**Web Developer**

**Unità Formativa (UF): Fondamenti di base di dati**

**Docente: Adelchi Valenti**

**Titolo argomento: Database NoSQL**

**Cosa sono i database NoSQL (database non relazionali)?**

Un database NoSQL (Not Only SQL, Non-SQL) è un database che gestisce i dati in una struttura non relazionale e non utilizza il linguaggio SQL per eseguire operazioni sui dati. Ciò significa che è senza schema (nessuno schema fisso) ed evita i join. Grazie alla sua natura scalabile, ad alte prestazioni e flessibile, la sua popolarità è in aumento nei Big Data e nelle applicazioni in tempo reale.

Esistono quattro tipologie di database NoSQL.

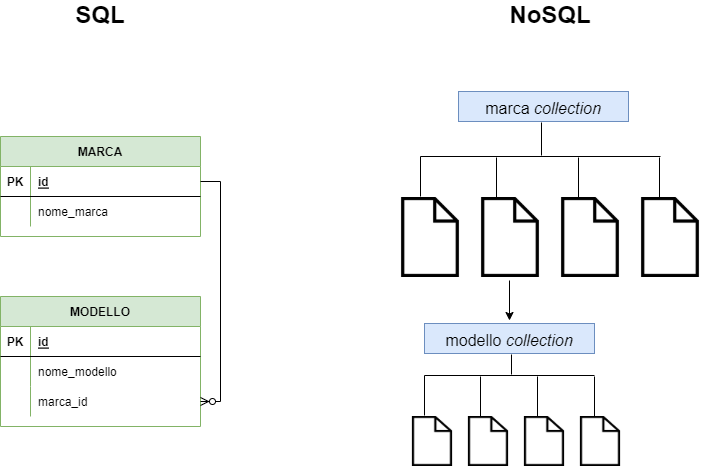
* Document-oriented
* Key-Value
* Wide-column
* Graph

#### ****Document-oriented**[[1]](#footnote-2) (MongoDB, ElasticSearch, CouchDB, CosmosDB)**

La rappresentazione dei dati è affidata a strutture simili ad oggetti, dette documenti, ognuno dei quali possiede un certo numero di proprietà che rappresentano le informazioni. Per i documenti non è obbligatorio avere una struttura rigida.

Ad esempio se i documenti devono rappresentare delle vetture, sicuramente ogni documento conterrà le informazioni relative a marca e modello ma potremmo definire in un documento la cilindrata ed in un altro le dimensioni. Questa è una delle principali caratteristiche dei documenti, che non necessitano della definizione di alcuna struttura fissa. Una cosa simile non si potrebbe fare in database relazionale.

Volendo fare un parallelo con un database relazionale, il documento è l’equivalente del record delle tabelle. I documenti possono essere messi in relazione tra loro con dei riferimenti.

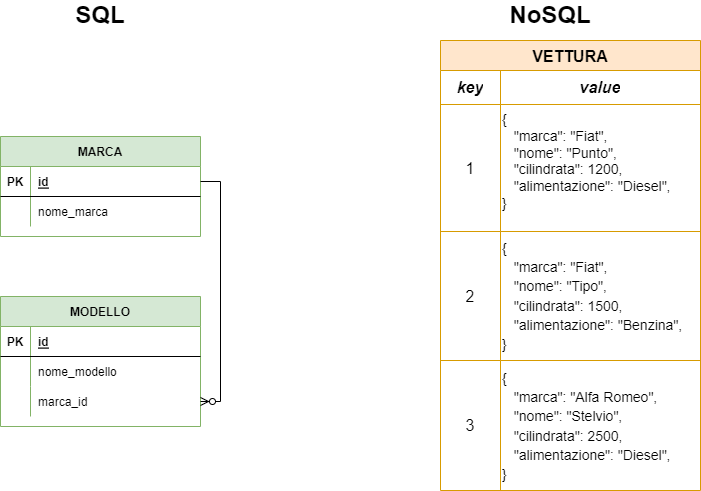


**Key-Value[[2]](#footnote-3)** (Redis, Aerospike, Memcached, Couchbase, Amazon DynamoDB, Oracle Berkeley DB)

Come in una tabella hash, tutto è archiviato in una coppia chiave-valore: ogni chiave univoca punta a un valore. Il valore può essere qualsiasi cosa, da numero, stringa a JSON e BLOB (Binary Large Objects) e quindi sono altamente scalabili e possono gestire elevati volumi di traffico.

I database key/store sono estremamente performanti nella ricerca di valori e sono spesso ottimizzati per conservare i dati in memoria e scriverli periodicamente su disco in modo da garantire, sia un certo livello di efficienza delle risposta, sia la persistenza dei dati.

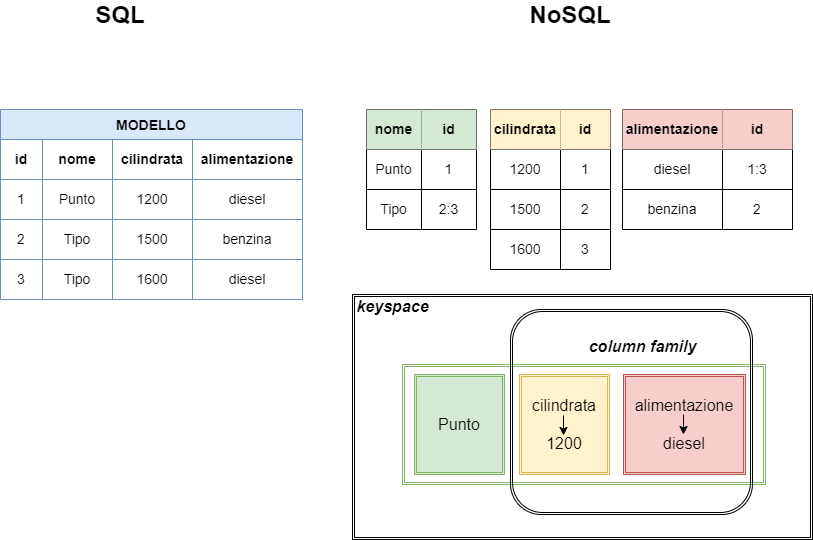
E’ un database ideale, per esempio,per la gestione delle sessioni o di carrello di un sito di e-commerce.



**Column-oriented[[3]](#footnote-4)** (Cassandra, Hbase, Bigtable, Hadoop,, Hypertable, Azure SQL Data Warehouse)

I database relazionali sono ottimizzati per memorizzare righe di dati mentre i database a colonne sono ottimizzati per recuperare rapidamente colonne di dati, in genere per applicazioni di analisi. Lo storage a colonne per tabelle di database è un fattore molto importante quando si tratta di prestazioni di query analitiche, perché è in grado di ridurre drasticamente i requisiti generali di I/O di un disco, nonché di diminuire la quantità di dati da caricare da disco.

Più specificamente, i database di colonne utilizzano il concetto di *keyspace* , che è una sorta di schema nei modelli relazionali. Questo spazio delle chiavi contiene tutte le famiglie di colonne, che poi contengono le righe, che poi contengono le colonne.

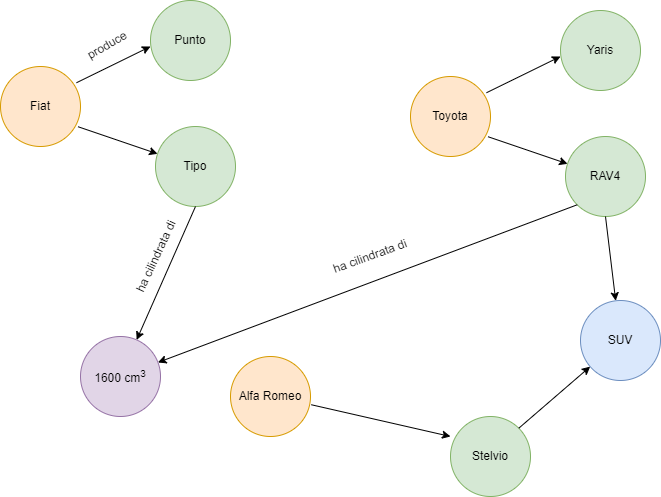


**Graph[[4]](#footnote-5)** (Neo4j, ArangoDB, OrientDB, GraphDB, Amazon Neptune, Fauna)

Funzionano come i grafi, strutture dati che rappresentano una rete di nodi collegati tra loro da archi.

Un nodo rappresenta un'entità, un arco rappresenta il modo in cui due nodi sono associati.

I database a grafo utilizzano nodi che contengono elenchi di record di relazione. Questi record rappresentano le relazioni con altri nodi, il che elimina l'operazione di ricerca-match (che richiede tempo) usata nei database relazionali.



**Differenza tra database SQL e NoSQL**

|  |  |  |
| --- | --- | --- |
|  | SQL | No SQL |
| Data Model | Struttura dati normalizzata | * Documenti * Key-Value * Column * Graph |
| Schema Design | Predefinito e rigido | Senza schema |
| Scalabilità | Scalabilità orizzontale (solo aggiungendo ulteriori server) | Scalabilità verticale (con aggiunta di RAM e aumentando la potenza dei processori) |
| Garanzia sulle transazioni | ACID  (**A**tomicity, **C**onsistency, **I**solation, **D**urability) | BASE  (**B**asically **A**vailable, **S**oft state, **E**ventually consistent) |
| Theorema CAP  **c**onsistency, **a**vailability, **p**artitioning | Forte Consistenza | Eventualmente consistente  Altamente disponibile  Tolleranza di partizione (il sistema continua a funzionare nonostante arbitrarie perdite di messaggi o malfunzionamenti) |
| Query language | SQL | Nessun linguaggio di query dichiarativo |

**SQL vs NoSQL: quale è più scalabile? (scalabilità)**

La scalabilità è un indicatore della capacità del sistema di aumentare o diminuire le prestazioni e i costi in risposta alla richiesta di elaborazione. Questo non deve essere confuso con la performance.

Il ridimensionamento orizzontale è più economico. L'aggiunta di più macchine è generalmente più semplice ed economica rispetto all'aggiornamento dell'hardware di un singolo computer (ridimensionamento verticale). ‍

Sia il ridimensionamento verticale che quello orizzontale sono possibili per NoSQL, tuttavia, il ridimensionamento orizzontale è difficile per SQL.

Limite tecnologico al ridimensionamento verticale: è limitato dalla tecnologia e dall'hardware disponibili.

**SQL vs NoSQL: quale è più veloce? (Prestazione)**

Le prestazioni sono un indicatore di quanto un sistema o un componente software soddisfi i propri requisiti di tempestività, misurati in base al tempo di risposta o alla velocità effettiva.

NoSQL è generalmente più veloce

NoSQL non è ACID - È stato progettato per essere più veloce rinunciando a una delle proprietà ACID

NoSQL archivia insieme i dati a cui si accede comunemente: hanno prestazioni migliori in lettura/scrittura su una singola entità di dati.

Gli schemi SQL convalidano le modifiche ai dati, ciò implica calcolo aggiuntivo necessario per garantire che le modifiche ai dati non violino gli schemi predefiniti. La natura senza schema di NoSQL salta questo passaggio e quindi esegue un maggior numero di operazioni di scrittura.

I database SQL vengono normalizzati: i dati vengono suddivisi e archiviati in tabelle. In quanto tale, è più veloce di NoSQL per query complesse come i join. I "join" in NoSQL vengono eseguiti recuperando più entità di dati e unendoli "manualmente". ‍

Gli indici sono più efficienti per SQL: gli indici funzionano duplicando i dati esistenti e archiviandoli in modo efficiente nella RAM. Funzionano meglio per i dati strutturati e quindi SQL ne trae maggiori benefici.

**SQL vs NoSQL: quale è più sicuro?**

Quando si tratta di confrontare la sicurezza dei database, utilizziamo 3 fattori:

Riservatezza - Solo gli utenti autorizzati possono accedere

Integrità - Precisione e coerenza dei dati

Disponibilità : i dati sono disponibili quando necessario

SQL è più sicuro.

SQL garantisce le transazioni ACID. I database NoSQL non lo fanno e quindi non garantiscono integrità dei dati. Ciò significa che le condizioni di concorrenza sono un problema e le transazioni simultanee possono influenzarsi a vicenda.

NoSQL ha funzionalità di sicurezza integrate minime: la sicurezza va integrata nelle logiche dell’applicazione. ‍

**Pro e i contro di SQL e NoSQL**

**Pro di SQL**

Archiviazione dei dati ridotta: la normalizzazione ha rimosso la ridondanza e la duplicazione dei dati.

Conforme ad ACID: Solida integrità, sicurezza e coerenza dei dati

Normalizzazione: i motori di database ottimizzano le query per adattarle alle rappresentazioni su disco.

Ottimo per query complesse: SQL è efficiente nell'elaborazione delle query e nell'unione di dati tra tabelle

Schema predefinito: le convalide vengono eseguite sul livello del database e non sul livello dell'applicazione.

Viene usato SQL come linguaggio standardizzato su diversi RDBMS.

**Contro di SQL**

Modello di dati rigido: richiede una progettazione iniziale e le modifiche in corso d’opera sono più difficili (richiede migrazioni dei dati e possibilmente tempi di inattività).

**Pro di NoSQL**

Schemaless: è facile apportare modifiche ai database al variare dei requisiti.

Modello dati flessibile: ogni entità può avere la sua struttura univoca.

Ridimensionamento orizzontale - Relativamente più economico e semplice, soprattutto per i Big Data

Altamente disponibile: Progettato per supportare il ridimensionamento continuo senza un singolo punto di errore significativo.

Prestazioni elevate: i dati a cui si accede insieme in genere vengono archiviati insieme.

**Contro di NoSQL**

Meno sicuro: difficile verificare l'integrità e la coerenza dei dati (solo eventuale coerenza raggiunta)

Schemaless: manca la standardizzazione dei tipi di dati.

Nessuna normalizzazione: i motori di database non sono così bravi nell'ottimizzazione delle query

Maggiore duplicazione dei dati a causa della mancanza di normalizzazione. Ma uno svantaggio relativamente piccolo in quanto lo spazio di archiviazione è relativamente economico.

Diversi tipi di database NoSQL: ognuno ha i suoi casi d'uso.

1. SEGA utilizza MongoDB per gestire 11 milioni di account di gioco. Le app iOS e Android di The Weather Channel forniscono avvisi meteo a 40 milioni di utenti in tempo reale. [↑](#footnote-ref-2)
2. *X* (ex Twitter) utilizza Redis per fornire la cronologia di tweets; Pinterest invece lo usa per archiviare elenchi di utenti, follower, unfollower e bacheche. [↑](#footnote-ref-3)
3. Spotify utilizza Cassandra per memorizzare gli attributi del profilo utente e i metadati su artisti, brani, ecc. per una migliore personalizzazione. Facebook ha inizialmente utilizzato HBase per il sistema di messaggistica interno. [↑](#footnote-ref-4)
4. Un esempio più familiare è un Social Network. Quando due utenti sono “amici” (Facebook) o “colleghi” (Linkedin), possono essere rappresentati da oggetti messi in relazione tra loro; pertanto si potrebbero proporre nuove potenziali conoscenze ad un utente navigando il grafo andando a recuperare “amici di amici” o “lavoratori” con cui si hanno in comune skills. [↑](#footnote-ref-5)