Relazione MAADB - NoSQL lab project

Azzurra Oberto e Sofia Trucco

Introduzione

In questa relazione viene descritto lo sviluppo di un'applicazione web per l'interrogazione del database LDBC (Linked Data Benchmark Council) utilizzando due database NoSQL: un database a grafo (Neo4J) ed un database document-based (MongoDB).

Architettura del Sistema

Il sistema è stato architettato dividendo la responsabilità di gestione dei dati in tre server distinti che collaborano per rispondere alle interrogazioni dell'utente.
Il server principale è il *Main_Server*, gestisce l'interfaccia utente e coordina le comunicazioni con i due server secondari, *MongoDB_Server* e *Neo4J_Server*. I server sono implementati in Node.js utilizzando il framework Express. Il *Main_Server* è composto da una semplice cartella *public* che si occupa della parte di frontend (*index.html* e *index.js*), mentre il server.js si occupa della gestione delle richieste HTTP e del routing.

Il server MongoDB rappresenta la componente dedicata alla gestione dei dati statici e descrittivi del sistema. La sua architettura segue il pattern Model-View-Controller, con una separazione, quindi, dei diversi livelli di responsabilità. I controllers per *Person* e *Comment* gestiscono la logica specifica di ciascuna entità, mentre i corrispondenti file di routing definiscono gli endpoint REST. Il modulo *database.js* si occupa della connessione al database MongoDB denominato *ldbc*, con i relativi dettagli di configurazione e connessione.

Il server Neo4J, oltre ai controllers e routes, presenta un layer di servizi dedicato allo svolgimento delle query Cypher. Questo approccio a tre livelli permette di separare bene la logica del server. Il file *routes.js* definisce tutti gli endpoint necessari, mentre *services.js*, come scritto precedentemente, contiene l'implementazione delle query Cypher.

La decisione di adottare un'architettura distribuita a tre server è dovuta ad una migliore distribuzione delle logiche poiché ogni parte si focalizza sul proprio compito e ad una migliore manutenibilità del sistema.
L'isolamento dei componenti contribuisce anche alla robustezza del sistema, poiché un eventuale malfunzionamento di un server non compromette necessariamente il

funzionamento degli altri. Infine, ciascun server viene configurato e ottimizzato specificamente per il database che gestisce, sfruttando al meglio le peculiarità di MongoDB e Neo4j.

Dataset e Preprocessing

Il dataset utilizzato per il progetto è stato generato attraverso *ldbc_snbdatagen_spark*, una scelta motivata dalla maggiore semplicità di configurazione rispetto alla versione Hadoop. Il dataset LDBC simula un social network con diverse entità interconnesse che verranno illustrate in seguito.

Prima del caricamento dei dati nei database, i dati sono stati sottoposti ad un processo di pulizia utilizzando un Jupyter Notebook. Le operazioni principali di cleaning si sono concentrate sulla standardizzazione del formato datetime e della conversione dei tipi degli attributi adatti in modo da garantire la compatibilità, l'affidabilità e la consistenza con MongoDB e Neo4J.

Distribuzione dei Dati

La distribuzione dei dati tra MongoDB e Neo4J è stata progettata per massimizzare le performance del sistema sfruttando i punti di forza di ciascun database.

MongoDB gestisce le entità che presentano attributi descrittivi e relazioni statiche e/o che non vengono utilizzate:

- Entità:
    - Comment
    - Forum
    - Organisation
    - Person
    - Place
    - Post
    - Tag
    - TagClass

- Relazioni:
    - comment_hasTag_tag
    - forum_hasTag_tag
    - organisation_isLocatedIn_place
    - person_hasInterest_tag
    - person_isLocatedIn_city
    - place_isPartOf_place
    - post_hasTag_tag

    - tagClass_isSubclassOf_tagClass
    - tag_hasType_tagClass

Neo4J gestisce le relazioni dinamiche e sono state salvate le entità principali utili ai fini della nostra applicazione con un set ridotto di attributi in generale abbiamo salvato solo *id*, ma per *forum* consideriamo anche *title* e per *organisation* consideriamo anche *name* perché lo abbiamo ritenuto opportuno per le nostre query; per le relazioni abbiamo considerato il set di attributi originale.

- Entità:
    - Comment
    - Forum
    - Organisation
    - Person
    - Post

- Relazioni:
    - comment_hasCreator_person
    - comment_replyOf_comment
    - comment_replyOf_post
    - forum_containerOf_post
    - forum_hasMember_person
    - forum_hasModerator_person
    - person_knows_person
    - person_likes_comment
    - person_likes_post
    - post_hasCreator_person
    - person_studyAt_university
    - person_workAt_company

Ciascun database può essere utilizzato per le operazioni più adatte ad esso: MongoDB gestisce efficacemente le query di ricerca per attributi e le aggregazioni sui dati descrittivi, mentre Neo4J è ottimo per la navigazione di informazioni dinamiche in grafi complessi.

Implementazione delle Query

Le tre query analitiche sviluppate sono:

1. *“Trovare la correlazione tra il numero di amici di una persona ed il numero di post che ha creato.”* -> Neo4J
2. *“Trovare la media di reply ai commenti.”* -> Neo4J
3. *“Trovare l’età media del genere femminile che frequenta l’università e l’età media del genere femminile che lavora in azienda.”* -> Cross-database

Le tre query parametriche sviluppate sono:

1. *“Trovare i nomi dei 3 forum più seguiti dagli universitari dell'università X.”* -> Neo4J
2. *"Trovare tutti i giorni in cui una certa persona X ha messo like ad un commento in un certo range di date, utilizzando il browser X."* -> Cross-database
3. *“Trovare tutti i messaggi in cui è presente il TagClass X e ha location IP Y.”* -> MongoDB

Conclusioni

L’esperienza di sviluppo ha evidenziato l’importanza della fase di progettazione iniziale nella definizione della strategia di distribuzione dei dati poiché ha un impatto significativo sulle performance e sulla complessità dell’applicazione.
La scelta di utilizzare un’architettura a moduli offre manutenibilità migliore rispetto ad approcci monolitici.
Il progetto ha raggiunto con successo tutti gli obiettivi prefissati, dimostrando l’efficacia di un’architettura NoSQL ibrida nella gestione di dataset complessi di social network.