Introduzione ai DBMS NoSQL

Il movimento **NoSQL** promuove l'adozione di DBMS **non basati sul modello relazionale**.
Il termine NoSQL viene usato per lo più nell'accezione **NoT Only SQL**.
Alcune proprietà dei sistemi NoSQL sono:

- Database **distribuiti**
- Tool **open-source**
- Non dispongono di **schema**
- Non supportano operazioni di **join**
- Non implementano le proprietà **ACID** delle transazioni
- Sono **scalabili** orizzontalmente
- Sono in grado di gestire **grandi moli di dati**
- Supportano le **repliche** dei dati

Si osservino ora le motivazioni principali legate alla diffusione dei databse *NoSQL*:

Gestione dei Big-Data
I **Big-Data** sono grandi moli di dati, eterogenei, destrutturati, difficili da gestire attraverso tecnologie tradizionali, come i *RDBMS*. Il termine *big-data* è oggi usato sia per denotare tipologie di dati, sia le **tecnologie** e i **tool** di gestione degli stessi.
Nel contesto dei big-data, osserviamo dei concetti fondamentali tramite le quattro **V**:

- **Volume**: grossa mole di dati. Alcuni esempi possono essere i dati di esperimenti scientifici, sensoristica, IoT
- **Velocità**: *stream* continuo di dati. Un esempio potrebbero essere i sistemi healt-care
- **Varietà**: dati eterogenei, multi sorgente. Un esempio potrebbero essere i social media
- **Valore**: possibilità di estrarre conoscenza dai dati. Alcuni esempi possono essere il **data mining**, ovvero delle tecniche di **apprendimento computerizzato** per analizzare ed estrarre **conoscenze** da collezioni di dati

Limitazioni del modello relazionale
Sono presenti tre limitazioni principali:

- Il modello relazionale presuppone implicitamente la presenza di **dati strutturati** in rappresentazione tabellare. Cosa accade se i dati non si presentano in tale forma?
- Alcune operazioni non possono essere implementate in SQL, come la memorizzazione di un grafo e il calcolo del percorso minimo tra due punti
- La scalabilità *orizzontale* dei DBMS relazioni. La **scalabilità** è la capacità di un sistema di migliorare le proprie prestazioni per un certo carico di lavoro, quando vengono aggiunte nuove risorse al sistema. Per quanto riguarda la scalabilità *verticale*, si parla di aggiungere più potenza di calcolo ad i nodi che gestiscono il databse. Invece, per scalabilità *orizzontale* si intende aggiungere più nodi al cluster

Teorema CAP
Il teorema di **Brewer** (**CAP Theorem**) afferma che **un sistema distribuito può soddisfare al massimo solo due delle tre proprietà elencate successivamente**:

- **Consistency**: tutti i nodi della rete vedono gli stessi dati
- **Availability**: il servizio è sempre disponibile
- **Partion Tolerance**: il servizio continua a funzionare correttamente anche in presenza di perdita di messaggi o di partizionamenti della rete

Si osservi ora i diversi casi, combinando le tre proprietà in coppie:

- **Consistency + Availability (CA)**: si parla di **No Partition Tolerance**, dove il sistema *non funziona correttamente* in caso di perdita di messaggi
- **Availability + Partion Tolerance (AP)**: si parla di **No Consistency**, dove si trovano repliche del dato non aggiornate
- **Consistency + Partion Tolerance (CP)**: si parla di **No Availability**, dove la query non produce risposta

Ritornando al concetto di DBMS NoSQL, esistono alcune proprietà base:

- **Basically Available**: i nodi del sistema distribuito possono essere soggetti a guasti, ma il servizio è sempre disponibile
- **Soft State**: la consistenza dei dati non è garantita in ogni istante
- **Eventually Consistent**: il sistema diventa consistente dopo un certo intervallo di tempo, se le attività di modifica dei dati cessano

Il termine NoSQL identifica **una varietà di DBMS non relazionali**, basati su **modelli logici differenti**:

Databse ***chiave-valore***
I dati di un databse sono come liste di coppie *chiave/valore*, come per array associativi o dizionari. La **chiave** è un valore univoco per operazioni di ricerca, mentre il **valore** è il valore associato alla chiave. Alcuni esempi possono essere **Project Voldemort** e **BerkeleyDB**.

Databse ***document-oriented***
Avviene una gestione di dati **eterogeneei** e **complessi**. Sono scalabili **orizzontalmente**, con supporto per partizionamento dei dati in sistemi distribuiti. I **documenti** sfruttano il concetto di coppie chiave/valore, tramite file *JSON*. Questa gestione dei databse fornisce funzionalità per aggregazione/analisi dei dati. Alcuni esempi possono essere **MongoDB** e **CouchDB**.

Databse ***column-oriented***
I dati vengono organizzati su colonne anziché delle righe. Vengono utilizzate le column family. Le **column family** sono contenitori di colonne, dove ogni column family è scritta

su un file diverso. Ogni riga dispone di una chiave primaria, detta **row key**. E' uno schema **flessibile**, con maggiore efficienza nello storage e maggiore possibilità di **comprensione dei dati**. Alcuni esempi possono essere **HBase** e **Cassandra**.

Databse ***graph-oriented***
I dati vengono strutturati sotto forma di grafi, dove ogni nodo ed arco ha diversi attributi. Alcuni esempi possono essere **Neo4J** e **Titan**.