Database

Samuele Vignotto

Indice

Introduzione

I database sono componenti fondamentali nell'architettura dei sistemi informatici, utilizzati per la gestione e la conservazione dei dati. Esistono vari tipi di database, ciascuno con caratteristiche e applicazioni specifiche. Inoltre, l'emergere del process mining ha introdotto nuove modalità di analisi e utilizzo dei dati contenuti nei database. In questo documento, esploreremo i diversi tipi di database, come vengono gestiti i dati al loro interno, e la relazione tra database e process mining.

Tipi di Database

Database Relazionali

I database relazionali sono tra i più comuni e utilizzati, grazie alla loro capacità di gestire dati strutturati in maniera efficiente. Si basano sul modello relazionale introdotto da E.F. Codd negli anni '70, che rappresenta i dati sotto forma di tabelle (o relazioni). In una tabella, i dati sono organizzati in righe e colonne, dove ogni riga rappresenta un record unico e ogni colonna un attributo del record.
Il modello relazionale utilizza chiavi primarie e chiavi esterne per definire le relazioni tra le diverse tabelle. Una chiave primaria è un attributo o un insieme di attributi che identificano univocamente ogni record in una tabella. Una chiave esterna, invece, è un attributo in una tabella che crea un collegamento a una chiave primaria in un'altra tabella, stabilendo una relazione tra le tabelle.
Le operazioni sui database relazionali vengono effettuate utilizzando il linguaggio SQL (Structured Query Language). SQL permette diverse operazioni, tra cui:

- **INSERT**: per aggiungere nuovi record a una tabella.
- **UPDATE**: per modificare i dati esistenti.
- **DELETE**: per rimuovere record.
- **SELECT**: per recuperare dati dalle tabelle.

Oltre a queste operazioni di base, SQL supporta anche funzioni avanzate come:

- **JOIN**: per combinare righe da due o più tabelle basate su una condizione correlata. Esistono vari tipi di join, tra cui INNER JOIN, LEFT JOIN, RIGHT JOIN e FULL JOIN.
- **TRANSAZIONI**: per garantire che un insieme di operazioni SQL venga eseguito come una singola unità atomica, mantenendo la coerenza dei dati. Le transazioni seguono le proprietà ACID (Atomicità, Consistenza, Isolamento, Durabilità).
- **INDICI**: per migliorare la velocità delle operazioni di ricerca e di recupero dei dati. Gli indici sono strutture di dati aggiuntive che memorizzano una piccola parte dei dati di una tabella in modo da consentire accessi più rapidi.
- **VISTE**: per creare una presentazione virtuale dei dati da una o più tabelle. Le viste non memorizzano dati fisicamente, ma generano dinamicamente i risultati delle query quando vengono richieste.

I database relazionali garantiscono l'integrità referenziale, assicurando che le relazioni tra tabelle rimangano coerenti. Questo viene realizzato attraverso vincoli, come i vincoli di chiave primaria, di chiave esterna, di unicità, e di controllo, che possono essere definiti a livello di tabella per mantenere la qualità e l'integrità dei dati.
Alcuni dei più popolari sistemi di gestione di database relazionali includono Oracle Database, MySQL, Microsoft SQL Server, PostgreSQL e SQLite. Ognuno di questi sistemi implementa il modello relazionale, ma può offrire funzionalità e ottimizzazioni specifiche.

Database NoSQL

I database NoSQL sono progettati per gestire grandi volumi di dati non strutturati o semi-strutturati, offrendo scalabilità e flessibilità che i tradizionali database relazionali potrebbero non fornire. Essi nascono come risposta alle crescenti esigenze di gestione dei dati generate da applicazioni moderne come i social media, l'IoT, l'analisi in tempo reale, e altre applicazioni web su larga scala.
A differenza dei database relazionali, i database NoSQL non si basano su uno schema rigido di tabelle e relazioni. Al contrario, adottano diversi modelli di dati per ottimizzare la memorizzazione e il recupero delle informazioni. Esistono vari sottotipi di database NoSQL, tra cui:

Database a documenti

I database a documenti memorizzano i dati sotto forma di documenti, tipicamente in formati JSON (JavaScript Object Notation), BSON (Binary JSON) o XML (eXtensible Markup Language). Ogni documento è un record autonomo che può contenere dati complessi e strutturati in modo gerarchico, con array e oggetti nidificati. Questo modello è particolarmente utile per applicazioni che gestiscono dati eterogenei e in continua evoluzione. Esempi di database a documenti includono MongoDB, CouchDB e Amazon DocumentDB.

Database a colonne

I database a colonne, noti anche come store di colonne, immagazzinano i dati in colonne anziché in righe, come nei database relazionali. Questo approccio è ottimizzato per le query che coinvolgono grandi dataset, poiché le colonne possono essere lette e compresse in modo efficiente. Ogni colonna è memorizzata separatamente, il che consente di leggere solo le colonne necessarie per una particolare query, migliorando le prestazioni delle letture massicce. Esempi di database a colonne includono Apache Cassandra, HBase e Google Bigtable.

Database a grafo

I database a grafo sono progettati per gestire dati con relazioni complesse e interconnesse, come reti sociali, mappature di percorsi o gerarchie organizzative. I dati sono rappresentati come nodi, che contengono le entità, e archi, che rappresentano le relazioni tra le entità. Questo modello permette query rapide e intuitive su strutture di dati complesse, che sarebbero difficili da gestire con modelli relazionali tradizionali. Esempi di database a grafo includono Neo4j, OrientDB e Amazon Neptune.

Database a Chiave-Valore

I database a chiave-valore sono tra i più semplici e memorizzano i dati come coppie chiave-valore. Ogni chiave è unica e viene utilizzata per recuperare il valore associato, che può essere una stringa, un numero, un blob binario o un documento. Questo modello è altamente performante per scenari che richiedono velocità di accesso elevate e operazioni di lettura/scrittura rapide. Esempi di database a chiave-valore includono Redis, Riak e Amazon DynamoDB.

Gestione dei dati nei database NoSQL

La gestione dei dati nei database NoSQL varia significativamente a seconda del tipo di database utilizzato. Nei database a documenti, i dati possono essere facilmente modificati aggiungendo o rimuovendo campi nei documenti, senza dover alterare uno schema predefinito. Nei database a colonne, le tabelle possono scalare orizzontalmente con l'aggiunta di nuove colonne senza influenzare le query esistenti. I database a grafo consentono aggiornamenti dinamici delle relazioni tra i nodi senza compromettere le prestazioni delle query.

La flessibilità dei database NoSQL li rende ideali per applicazioni che richiedono scalabilità orizzontale, gestione di dati eterogenei e capacità di adattarsi rapidamente ai cambiamenti dei requisiti dei dati. Tuttavia, questa flessibilità può comportare una maggiore complessità nella gestione delle transazioni e nella garanzia della consistenza dei dati rispetto ai tradizionali database relazionali.

Database NewSQL

I database NewSQL rappresentano una nuova generazione di sistemi di gestione dei database che cercano di combinare i benefici dei tradizionali database relazionali con la scalabilità e le prestazioni dei database NoSQL. Questi sistemi sono progettati per supportare transazioni ACID (Atomicità, Consistenza, Isolamento, Durabilità) e per scalare orizzontalmente su più nodi, mantenendo al contempo la semplicità e la potenza del linguaggio SQL.

Caratteristiche principali dei database NewSQL

I database NewSQL offrono diverse caratteristiche che li distinguono sia dai database relazionali tradizionali sia dai database NoSQL:

- **Scalabilità Orizzontale**: A differenza dei database relazionali che generalmente scalano verticalmente (aggiungendo risorse a un singolo server), i database NewSQL sono progettati per scalare orizzontalmente, ovvero aggiungendo più server per distribuire il carico di lavoro. Questo approccio consente di gestire grandi volumi di dati e alte velocità di transazioni.
- **Transazioni ACID**: I database NewSQL garantiscono transazioni ACID, offrendo la stessa affidabilità e consistenza dei database relazionali. Questo è un vantaggio significativo rispetto a molti database NoSQL che spesso sacrificano queste proprietà per ottenere maggiore scalabilità.
- **SQL Avanzato**: I database NewSQL supportano SQL come linguaggio di query, permettendo agli sviluppatori di utilizzare competenze esistenti e strumenti familiari per interagire con il database. Questo include il supporto per join complessi, subquery, funzioni di aggregazione e altre caratteristiche avanzate di SQL.
- **Gestione Distribuita dei Dati**: Questi sistemi gestiscono i dati in modo distribuito, con algoritmi sofisticati per il bilanciamento del carico, la replica dei dati e la tolleranza ai guasti. Questo assicura che il sistema rimanga operativo anche in caso di guasti hardware o altre interruzioni.

Esempi di database NewSQL

Alcuni esempi di database NewSQL includono:

- **Google Spanner**: Un database relazionale distribuito sviluppato da Google, che offre consistenza globale, scalabilità orizzontale e supporto per transazioni ACID.
- **CockroachDB**: Un database SQL distribuito che è progettato per scalare orizzontalmente e per resistere ai guasti. Utilizza il protocollo di consenso Raft per garantire la consistenza dei dati.
- **VoltDB**: Un database in-memory relazionale che combina la scalabilità orizzontale con transazioni ACID. È particolarmente adatto per applicazioni che richiedono elevate velocità di transazione.
- **NuoDB**: Un database SQL distribuito che supporta transazioni ACID e offre scalabilità elastica, permettendo di aggiungere o rimuovere risorse dinamicamente.

Architettura dei database NewSQL

L'architettura dei database NewSQL varia tra i diversi implementazioni, ma generalmente include i seguenti componenti chiave:

- **Layer di Distribuzione**: Gestisce la distribuzione dei dati tra i nodi del cluster, bilanciando il carico di lavoro e garantendo la disponibilità e la tolleranza ai guasti.
- **Motore di Esecuzione SQL**: Responsabile dell'interpretazione e dell'esecuzione delle query SQL. Questo motore supporta funzioni avanzate di SQL, come join, subquery e transazioni.
- **Sistema di Replica e Consenso**: Utilizza algoritmi di consenso come Paxos o Raft per garantire che le copie dei dati sui diversi nodi rimangano consistenti. Questo componente è cruciale per mantenere le proprietà ACID in un ambiente distribuito.
- **Cache In-Memory**: Molti database NewSQL utilizzano cache in-memory per migliorare le prestazioni delle operazioni di lettura e scrittura. Questo permette di ridurre la latenza e aumentare la velocità delle transazioni.

Vantaggi e sfide

I database NewSQL offrono numerosi vantaggi, tra cui la capacità di gestire elevati carichi di lavoro transazionale con consistenza ACID, utilizzando un linguaggio di query familiare come SQL. Tuttavia, presentano anche delle sfide, come la complessità della gestione di un sistema distribuito e il costo potenziale delle risorse necessarie per mantenere le performance e la disponibilità desiderate.