergio Peignier sergio.peignier@insa-lyon.fr
Erwan Cruché erwan.cruche@insa-lyon.fr

INSA Lyon
Biosciences department

Definition

NoSQL : Not Only SQL

Umbrella term for new-generation non-relational DBMS.

Definition

Distributed databases

Database in which the data is **replicated and stored across a network of physical locations**.

CAP / Brewer's Theorem

Distributed DBMS only provide 2 out of 3 properties:

- **Consistency**: Every node request returns either the most recent write or an error. (Different from the Consistency in the ACID paradigm)
- **Availability**: Every node request receives a non-error response (even if other node fails), but do not always get most recent updates.
- **Partition Tolerance**: The system can operate in the presence of network partitions (i.e., the network loses its connectivity and some messages are dropped or delayed).

BASE properties

NoSQL DBMS follow the **BASE** paradigm (replacing **ACID**)

- **BAsic availability** :
  Highly distributed system $\Rightarrow$ high level of data replication $\Rightarrow$ Availability in presence of failures or high traffic.
- **Soft state**: Data state **can change**, without user intervention, due to **eventual consistency**. Data could expire unless it is maintained.
- **Eventual Consistency**: No guarantee of consistency at transaction levels. But data is replicated in many nodes $\Rightarrow$ possible convergence to a consistent state.

Benefits

- **No predefined data model schema**:
  - ⇒ **Flexible storage**:
    For **heterogeneous** and **unstructured data**
  - ⇒ **Simple storage mechanism**
    High performance (fast read and write)

- **Horizontal scaling**:
  When **DB grows**, more **processing/storage nodes** can be added to the DBMS **distributed system**.

- NoSQL DBMS are often **open-source**

Disadvantages

- **Non-standardized Query Language**

  Each DBMS has its own query language

- NoSQL DBMS **do not guarantee ACID** properties:

  Weak control over transactions' **Consistency**, **Isolation** and **Durability**

- **Integrity** is often **not ensured** per se.

  e.g. Domains restrictions, null values ...

  External functions needed to ensure integrity.

SQL or NoSQL?

Factors to be considered

- DB **storage volume**
- Desired level of **scalability**
- **Number of operations** per time unit
- **Type** of **most frequent** operations
- Level of **concurrence**
- Level of **integrity** and **consistency** required

Major NoSQL Paradigms

- **Key-value** databases
- **Document-oriented** databases
- **Graph-oriented** databases
- **Column-oriented** databases

Key-value database

DB store **Values indexed by keys**:

- Store **structured** and **unstructured** data
- **Values** stored as **arrays of bytes**
  $\rightarrow$ **Content** is **disregarded** by DB
- High **performance**, **scalability**, **simplicity** and **flexibility**
- **No complex queries supported** (only look for keys)
- Examples: Amazon DynamoDB, Cassandra, Voldemort

Document-oriented database

- **Similar** to **Key-value** storage
- **Format**: JSON (JavaScript Object Notation), XML ...
- **Difference**: DBMS can read and query the values
- Allows more **advanced data queries**
- Example: CouchDB, MongoDB

Graph-oriented databases

DB = Graph: with **nodes=entities** inter-connected by **edges=relationships**

- Nodes and edges can have attributes
- **Queries** can be applied to **nodes and edges**
- **Performance** and **scalability** is **variable**
- **Complex to implement** but **very flexible**
- Examples: Neo4J, Hyperbase-DB, Infogrid

Column-oriented database

Very **similar** to **traditional Relational systems**

- Relies on table-based storage
- Can support SQL-like queries

Difference: **Tables** stored by **columns** rather than by **rows**

- Efficient: many operations on a column
- Inefficient: operations on many columns of same row

ID	Name	Salary
001	Gogo	10
002	Didi	9

Row-oriented

001:Gogo,10
002:Didi,9

Column-oriented

Gogo:001,Didi:002
10:001,9:002

DBMS Choice

	Key-val.	Docs.	Graphs	Cols.	Relational
Performance	H	H	Var	H	Var
Scalability	H	H	Var	H	Var
Flexibility	H	H	H	M	L
Complexity	L	L	H	L	M
Functionality	L	M	H	M	H