

**Corso di Laurea Triennale in INFORMATICA**

Progetto di Basi di Dati Mod.B

Realizzato da:

Gabriele Agosta

Vitaliy Lyaskovskiy

Indice

[***Problematica Affrontata*** 3](#_Toc138456974)

[***Soluzione DBMS considerata*** 4](#_Toc138456975)

[***Design*** 6](#_Toc138456976)

[***Implementazione*** 9](#_Toc138456977)

[*Inserimento dei dati* 9](#_Toc138456978)

[*Query* 11](#_Toc138456979)

[***Esperimenti*** 15](#_Toc138456980)

[***Conclusione con Analisi dei Risultati*** 19](#_Toc138456981)

# Problematica Affrontata

*“Rilevamento delle frodi nell'e-commerce”*

Le frodi nell'e-commerce sono un fenomeno in rapida crescita alimentato dall'espansione costante e inarrestabile dello shopping online. L'anonimato fornito dal web, l'accesso globale e la natura senza limiti del commercio elettronico creano un terreno fertile per le attività fraudolente. Da frodi con carta di credito a operazioni di phishing, gli attacchi fraudolenti possono assumere molteplici forme ed il miglioramento della loro sofisticazione ne rendono difficile la prevenzione.

In questo contesto, questo progetto si propone come uno strumento chiave per l'identificazione e la prevenzione delle frodi nell'e-commerce. L'obiettivo è sviluppare un sistema di database robusto e affidabile progettato per monitorare, analizzare e identificare comportamenti sospetti e attività irregolari nelle transazioni online.

Per raggiungere questo obiettivo il sistema sfrutta una vasta gamma di informazioni, queste includono i dati sugli utenti, le transazioni effettuate, i prodotti acquistati e le segnalazioni di frode già verificate. Ciascuna di queste categorie fornisce una parte dello schema complessivo grazie al quale l'analisi dei modelli di comportamento degli utenti può rivelare anomalie come frequenti transazioni di grandi dimensioni o acquisti ripetuti di un singolo prodotto, contribuendo quindi a rendere il commercio elettronico un ambiente più sicuro e affidabile per consumatori e venditori.

# Soluzione DBMS considerata

In questo progetto verranno utilizzati due database per la gestione e l'analisi dei dati, ognuno di essi presenta un insieme di caratteristiche che lo rendono particolarmente adatto a specifici tipi di operazioni e analisi.

Iniziamo con una breve presentazione delle principali caratteristiche di MongoDB, il primo dei due database.

MongoDB è un database di tipo NoSQL orientato ai documenti in cui i dati vengono archiviati utilizzando il formato BSON, una variante binaria del formato JSON.

I documenti in MongoDB sono insiemi di coppie *chiave – valore*, vengono organizzati all'interno di collezioni tramite le quali è possibile raggruppare entità simili senza richiedere una struttura uniforme. Questa flessibilità rende MongoDB particolarmente adatto per gestire grandi volumi di dati non strutturati o semi-strutturati.

MongoDB supporta la scalabilità orizzontale attraverso il meccanismo dello sharding permettendo una distribuzione automatica dei dati su più server, questa caratteristica facilita l'ampliamento della capacità di elaborazione del database andando semplicemente ad aggiungere più server al sistema.

Un'altra importante caratteristica di MongoDB è l'uso dei Replica Set per garantire la disponibilità dei dati infatti, in caso di guasto di un nodo, le operazioni possono essere automaticamente reindirizzate verso un altro nodo del Replica Set assicurando così la continuità del servizio.

Passiamo ora al secondo database utilizzato nel progetto, ovvero Neo4j.

Neo4j è un database di tipo NoSQL orientato ai grafi, questa caratteristica è particolarmente utile in contesti come i social network o il rilevamento delle frodi nell'e-commerce in quanto permette di rappresentare le relazioni tra i dati in modo naturale.

Il modello di dati a grafo di Neo4j si basa su nodi, archi e proprietà. I nodi rappresentano le entità e incorporano al loro interno le proprietà li caratterizzano, gli archi possiedono una direzione e rappresentano le relazioni tra i nodi consentendo una rappresentazione visiva chiara delle relazioni tra i dati.

Uno dei punti di forza di Neo4j è il suo linguaggio di interrogazione Cypher che permette di formulare query complesse per recuperare dati specifici dai grafi.

L’ultimo aspetto di Neo4j di cui vogliamo parlare è la sua natura senza schema, infatti non è necessario definire uno schema rigido in anticipo consentendo quindi una maggiore flessibilità nell’aggiungere, modificare o rimuovere nodi, archi e proprietà.

# Design

Immagine che contiene testo, diagramma, cerchio, schermata

Descrizione generata automaticamente

Lo schema soprastante rappresenta le relazioni tra i dati: in particolare abbiamo che, per quanto riguarda gli utenti, i dati che andiamo a rappresentare sono:

* **ID Utente**: Identificativo univoco per ogni utente, generato con un numero casuale univoco.
* **Nome**: Il nome dell'utente.
* **Cognome**: Il cognome dell'utente.
* **Email**: L'email dell'utente.
* **Stato Account**: Lo stato dell'account dell'utente, selezionato casualmente da una lista di stati account.
* Riferimenti alla transazione e alla carta di credito, per poter effettuare eventuali operazioni di join in MongoDB

Passando alla carta di credito abbiamo:

* **Numero Carta di Carta**: Numero univoco della carta di credito.
* **Scadenza**: La data di scadenza della carta di credito.
* **Banca**: Il nome della banca che ha emesso la carta di credito.

Per quanto riguarda le transazioni abbiamo usato:

* **ID Transazione**: Identificativo univoco per ogni transazione.
* **Data**: La data in cui è stata effettuata la transazione
* **Metodo di Pagamento**: Il metodo di pagamento utilizzato, selezionato casualmente da una lista di metodi di pagamento.
* **Indirizzo IP**: L'indirizzo IP da cui è stata effettuata la transazione.
* **Browser**: Il browser utilizzato per effettuare la transazione.
* **Paese**: Il paese da cui è stata effettuata la transazione.
* Riferimento al prodotto per poter effettuare l’operazione di join in MongoDB

Proseguiamo con le segnalazioni di frode:

* **ID Segnalazione**: Identificativo univoco per ogni segnalazione di frode.
* **Data**: La data in cui è stata segnalata la frode.
* **Dettagli**: Dettagli specifici o descrizione della segnalazione di frode
* Riferimento alla transazione per poter effettuare eventuali operazioni di join in MongoDB

Ed infine per quanto riguarda il prodotto:

* **ID Prodotto**: Identificativo univoco per ogni prodotto.
* **Nome Prodotto**: Il nome del prodotto, selezionato casualmente da una lista di nomi di prodotti.
* **Categoria**: La categoria del prodotto, selezionata casualmente da una lista di categorie di prodotti.
* **Prezzo**: Il prezzo del prodotto.

Per quanto riguarda la loro generazione è stata usata faker, una libreria di Python che consente di generare dati fittizi. Di seguito viene riportato lo script per intero per generare i file csv i quali saranno successivamente importati dai database.

Immagine che contiene testo, schermata, software

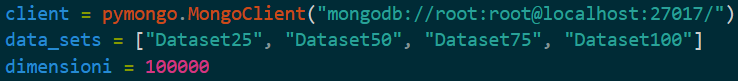
Descrizione generata automaticamente

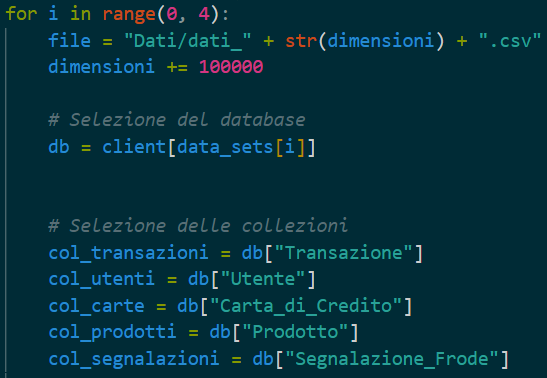
# Implementazione

In questa fase verranno riportate e descritte le parti principali degli script utilizzati per importare i quattro dataset nei due database considerati, successivamente verrà descritta l’implementazione delle quattro query. L'implementazione è stata eseguita sfruttando il linguaggio Python.

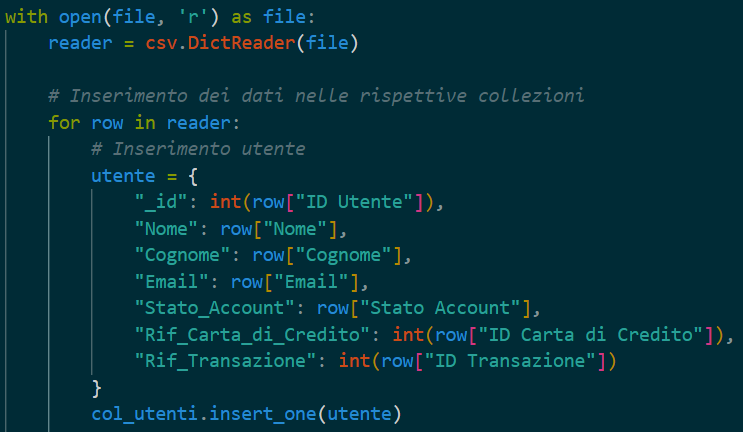
## *Inserimento dei dati*

Per quanto riguarda l’inserimento dei dati, partiamo da MongoDB.

****In questo primo blocco il programma si connette al server MongoDB utilizzando la funzione MongoClient di PyMongo. L'URL del server è specificato come parametro. Si definiscono le dimensioni del dataset e l'array data\_sets che contiene i nomi dei vari dataset utilizzati.



Questo blocco di codice esegue un ciclo attraverso i quattro dataset. Per ogni iterazione seleziona un file CSV basato sulla dimensione corrente del dataset, incrementa la dimensione del dataset per la prossima iterazione e poi seleziona il database MongoDB corrispondente. Successivamente, all'interno di ciascun database, seleziona le specifiche collezioni con cui lavorare: "Transazione", "Utente", "Carta\_di\_Credito", "Prodotto" e "Segnalazione\_Frode".



Per concludere nel seguente blocco di codice il programma apre in lettura il file CSV selezionato e crea un oggetto DictReader che consente di accedere ai dati del CSV come dizionari. Successivamente itera attraverso ogni riga del file CSV e per ognuna di queste estrae le informazioni necessarie, crea un nuovo dizionario che rappresenta un utente ed infine lo inserisce nella collezione "Utenti" del database. La procedura viene ripetuta per tutte le collezioni precedentemente indicate.

Passando ora a Neo4j l’importazione dei dati avviene in maniera simile tramite l’utilizzo della libreria py2neo. Vengono prima creati i nodi

Immagine che contiene testo, schermata, Carattere, software

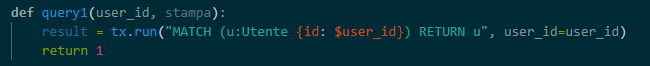
Descrizione generata automaticamente

Successivamente vengono create le relazioni seguendo lo schema del modello prima descritto.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

## Query

Per quanto riguarda la prima query vengono recuperati i dati di un utente specifico. Di seguito le rappresentazioni, rispettivamente in Neo4j e MongoDB.****

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

Passando poi alla seconda query vengono recuperati i prodotti il cui prezzo rientra nel range specificato. Anche in questo caso mostriamo prima la query in MongoDB e successivamente quella in Neo4j.

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente**

La terza query effettua una ricerca delle transazioni che sono state effettuate da un certo paese con un certo metodo di pagamento tra quelli a disposizione partendo da una data specifica. Di seguito le rappresentazioni, rispettivamente in Neo4j e MongoDB.

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Ed infine la quarta query ricerca le transazioni che riguardano prodotti che sono stati acquistati dopo una certa data, il risultato viene ordinato prima per data e poi per prezzo, viene prima mostrata l’implementazione in Neo4j e successivamente per MongoDB.

**Immagine che contiene testo, schermata, Carattere

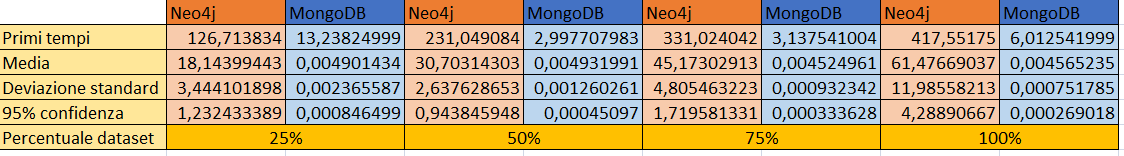
Descrizione generata automaticamente**

**Immagine che contiene testo, schermata

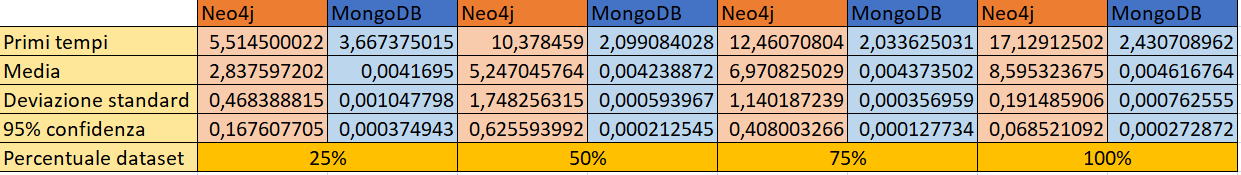
Descrizione generata automaticamente**

# Esperimenti

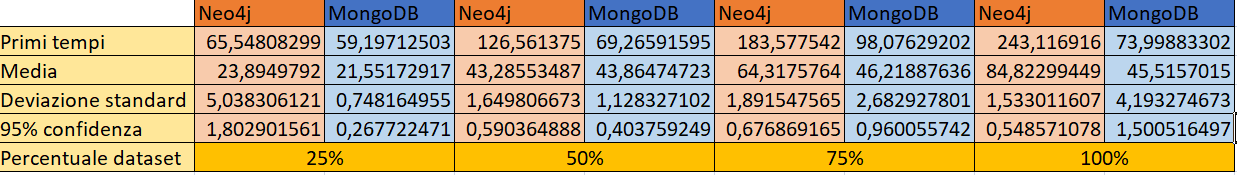
**1 Query**

****

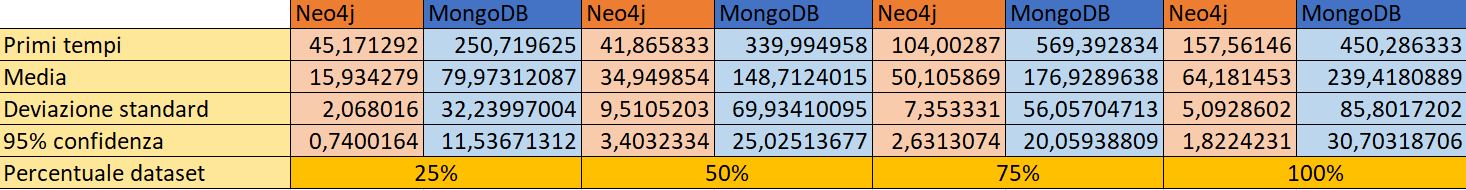
**2 Query**

****

**3 Query**

****

**4 Query**

****

# Conclusione con Analisi dei Risultati

Le query sulle quali sono stati eseguiti i benchmark sono equivalenti in termini di funzionalità ma sono strutturate in modi diversi a causa delle differenze nei linguaggi di query e nei modelli di dati dei due database. Ecco una analisi dei risultati:

* **1 Query**: MongoDB ha dimostrato prestazioni significativamente migliori rispetto a Neo4j, dove la media dei tempi di esecuzione era quasi trascurabile rispetto a quella di Neo4j.
* **2 Query**: Ancora una volta MongoDB ha avuto una media dei tempi di esecuzione inferiore rispetto a Neo4j anche se la differenza non è stata così drastica come nella prima query.
* **3 Query:** In questi risultati è interessante notare come la media dei tempi di esecuzione per MongoDB e Neo4j sembrano essere molto più vicini rispetto alle precedenti query. Tuttavia per il dataset completo MongoDB continua ad avere tempi di esecuzione leggermente migliori.
* **4 Query**: Nei risultati di quest’ultima query Neo4j ha dimostrato di avere un tempo medio di esecuzione inferiore rispetto a MongoDB, ciò evidenzia che Neo4j rappresenta una soluzione più efficace per l'elaborazione di interrogazioni che prevedono una gestione complessa di relazioni tra i dati.

Nell'analisi complessiva dei risultati si può osservare come MongoDB ottenga risultati più performanti nelle operazioni di interrogazione 'convenzionali', ovvero quelle principalmente focalizzate sulle proprietà degli oggetti. Dall’altra parte Neo4j mostra una superiorità nelle situazioni in cui la complessità delle relazioni è un fattore predominante. Concludendo possiamo affermare che la scelta tra i due sistemi potrebbe essere guidata dal tipo specifico di operazioni e dalla struttura dei dati con cui si prevede di lavorare.