

**UNIVERSITÀ DEGLI STUDI DI MESSINA**

DIPARTIMENTO DI SCIENZE MATEMATICHE E INFORMATICHE, SCIENZE FISICHE E SCIENZE DELLA TERRA

#### Corso di Laurea Triennale in Informatica

Investigazione sulle frodi assicurative

Docenti:

Prof. Massimo Villari Prof. Antonio Celesti

Studente: Davide Capra, 515886

### ANNO ACCADEMICO 2023/2024

**Indice**

1. [Introduzione](#_bookmark0) 3
2. [DataBase](#_bookmark1) 4
   1. [MongoDB](#_bookmark2) 4
   2. [Neo4j](#_bookmark3) 4
3. [Data Model](#_bookmark4) 5
4. [Implementazione](#_bookmark5) 6
   1. [Importazione dei dati Neo4j](#_bookmark6) 6
   2. [Importazione dei dati MongoDB](#_bookmark7) 7
   3. [Query MongoDB](#_bookmark8) 8
5. [Test](#_bookmark9) 8
   1. [1°Query](#_bookmark10) 8
   2. [2°Query](#_bookmark11) 10
   3. [3°Query](#_bookmark12) 12
   4. [4°Query](#_bookmark13) 14
6. [Conclusioni](#_bookmark14) 16

# Introduzione

Le frodi assicurative negli Stati Uniti provocano danni stimati in circa 80 miliardi di dollari. Per affrontare questa sfida, le compagnie assicurative necessitano di identifi- care e possibilmente prevenire le frodi passate e future. Tuttavia, questa operazione è estremamente complessa a causa della natura intricata delle reti coinvolte, spesso difficili da interconnettere. Una parte significativa del problema deriva dall’am- pio utilizzo, da parte delle compagnie assicurative, di database relazionali. Questi database, sebbene capaci di stabilire relazioni tra i dati, impiegano una struttura tabulare, ormai considerata lenta. In alternativa, la nuova prospettiva suggerisce di trattare ogni dato relativo a clienti, avvocati, cause e altri fattori come un nodo, con le relazioni tra di essi rappresentate da archi, creando così una struttura a grafo. L’utilizzo di un database basato su questa struttura dati offre vantaggi significativi:

* + Maggiore velocità operativa
  + Flessibilità aumentata
  + Possibilità di eseguire query in tempo reale

Esaminando più approfonditamente il problema, emerge che le frodi spesso coinvol- gono una serie di cause legali, orchestrate da un gruppo che ha costruito una rete di frodi, ognuno con un ruolo specifico. Ad esempio, un cliente e un avvocato pos- sono avviare una causa, seguiti da un perito. Identificare frodi diventa più semplice quando si individuano correlazioni tra diverse cause (ad esempio, coinvolgimento dello stesso cliente, riferimento alla stessa abitazione, e coinvolgimento degli stessi avvocati e periti). I database NoSQL, in particolare Neo4j e MongoDB, sono conce- piti per individuare tutte le connessioni a partire da un sospetto, che può essere una persona o persino una proprietà. Inizialmente, questi database saranno popolati con un set di dati ridotto (circa il 25% del totale), che crescerà progressivamente fino a raggiungere un numero di nodi simile a quello di una piccola città. I dati, generati casualmente, sono stati salvati in file CSV e JSON per facilitare il caricamento nei database.

# DataBase

Come già precedentemente accennato, i due database utilizzati per la realizzazione di questo progetto sono MongoDB e Neo4j, per via del loro diverso approccio nella gestione di grandi quantità di dati, che si rileva efficiente in base a come questi dati sono organizzati tra loro.

## MongoDB

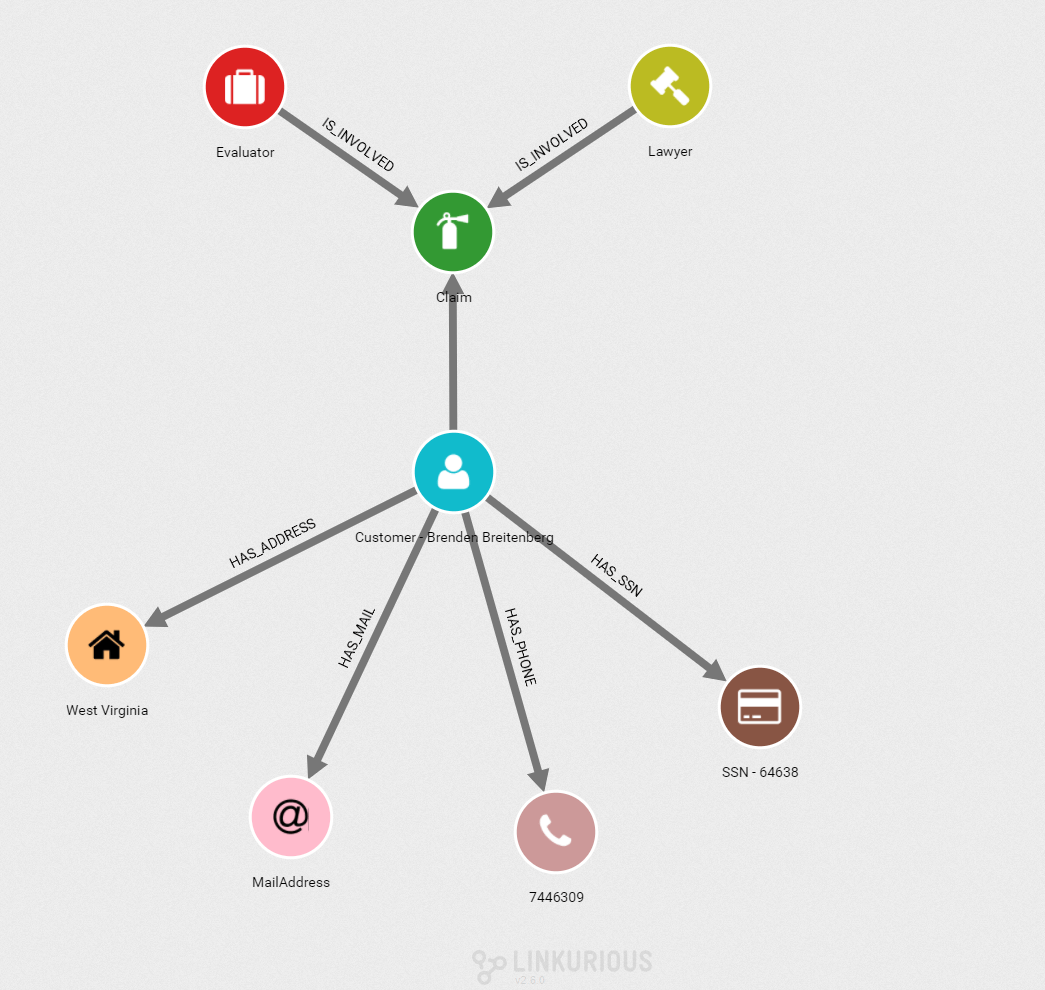
MongoDB rappresenta un tipo di database NoSQL orientato ai documenti, a dif- ferenza dei database SQL che sono focalizzati sul concetto di relazioni e tabelle. In MongoDB, l’unità organizzativa principale è la "Collection", che raggruppa dati sotto forma di "Documenti". Un documento è costituito da coppie chiave-valore (key-value) che formano uno schema dinamico. Ciò implica che i documenti all’in- terno di una collezione possono variare nei campi, mantenendo comunque una certa coerenza o correlazione tra di essi. Non è necessario seguire uno schema rigido o predefinito. MongoDB è particolarmente adatto per gestire dati con numerose ca- ratteristiche, senza necessariamente avere correlazioni tra di esse. Questi dati sono immagazzinati come file BSON (Binary JavaScript Object Notation), derivati dai file JSON (JavaScript Object Notation), rappresentando la versione codificata in bi- nario per un’elaborazione più veloce. Tuttavia, MongoDB presenta uno svantaggio significativo nella gestione delle relazioni tra diverse collezioni. Per affrontare questo problema, è stato introdotto l’Aggregation Framework, che consente di analizzare, processare e trasformare i documenti attraverso un meccanismo noto come "Ag- gregation Pipeline". L’utilizzo del metodo "aggregate" in questo contesto facilita l’esecuzione di query all’interno del database.

## Neo4j

Neo4j è un database NoSQL sviluppato in Java, basato sulla struttura dati del grafo, progettato per conservare e interrogare grandi volumi di dati. Questa carat- teristica rende Neo4j adatto a una vasta gamma di contesti, poiché molte situazioni possono essere rappresentate in modo efficace utilizzando nodi e archi. Neo4j con- sente di assegnare proprietà uniche a ciascun nodo, differenziandoli l’uno dall’altro, e di definire la direzione degli archi per creare relazioni. Le operazioni sui dati in questa struttura risultano generalmente più veloci rispetto ai tradizionali database relazionali, poiché la ricerca di nodi in relazione ad altri è un’operazione primitiva che non richiede passaggi aggiuntivi. L’app Desktop di Neo4j offre diverse soluzioni per interagire con il database. Neo4j Browser è la prima opzione, fornendo un’inter- faccia utente intuitiva per utilizzare Cypher, il linguaggio dichiarativo di query del sistema. Un’alternativa è Neo4j Bloom, più orientato alla visualizzazione grafica, consentendo di generare grafici in modo visuale selezionando nodi e relazioni da mostrare.

# Data Model

Sulla base del caso di studio considerato, si suppone di aver generato il Data Mo- del utilizzato per effettuare i test mettendo a disposizione l’insieme dei dati che riguardano i clienti, cause, avvocati e periti.

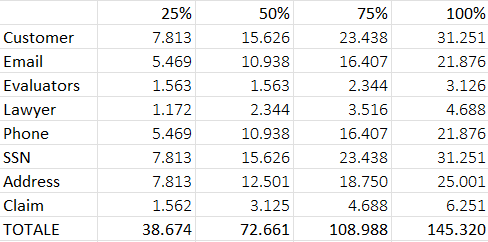


Per quanto riguarda Neo4j si è strutturato il grafo in maniera tale da collegare ogni Cliente con il suo SSN (univoco) e con ognuno dei suoi numeri di telefeono, mail e indirizzi. Successivamente ogni Cliente è messo in relazione alle varie Cause, alle quali sono collegati anche Avvocati e Periti.

Invece per MongoDB ogni nodo viene tradotto in documento e successivamente rag- gruppato in collezioni omogenee. Tutti gli attributi mantengono le stesse proprietà, tranne i Clienti, gli Avvocati ed i Periti ai quali vengono aggiunte altre proprietà per rendere coerente la randomizzazione effettuata in precedenza.

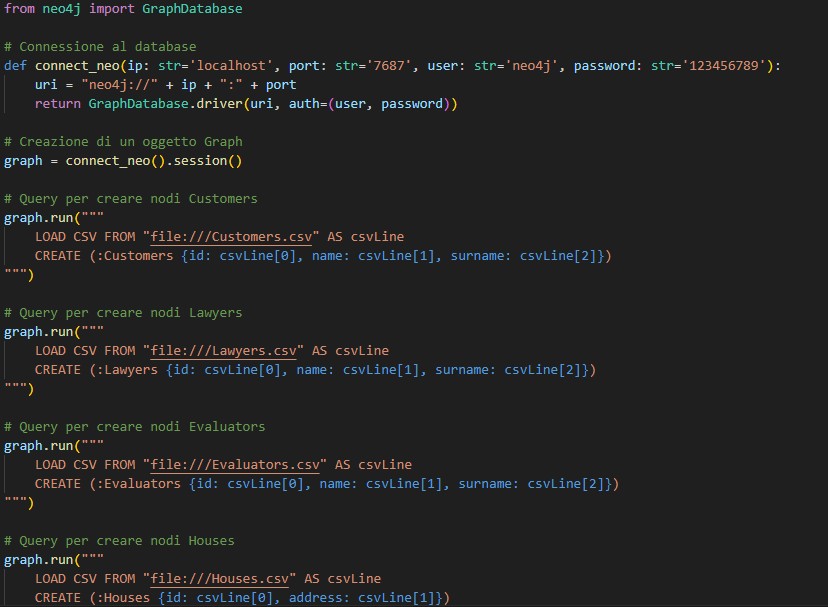
# Implementazione

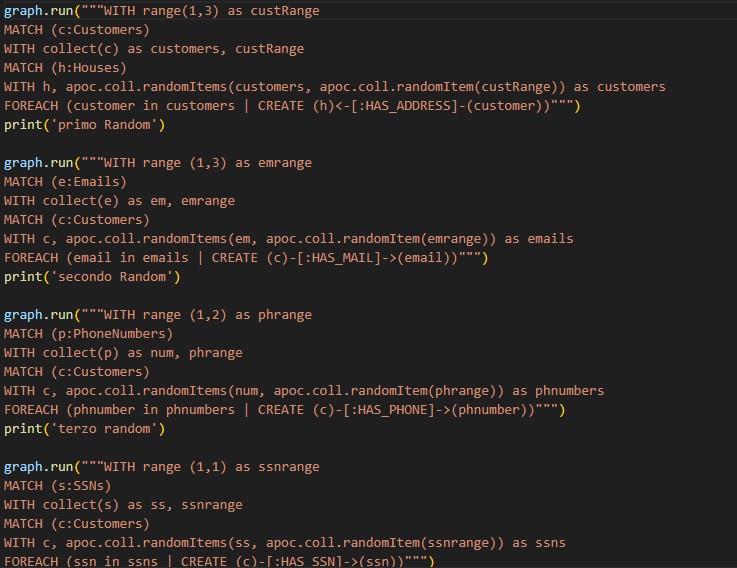
Il Data Model considerato è stato popolato attraverso la generazione casuale di dati tramite uno script in Python in cui viene utilizzato il modulo Faker. I valori ottenuti con le varie percentuali sono riassunti nella tabella seguente:



## Importazione dei dati Neo4j

I dati sono stati importati in maniera automatizzata tramite degli script Python. Per Neo4j, bisogna eseguire alcune operazioni preliminari: dall’app Neo4j Desktop occorre creare un progetto e al suo interno uno o più Database (nel nostro caso 4). Per ognuno di essi è necessario inserire i CSV contenti i dati nella cartella Import, scaricare il plugin APOC(necessario ad eseguire alcune funzioni di tipo matematico).È stato sufficiente un solo script in quanto la connessione viene stabi- lita sul localhost sulla porta 7687: basta dunque avviare il database desiderato e successivamente eseguire lo script.

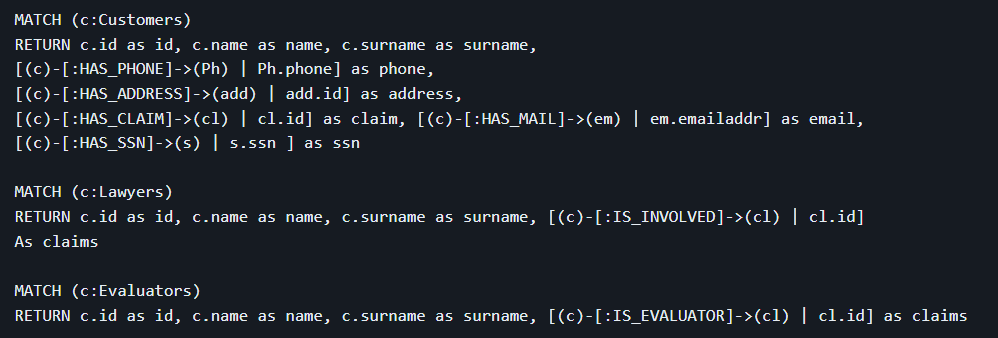




Le operazioni di randomizzazione sono necessarie a creare le relazioni fra i nodi: ad esempio si è deciso di assegnare obbligatoriamente un solo SSN ad ogni cliente, mentre lo stesso cliente può avere più numeri di telefono, email e indirizzi di casa. Per quanto riguarda le cause, si è imposto che avvocati e periti debbano necessa- riamente seguire una fino a cinque cause, mentre per il cliente è facoltativo averne avviata una, sempre con un massimo di cinque.

## Importazione dei dati MongoDB

Per mantenere la corenza delle randomizzazioni appena eseguite, è necessario ese- guire le seguenti query in Neo4j per avere le informazioni necessarie per ogni cliente, avvocato e perito.



Successivamente i risultati di queste query vengono scritte su un file Json che tramite uno script python vengono inseriti nel database di interesse. E’ sufficente modificare la variabile database per selezionare il Db di destinazione.



## Query MongoDB

# Test

Lo scenario immaginato è quello in cui un avvocato sia sospettato di essere coinvolto in delle frodi. Le query sono di complessità man mano crescente in quanto la informazioni da ottenere crescono man mano. In particolare saranno trovate le informazioni legate all’avvocato con ID 3. Nell’ordine troveremo:

* + Cause seugite dall’avvocato
  + Cause e periti coinvolti in esse
  + Cause e clienti che le hanno avviate
  + Clienti il cui indirizzo contiene il numero civico ’5’

Per ciascuna query, vengono registrati il tempo della prima esecuzione (non in ca- che) e il tempo medio delle successive 30 esecuzioni. Questa distinzione è necessaria poiché entrambi i database considerati implementano meccanismi di caching che migliorano le prestazioni nelle query ripetute. Successivamente, basandosi sui risul- tati ottenuti, sono stati calcolati la deviazione standard e l’intervallo di confidenza al 95%. La deviazione standard fornisce una misura di quanto ciascuna iterazione si discosti dalla media delle altre: un valore più alto indica una maggiore variabilità nei tempi di esecuzione delle query. L’intervallo di confidenza, d’altro canto, offre uno strumento per identificare i risultati significativamente al di fuori dalla media e limitarne l’impatto sui risultati complessivi.

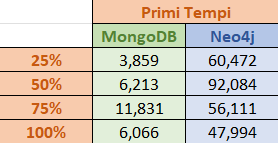
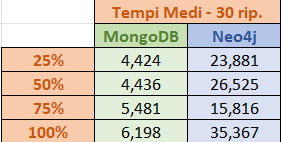
## 1°Query

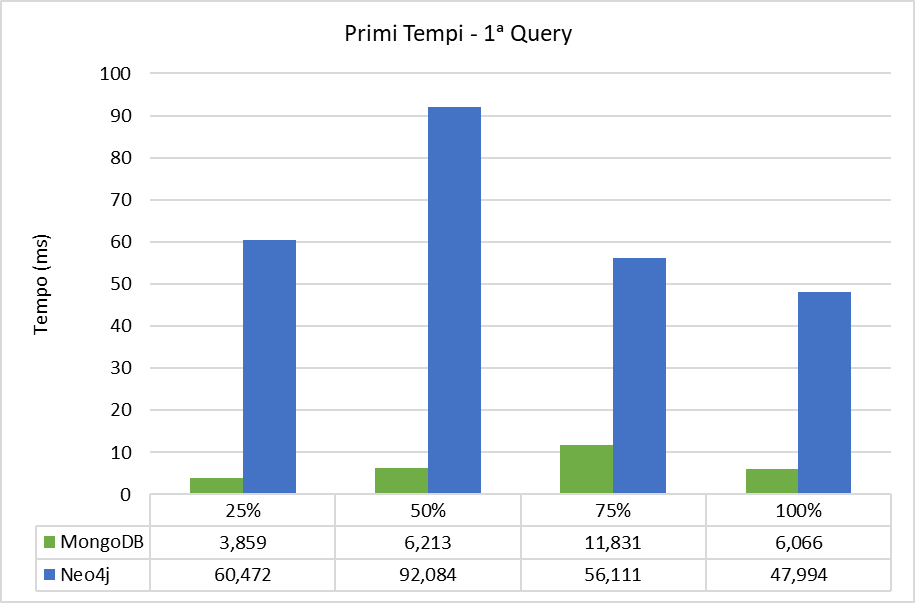
#### NEO4J:

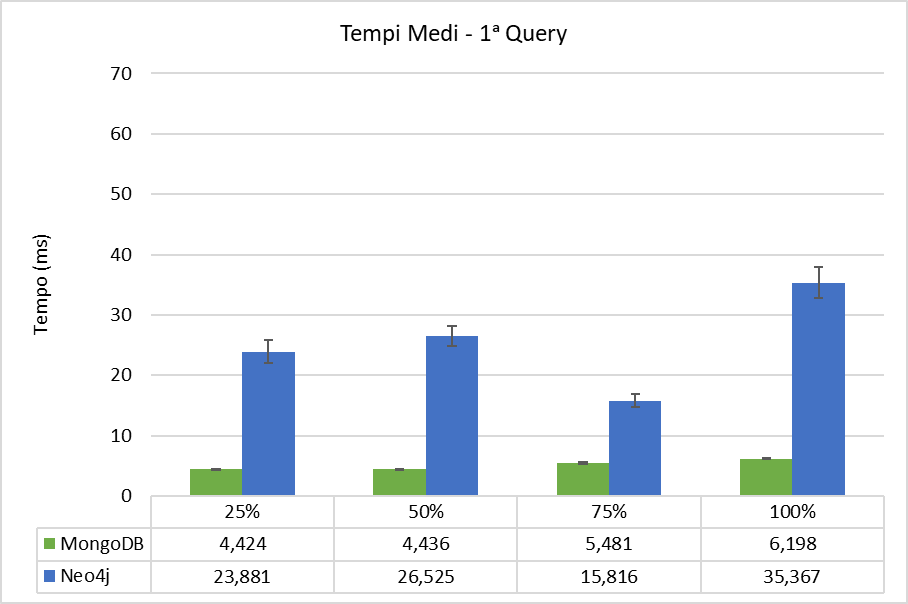


#### MONGODB:





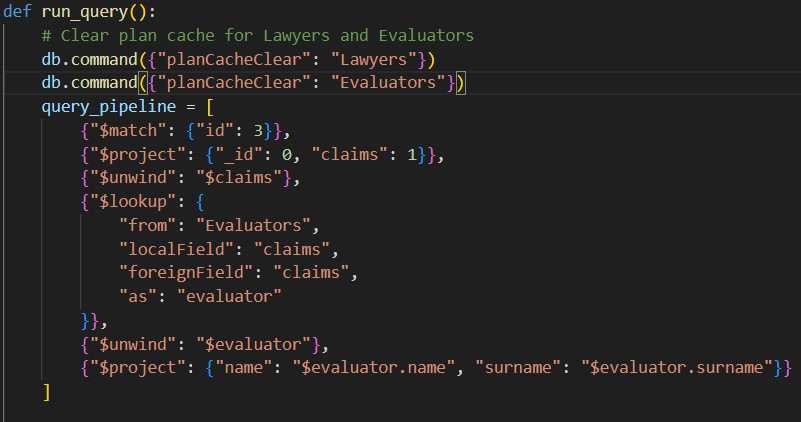


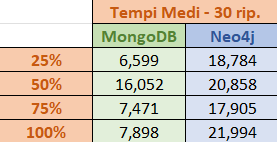
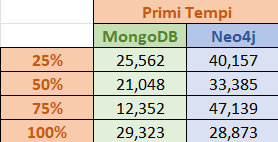
## 2°Query

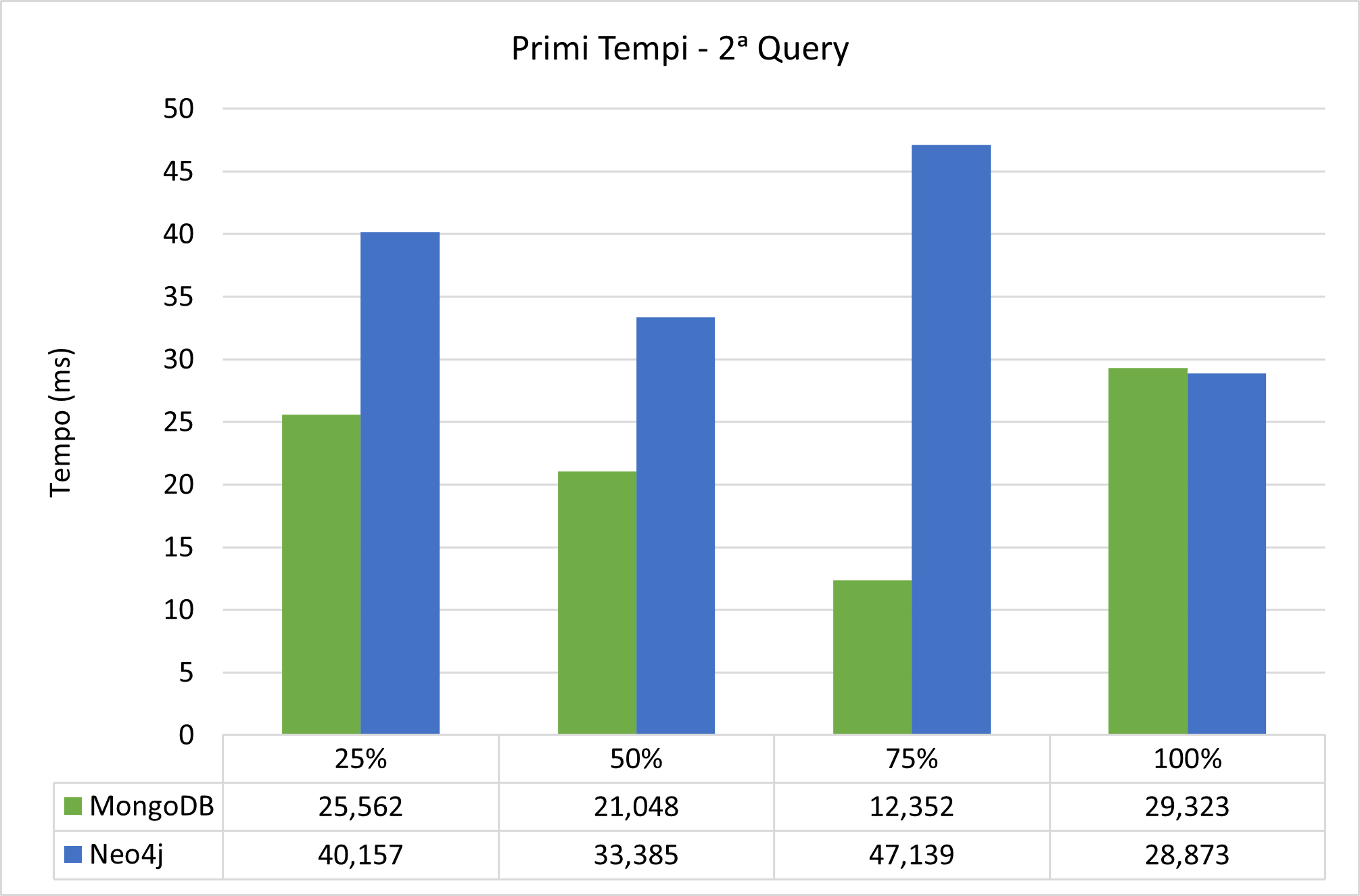
#### NEO4J:

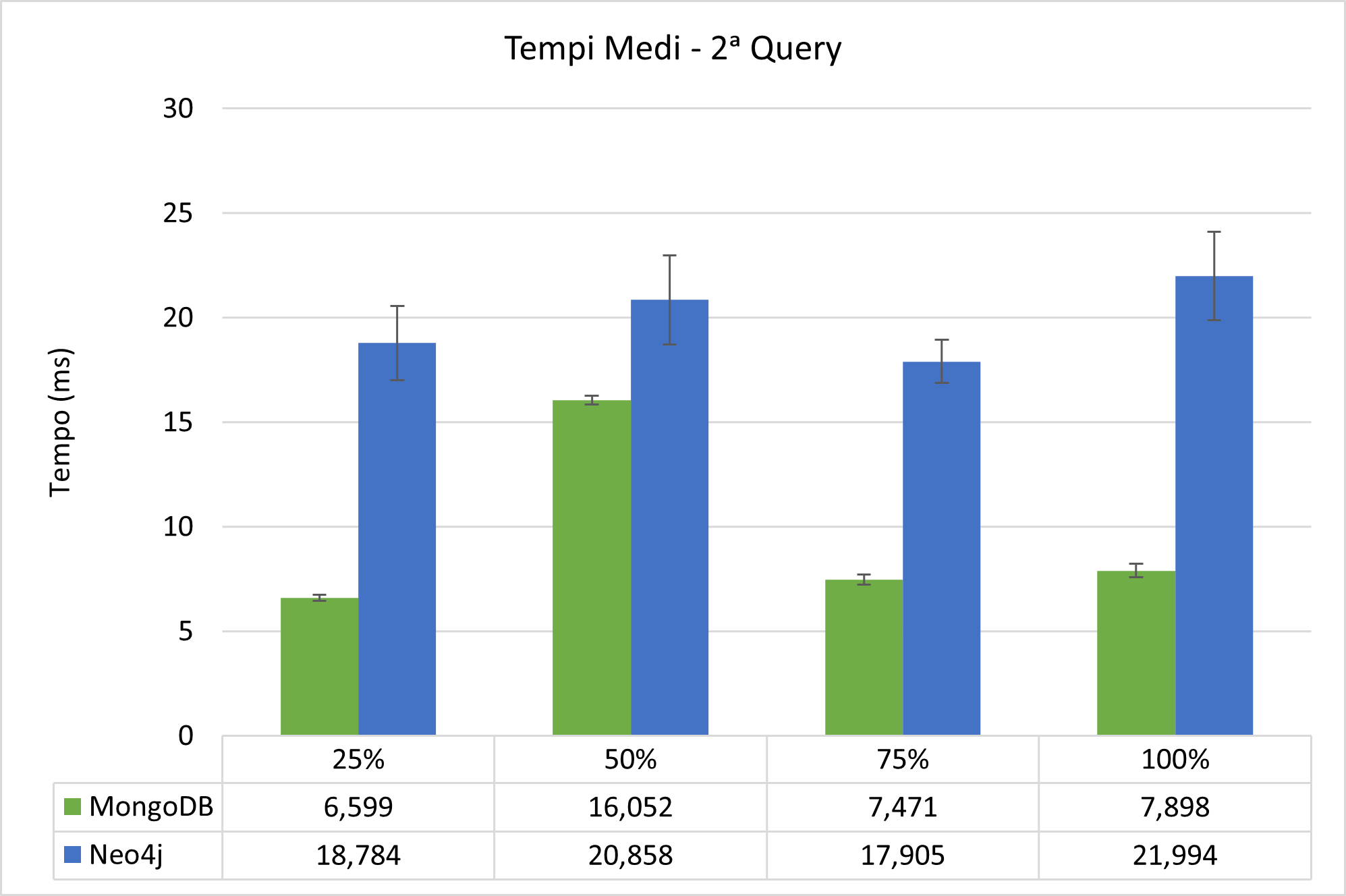


#### MONGODB:



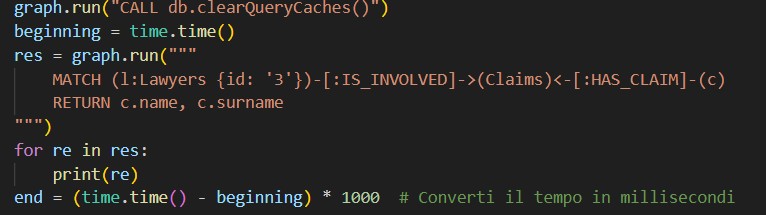




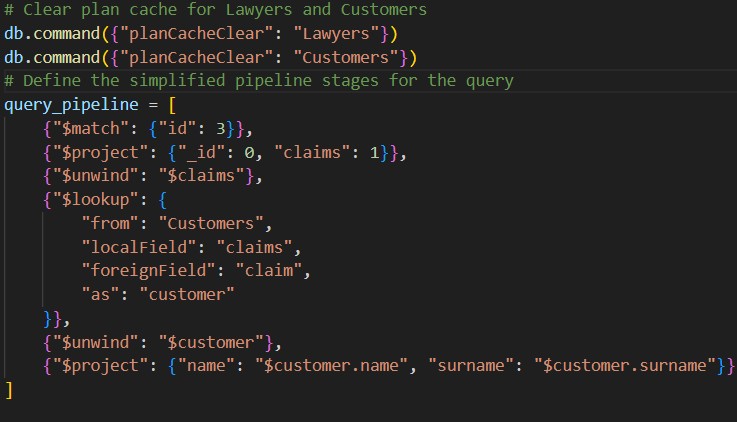


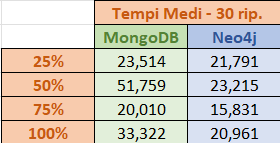
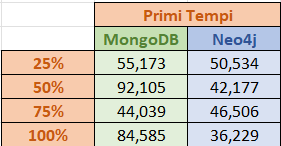
## 3°Query

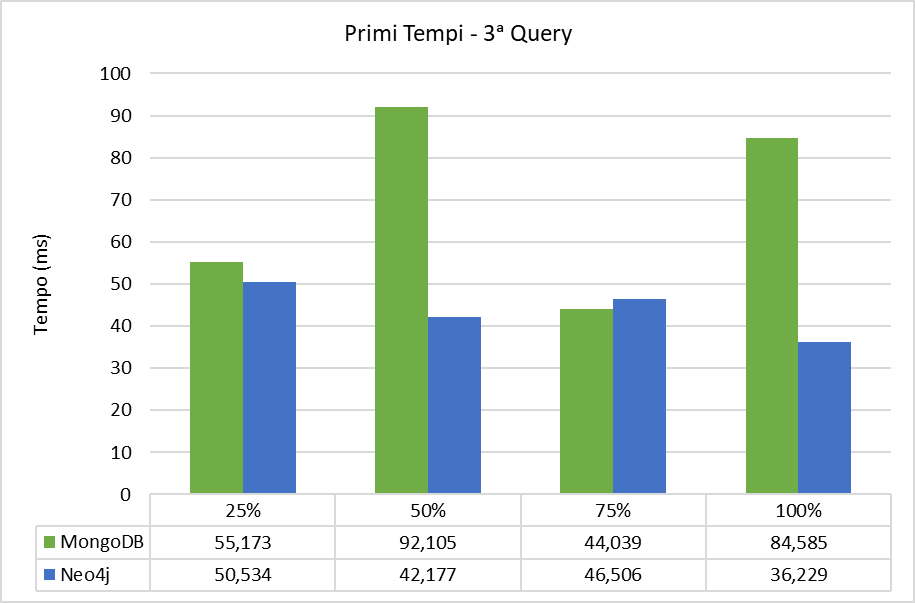
#### NEO4J:

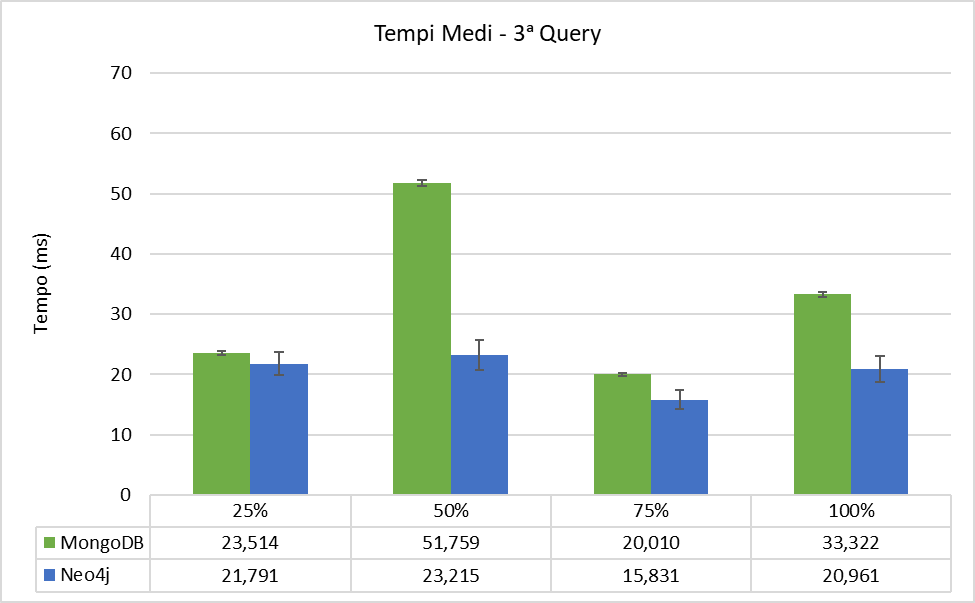


#### MONGODB:



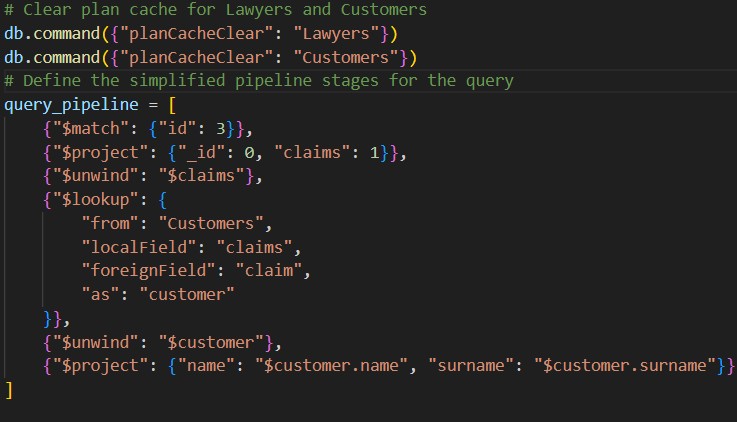






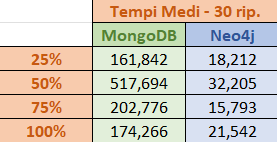
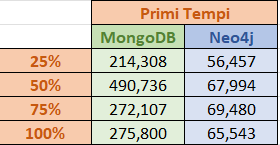
## 4°Query

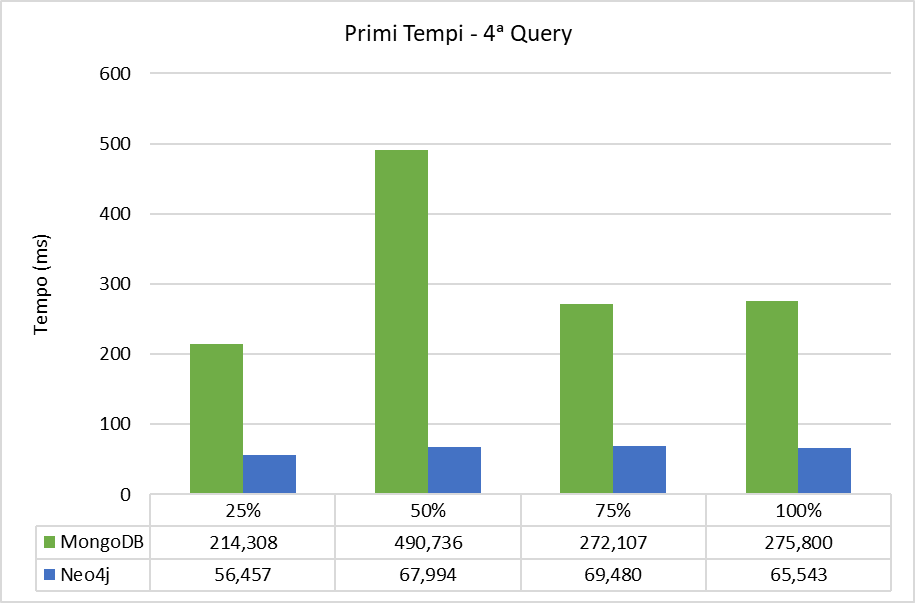
#### NEO4J:

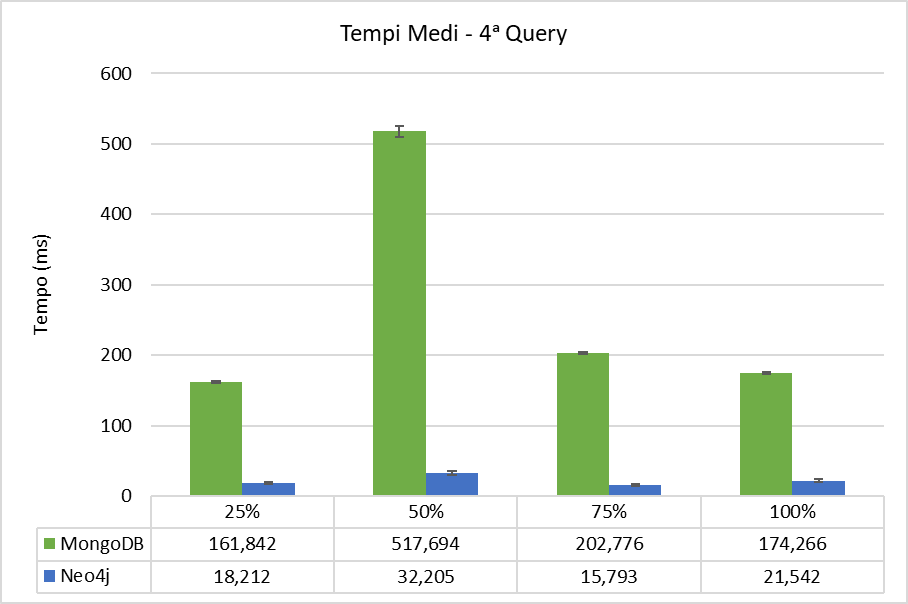


#### MONGODB:









# Conclusioni

Sulla base dei risultati ottenuti dai test effettuati a parità di dataset si può osser- vare come, considerando le caratteristiche dei database NoSQL, sia MongoDB che Neo4j riescano a gestire una grande quantità di dati in tempi decisamente brevi. Come si può evincere graficamente, vi è però una netta differenza tra i due data- base in base all’interrogazione considerata e al numero di entità coinvolte. Nella prima query si ha infatti una totale superiorità di MongoDB che si tiene al di sotto di 7 millisecondi per l’esecuzione del 100% del dataset mentre Neo4j rileva per la prima esecuzione un tempo di circa 47 millisecondi, diminuito per le 30 esecuzioni successive a circa 35 millisecondi. Ciò è dovuto alla totale assenza di relazioni tra le entità nell’interrogazione e al solo confronto di un capo, per cui un Graph Da- tabase come Neo4j non riesce a mostrare le sue ottimizzazioni per un caso che non ne necessita. La situazione cambia di netto dalla seconda query in poi, momento in cui iniziano ad essere inserite le prime relazioni tra le entità; si nota infatti che, all’aumentare della percentuale di dataset e della complessità delle interrogazioni, MongoDB riesca inizialmente a dare una risposta rapida, per poi incrementare no- tevolmente il tempo della prima esecuzione e a non riuscire a sfruttare i meccanismi di caching nelle 30 esecuzioni successive mantenendo i valori di esecuzione alti e quasi costanti, proprio a causa delle molteplici associazioni da eseguire. Di contro, come mostrato dall’andamento dei grafici, all’aumentare delle relazioni Neo4j riesce a sfruttare gli algoritmi di attraversamento dei grafi di cui dispone non andando quasi a superare i 200 millisecondi per l’esecuzione della quarta query, e soprattutto a utilizzare i meccanismi di caching che gli permettono di abbassare notevolmente i tempi nelle 30 esecuzioni successive. Sulla base di queste osservazioni si può quindi concludere che per il caso di studio considerato si ha , una maggiore efficienza nel DBMS orientato ai grafi Neo4j, utilizzato infatti generalmente per l’ottenimento di informazioni strettamente correlate tra loro, a discapito del DBMS orientato ai do- cumenti MongoDB, che invece dimostra le sue migliori prestazioni nelle operazioni di lettura e scrittura dei dati in assenza di relazioni; attività più consistenti infatti possono anche essere distribuite in più server replicati, e come si evince da questo studio le interrogazioni su questo tipo di dati sono più semplici e riescono a fornire dei risultati in maniera quasi immediata nonostante coinvolgano enormi quantità di elementi.