



**Rapporto finale Big Data
Secondo Progetto 2021**

Omar Moh'd(510448) - Jomin Neelikatt(507937)

1 Introduzione

In ogni progetto, a prescindere dalle dimensioni, ci si imbatte nel problema di salvare e gestire i propri dati. Questi dati vengono spesso salvati in database, i quali sono software che permettono di migliorare l'accesso e la modifica di suddetti dati.

L'obiettivo di questo progetto è quello di analizzare le tre tipologie di database, la loro storia e il contesto in cui sono state create, testare vari database di test con almeno due software per ogni tipologia e confrontarne i risultati alla ricerca di intuizioni e spunti di riflessione.

2 Storia ed Evoluzione

La prima tipologia a venire introdotta è l'SQL (Structured Query Language) durante gli anni '70. Questa tipologia segue un modello relazionale basato su tabelle diverse che fra loro possono avere relazioni. Ogni campo ha degli attributi, definiti da un tipo.

Il linguaggio SQL permette l'esecuzione di svariate operazioni ma quelle fondamentali rimangono quelle denominate CRUD(Create, Read, Update, Delete) che per l'appunto servono per la creazione, lettura, aggiornamento e rimozione dei dati.

Possiamo definire come transazione qualsiasi sequenza di operazioni che genera un cambiamento nello stato del database. Per il modello relazionale ogni transazione garantisce delle proprietà, definite come ACID, ossia:

- atomicità: ossia le operazioni possono essere divise in un numero finito di sottoazioni definite come transazioni - da ciò l'esecuzione di una transazione deve essere completa o nulla.
- coerenza: il database segue le regole imposte, ad esempio vincoli d'integrità, sia prima che dopo l'esecuzione della transazione.
- isolamento: ogni transazione viene eseguita in maniera indipendente dalle altre.
- durabilità: questa proprietà si riferisce alla persistenza delle transazione, ossia al fatto che le variazioni non vengano perse.

Nel tempo, soprattutto a partire dagli anni 2000, anche per l'ingente crescita di dati e lavoro su esso, si affermano altre filosofie per quanto riguarda i database. La tipologia NoSQL propone alternative efficienti al classico database relazionale. La corrente NoSQL nasce principalmente per arginare alcune lacune del suo concorrente come una più ampia e sostenibile scalabilità, soprattutto quando si ha l'esigenza di lavorare con dati non strutturati.

I database NoSQL fanno uso di coppie chiave-valore, di dati sottoforma di documento(ogni record è definito come documento ed è formato da alcune caratteristiche), di grafo(nodi e archi per rappresentare l'informazione) e così via.

Da sottolineare che il NoSQL non è un rifiuto dell'SQL ma bensì una proposta di miglioramento - infatti un'interpretazione della sigla può essere 'Not Only SQL'.

Nonostante un acceso entusiasmo sulla corrente si è notato come molti di questi database non supportino le proprietà ACID ed altre criticità come l'esistenza implicita di una struttura e di assunzione d'esistenza dei campi. Dall'altro lato c'è da evidenziare una flessibilità elevate, ottime proprietà di caching, scalabilità orizzontale(i dati sono divisi in vari server senza che l'applicazione conosca effettivamente tutti i server).

Dalle considerazioni soprantanti, nasce, nell'ultimo decennio l'approccio NewSQL che cerca di unire le due tipologie precedenti ottenendo un database relazione con proprietà di scalabilità affidabili, oltre ad ottenere un numero elevato di operazioni online, tutte conformi alle proprietà ACID.

Di seguito una tabella riepilogativa delle tre tipologie.

Proprietà	SQL	NoSQL	NewSQL
Proprietà relazionali	Si	No	Si
Proprietà ACID	Si	No	Si
SQL	Si	No	Supportato e migliorato
OLTP	Inefficiente	Supportato ma non ideale	Supportato ed efficiente
Scalabilità	Verticale	Orizzontale	Verticale ed Orizzontale
Query	Ottima gestione per quelle semplici	Migliore dell'SQL per le query complesse	Efficiente con qualsiasi tipo di query
Database distribuito	No	Si	Si

3 Obiettivi del Progetto

In questo progetto per il corso di Big Data abbiamo cercato di sperimentare intensivamente vari dataset con 6 database diversi (PostgreSQL, MariaDB, Neo4j, MongoDB, Singlestore, TiDB), rispettivamente due per ogni tipologia di database (SQL, NoSQL, NewSQL), al fine di estrapolare intuizioni e considerazioni e trovare, in caso, una tipologia di database generalmente migliore delle altre.

Di seguito, per ogni tipologia di database, verranno esplicitati i due database usati con relativi esperimenti e considerazioni effettuate.

4 Analisi Sperimentale

Inizialmente, per prendere confidenza con i vari database, abbiamo 'giocato' con vari dataset dummy per testare le varie funzionalità e capire maggiormente le qualità d'ognuno.

Successivamente, abbiamo scelto sei dataset (tre strutturati e tre non strutturati) con i quali testare i database.

4.1 Datasets

- Cambio Euro-Dollaro 1999/2021, CSV di due colonne(data, valore cambio) con circa 6k righe.
- Nomi di bambini nati in Francia dal 1900 al 2016, file TSV con quattro colonne(sesso, nome, anno e n° nati) con circa 620k righe.
- Crimini commessi ad Atlanta nel 2017, CSV con 11 colonne(id progressivo, crimine, id crimine, data, indirizzo, quartiere, latitudine, longitudine, coordinate, municipio) con circa 270k righe.
- Album di foto stock, 5000 foto per un totale di 815 MB.
- Recensioni IMDB di film, 100k recensioni in formato .txt
- Riassunti di casi giudiziari, 4k casi descritti sottoforma di file .xml

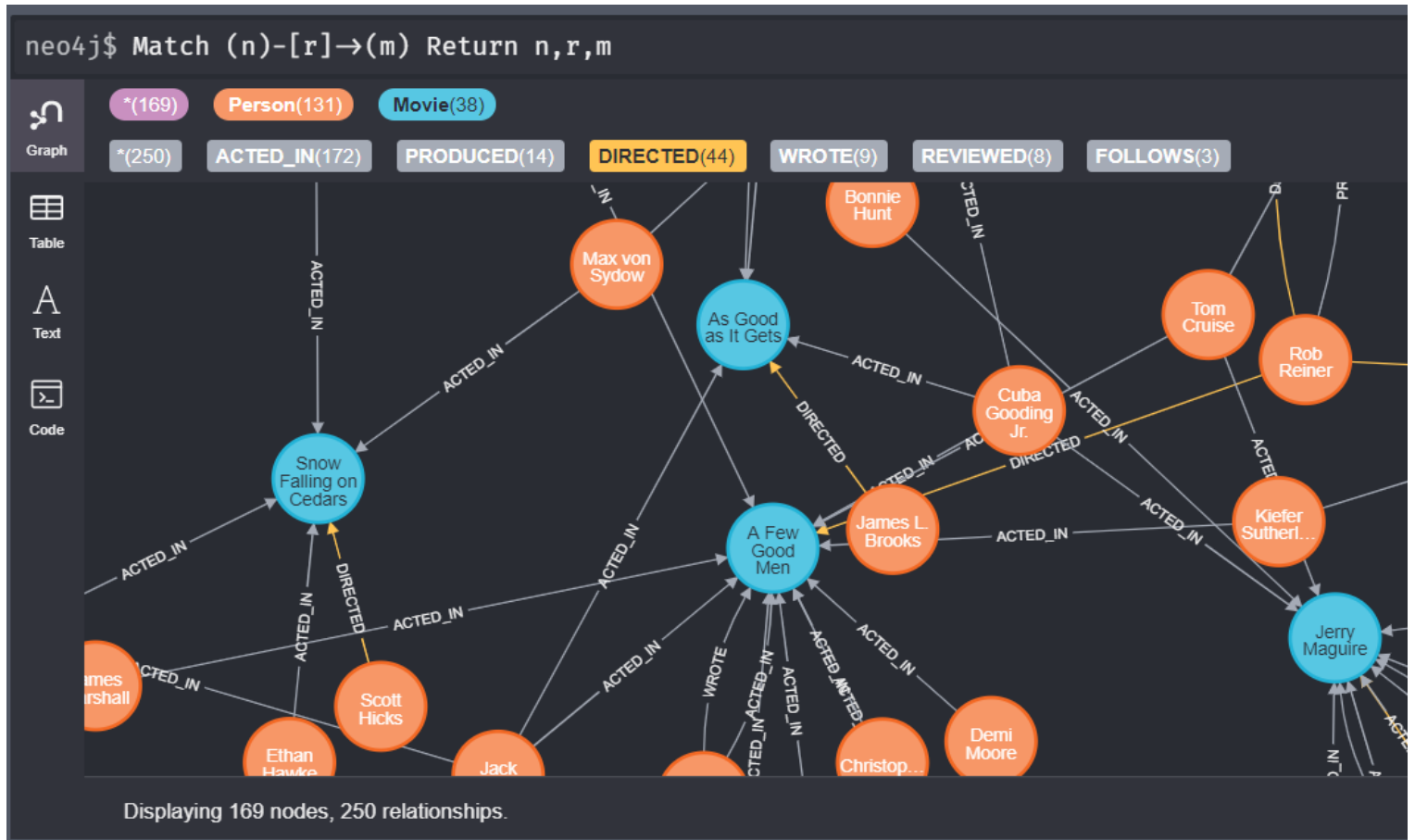


Figure 1: Esempio di dataset dummy usato per Neo4j - 169 nodi e 250 relazioni fra film, attori e registi.

Oltre a sperimentare con i vari dataset, per avere anche dati da confrontare sono state realizzate delle query per i dataset strutturati così da poter misurare i tempi delle esecuzioni per i vari database. Il processo è stato ripetuto per i dataset amplificati(x4 e x8).

Di seguito qualche appunto generale su ogni database usato rispetto ad ogni dataset. Infine nei capitoli 5 e 6 si possono trovare considerazioni finali sia sui database che sul tipo di database(SQL, NoSQL, NewSQL).

4.2 SQL

4.2.1 PostgreSQL

PostgreSQL è un potente database relazionale open source che utilizza ed estende il linguaggio SQL combinato con molte funzionalità che archiviano e ridimensionano in modo sicuro i carichi di lavoro di dati più complicati. Le origini di PostgreSQL risalgono al 1986 come parte del progetto POSTGRES e ha più di 30 anni di sviluppo attivo sulla piattaforma principale.

Le sue caratteristiche principali sono architettura, affidabilità, integrità dei dati, robusto set di funzionalità, estensibilità e dedizione della comunità open source dietro il software per fornire costantemente soluzioni performanti e innovative.

Sperimentazione

Dataset 1 - Cambio Euro/Dollaro Grazie all'utilizzo di pgadmin che è il tools management più utilizzato per PostgreSQL, molte operazioni elementari sono state di molto semplificate. Ad esempio creazione, analisi e gestione delle tabelle e delle viste, la possibilità di fare import di file direttamente dalla finestra di dialogo senza scrivere codice, il query tool per interrogare il db, la visualizzazione dei risultati e così via. Nello specifico il caricamento di questo dataset in formato csv è stato molto rapido come anche l'esecuzione delle query.

Dataset 2 - Nomi bambini francesi La particolarità di questo dataset è il formato tsv del file. PostgreSQL non supporta nativamente l'import di file tsv(è possibile scegliere tra csv, binario o testuale), ma questo può essere fatto semplicemente scegliendo come formato csv e come separatore il "tab"(tipico dei file tsv) al posto della ","(tipico dei file csv). In questo modo la conversione verrà effettuata in automatico. I tempi di caricamento ed esecuzione

change/postgres@Local PG ▾

Query Editor Query History

```
1 DROP VIEW IF EXISTS odd_year CASCADE;
2
3
4 CREATE VIEW odd_year AS
5 SELECT * FROM dataset
6 WHERE dataset IS NOT null
7 AND (extract(year from dataset.date)::integer)%2=1;
8
9
10 SELECT SUM(a.change)/COUNT(*) odd_mean
11 FROM odd_year AS a;
```

Explain Messages Data Output Notifications

	odd_mean numeric	🔒
1	1.1990645622602023	

Figure 2: Risultato da PostgreSQL.

query risultano maggiori rispetto al precedente dataset poichè questo è molto più grande.

Dataset 3 - Crimini commessi ad Atlanta Valgono le considerazioni precedenti.

Dataset 4 - Album di foto stock L'inserimento di un'immagine è molto efficiente ed intuitivo. Le difficoltà tecniche sorgono quando si lavora con molte immagini, anche con l'utilizzo del tools management più famoso per PostgreSQL, pgadmin, la situazione non cambia. Le immagini sono dati binari, il tipo di dati standard nei database è BLOB. Però il database PostgreSQL ha un tipo di dati speciale per memorizzare dati binari chiamato bytea. Questo è un tipo di dati non standard. Si crea una tabella con una colonna che contiene dati di tipo bytea e si importano le immagini:

```
CREATE TABLE IF NOT EXISTS public.images
(
    id numeric,
    image bytea
)
```

)

```
INSERT INTO images(id, image) VALUES (0, bytea('/home/omar/Scrivania/  
Big Data/Secondo Progetto/Datasets/stock_album/000000000285.jpg'))
```

La tabella contiene ora l'id e l'immagine in formato binario. Per l'inserimento di immagini multiple si potrebbe effettuare manipolazione testuale dei dati. Un inserimento ha tempo che varia dai 250 ai 500 msec, per il dataset utilizzato che contiene 5000 immagini il tempo è di circa 29 minuti. Invece di importare l'immagine dal path si potrebbe convertirla in base64 ed importare direttamente la stringa:

```
INSERT INTO images(id, image) VALUES (0, '<base64 string>'))
```

Dataset 5 - Recensioni IMDB Per il dataset movie_reviews, che contiene file testuali, è stato utilizzato un inserimento multiplo manuale in formato testuale tenendo in considerazione i problemi riscontrati in precedenza per l'import di file multipli. Ogni insert ha tempo di circa 200 msec.

```
CREATE TABLE IF NOT EXISTS public.reviews  
(  
    testo text  
)
```

Dataset 6 - Riassunti di casi giudiziari Per il dataset case_summary contenente file di tipo .xml si sono verificati gli stessi problemi dato che PostgreSQL non permette un'inserimento efficiente di file multipli. Una soluzione è quella di convertire i file in formato csv ed importarli come normali tabelle.

```
CREATE TABLE IF NOT EXISTS public.cases  
(  
    id text,  
    text text  
)
```

4.2.2 MariaDB

MariaDB è un database relazionale open source, nato da un fork di MySQL. Sono presenti gli Storage Engine (librerie per la gestione dei dati) originali di

	testo text
1	Once again Mr. Costner has dragged out a movie for far longer than neces
2	I went and saw this movie last night after being coaxed to by a few friends
3	My boyfriend and I went to watch The Guardian.At first I did not want to w
4	It seems ever since 1982, about every two or three years we get a movie t
5	My yardstick for measuring a movie s watch-ability is if I get squirmy. If I s
6	Wow, another Kevin Costner hero movie. Postman, Tin Cup, Waterworld, B
7	How many movies are there that you can think of when you see a movie li
8	Alas, another Costner movie that was an hour too long. Credible performa
9	This movie was sadly under-promoted but proved to be truly exceptional.
10	I wish I knew what to make of a movie like this. It seems to be divided into

Figure 3: Risultato dataset txt.

MySQL. Sono inoltre presenti connettori per sorgenti dati esterne o remote, nonchè istanze per il partizionamento e connettori per DB NoSQL come Cassandra(CassandraSE).

Inoltre, MariaDB nasce anche per migliorare le performance - ciò è stato perseguito aumentando il numero di connessioni possibili, aggiungendo tipologie di indici o incrementando la velocità di ricerca di quelli esistenti, introducendo o ampliando meccanismi di cache di dati a supporto dei lavori di elaborazione e rendendo disponibili le cosiddette “colonne virtuali” automaticamente calcolate al momento del bisogno in base ad una funzione deterministica.

test/postgres@Local PG	
Query Editor	Query History
<pre> 1 SELECT * FROM public.cases 2 </pre>	
Explain	Messages
Data Output	Notifications
id	text
51	s39 9 The Sharman applicants sought to emphasise the existence of practical burdens imposed by the orders made by Moore J on 17 November 2005, although in reality it was principally with respect to the order for cross-examination of Ms He
52	s40 They maintained that the requirement to appear for cross-examination was not analogous to the 'routine steps' of swearing an affidavit during the normal course of a trial and entering the witness box to answer questions related to its conten
53	s41 Rather, the requirement to be cross-examined on an asset disclosure affidavit was said to be 'at the extreme end of the Court's armoury', by virtue of the fact that the making of an order for cross-examination on an affidavit of this kind is pred
54	s42 I was referred by the Sharman applicants to an earlier Full Federal Court decision in Minister for Immigration & Multicultural & Indigenous Affairs v Wong [2002] FCAFC 327 in support of their submission that since the present circumstances
55	s43 The parties' respective submissions concerning material matters bearing upon the justification or otherwise for the grant of orders ancillary to the Mareva relief and my incidental observations
56	s44 Any subsequent challenge to those orders would then be futile and academic.
57	s45 The orders are interlocutory in form but final in the result in relation to the issues raised.
58	s46 'So much is correct as a general statement as to the consequences of his Honour's orders.
59	s47 However, the nature and context of the application for leave here involved in practical terms an appeal, or virtually so, as will hereafter emerge.
60	s48 'The Music companies contended that the Sharman applicants' submissions proceeded in reality 'on the erroneous basis that every interlocutory decision that requires a party to do an act will be automatically susceptible to leave to
61	s49 The Music companies' position was that the orders requiring Ms Hemming to file an affidavit disclosing the assets of Sharman Networks, and to attend for cross-examination on the affidavit she had earlier sworn as to disclosure of her asset
62	s50 So much accorded with Moore J's understanding of the nature of those orders, as indicated in [30] of his Honour's reasons for judgment.
63	s51 12 The primary submission made by the Music companies as to why leave to appeal ought not to be granted related to the failure of the Sharman applicants to adduce evidence demonstrative of the claim that they would suffer 'substantial ir
64	s52 The Music companies rejected any notion to the effect that the mere fact of Ms Hemming being compelled to swear an affidavit as to disclosure of Sharman Networks' assets, and to attend for cross-examination upon her own disclosure aff
65	s53 Reference was made to Gerlach v Clifton Bricks Pty Ltd [2002] HCA 22, (2002) 209 CLR 478, in which Gaudron, McHugh and Hayne JJ, in a different context, considered that the bare proposition that an order mandating 'trial by judge alone, a
66	s54 The reality of the dilemma in which the Court is placed by the present application is the need for consideration in some depth of the matters to which attention has been drawn by the Sharman applicants, before determining whether leave to
67	s55 The Sharman applicants relied upon a further passage from the majority's reasons for judgment in Gerlach at [13], as follows: 'The principles governing the grant of leave to appeal against interlocutory orders are well established...If it is
68	s56 In refusing leave to appeal from Wilcox J's decision to decline to set aside earlier Anton Piller orders that had been made in the course of the subject litigation, their Honours were said not to have considered at all the merits of the contempla
69	s57 The following passages appearing at [7]-[8] and [13]-[14] of their Honour's reasons for judgment in Brilliant Digital were asserted by the Music companies to be apposite to the present application: 'In our view these matters [the matte
70	s58 While it is neither necessary nor appropriate to attempt to define the concept of substantial injustice some observations may be ventured.
71	s59 The flexibility of the principles governing the grant of leave to appeal indicates that the concept of substantial injustice must also be flexible.
72	s60 The requirement of leave to appeal indicates, however, that substantial injustice requires something more than that the subject decision is incorrect, otherwise the criterion would be superfluous.
73	s61 The qualification of injustice by substantial points to a detriment that, while not necessarily irreparable, is more than mere inconvenience or delay in the exercise of a right.
74	s62 'The distinction between interlocutory decisions concerning matters of practice and procedure and those that concern the substantive interests of the parties recognises both the greater likelihood of an incorrect decision as to a sub
75	s63 In determining whether there has been substantial injustice it is appropriate for the court to take these factors into account.
76	s64 The need to keep a tight rein upon interference with orders at first instance that do not determine substantive rights has even more force today in the context of procedural reforms and active case management undertaken with a view to the
77	s65 They were directed to the capture and preservation of certain data and information and this purpose has now been effected.
78	s66 Importantly, the applicants for leave do not seek to reverse this by destruction of the material seized pursuant to the challenged orders.

Figure 4: Risultato xml.

Sperimentazione

Dataset 1 - Cambio Euro/Dollaro Il caricamento del file CSV è stato abbastanza agevole anche grazie all’interfaccia grafica che trasforma i settaggi in una query per l’import dei dati. Per quanto riguarda l’esecuzione delle query, il database si è dimostrato solido anche per il file x8.

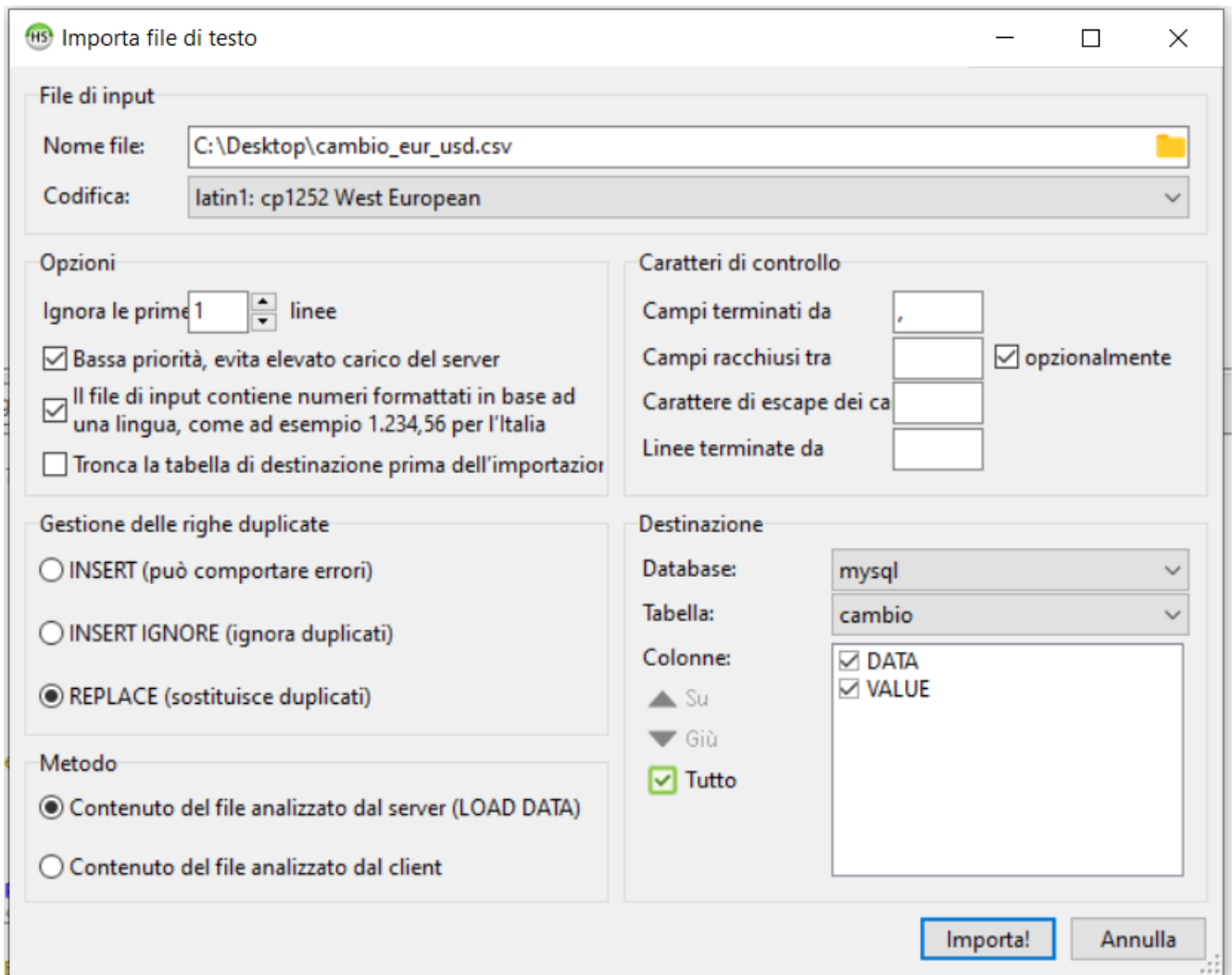


Figure 5: Il cliente HeidiSQL permette la conversione da impostazioni a query.

Dataset 2 - Nomi bambini francesi La particolarità del secondo file, oltre ad essere il più grande fra i datasets, è quello di essere in formato tsv. MariaDb in particolare non supporta nativamente questo formato - è stato necessario una conversione in CSV, alternativamente si possono essere prodotti terzi per ottenere la compatibilità come Withdata([Connettore TSV MariaDB](#)). Detto ciò, il caricamento e l'esecuzione delle query è risultata non immediata, anche date le dimensioni dei datasets(620k righe, 2M righe e 5M righe), mante-

nendo comunque proporzioni di tempistiche minori all'aumentare del dataset(se il dataset veniva quadruplicato, i tempi si raddoppiavano/triplicavano al più).

Dataset 3 - Crimini commessi ad Atlanta Per questo file strutturato(CSV) i tempi sono simili al dataset precedente nonostante questo abbia meno della metà di ennuple(270k) - ciò è dovuto al fatto che in questo dataset si hanno ben 11 colonne fitte d'informazioni. E' stata necessaria qualche operazione preliminare di pulizia.

Dataset 4 - Album di foto stock Anche in questo caso MariaDB è stato efficiente, sempre grazie all'interfaccia utente è stato possibile caricare facilmente le immagini (formato longblob, ossia le informazioni delle immagini vengono salvate come informazioni binarie) tramite il costrutto `%filecontent%` - per 5000 foto sono stati impiegati circa 6 minuti. Per avere prestazioni migliori, è possibile caricare le foto in storage cloud(ad esempio S3 di AWS) e poi dare a MariaDB solamente i riferimenti([Integrazione MariaDB e S3](#)).

Dataset 5 - Recensioni IMDB Questo dataset non strutturato contiene 100k file .txt con recensioni cinematografiche. MariaDB permette il caricamento di file blob o longtext ma il caricamento è risultato molto prolisso(1 ora e 40 minuti) e non esente da problematiche, soprattutto legate a formati e codifiche.

Dataset 6 - Riassunti di casi giudiziari Casi giudiziari - MariaDB non ha il tipo XML (PostgreSQL o Microsoft SQL Server lo supportano), quindi abbiamo caricato con l'opzione apposita i vari xml riconoscendoli in base ai tag. E' stato leggermente macchinoso ma comunque fattibile. In alternativa, una modalità più veloce è caricare ogni riassunto come un LONGBLOB, il caricamento risulta più veloce ma si perde il senso dell'xml.

4.3 NoSQL

4.3.1 MongoDB

MongoDB(da "huMONGOus", enorme) è un database distribuito di tipo NoSQL orientato ai documenti, quindi non relazionale. MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti flessibili, simili a JSON, rendendo l'integrazione di dati di alcuni tipi

1	<code>SELECT * from reviews;</code>
reviews (100.249r × 1c)	
Review	Robert DeNiro plays the most unbelievably intelligent illiterate of all time. This mo...
	My boyfriend and I went to watch The Guardian.At first I didn't want to watch it, b...
	This is a pale imitation of 'Officer and a Gentleman.' There is NO chemistry betwee...
	If you like adult comedy cartoons, like South Park, then this is nearly a similar form...
	William Boyd and Louis Wolheim are the "Two Arabian Knights" referred to in the t...
	I saw the capsule comment said "great acting." In my opinion, these are two great ...
	It seems ever since 1982, about every two or three years we get a movie that claims...
	My yardstick for measuring a movie's watch-ability is if I get squirmy. If I start shift...
	Bromwell High is nothing short of brilliant. Expertly scripted and perfectly delivere...
	Not very many movies come to my mind that covered as much geography as this ...
	"All the world's a stage and its people actors in it"--or something like that. Who th...

Figure 6: Recensioni cinematografiche.

di applicazioni più facile e veloce. Query ad hoc, indicizzazione e aggregazione in tempo reale forniscono potenti modi per accedere e analizzare i dati. Inoltre l'alta disponibilità, l'alta scalabilità orizzontale e la distribuzione geografica sono integrati e facili da usare.

Sperimentazione

Dataset 1 - Cambio Euro/Dollaro Superate le iniziali difficoltà, dovute ovviamente all'apprendimento di un linguaggio completamente nuovo, ovvero MongoDB Query Language(MQL), è stato subito molto intuitivo e soddisfacente utilizzare le funzionalità messe a disposizione. Sono state utilizzate in cooperazione la shell classica di MongoDB e il tools management MongoCompass. Quest'ultimo utilizzato più per la gestione del db e delle collezioni che per le query. L'import del csv è stato fatto con un tool appositamente installato (che consente di fare anche molte altre cose, come caricamento di immagini o altri files): mongotools.

```
mongoimport --type csv -d progetto2 -c change --headerline --drop eur_usd_change.csv
```

Import molto veloce, ordine di pochi millisecondi. Per le query si è invece deciso(al contrario degli altri due datasets con query) di modificare direttamente la collezione iniziale, per fare ulteriori test sui comandi principali.

```
db.change.updateMany({}, [{ $project: { year: { $substr: ["$DATE", 0, 4] },
dexuseu: "$DEXUSEU" } }]); --PERSISTENTE: estraggo solo anno
db.change.updateMany({}, [ { $project: { year: { $toInt: ["$year"] },
dexuseu: "$dexuseu" } } ]]);
db.change.updateMany({}, [{ $project: { remainder: { $mod: [ "$year", 2 ] }, year:"$year",
dexuseu: "$dexuseu" } }]);
db.change.remove({"remainder":{"$eq:0"}});
db.change.update({}, {$set : {"id":1}}, {upsert:false, multi:true});
db.change.aggregate([
    { $group: { _id: "$id", sum: { $sum: "$dexuseu" }, count: { $sum: 1 } } },
    { $project: { mean: { $divide: [ "$sum", "$count" ] } } }]); --visualizzo risultato
```

Dataset 2 - Nomi bambini francesi L'import del tsv è supportato allo stesso modo del csv, specificando ovviamente il tipo corretto.

```
mongoimport --type tsv -d progetto2 -c france --headerline --drop france_newborns.tsv
```

Le query sono state create per la sola visualizzazione, non è stata modificata la collezione originale importata(a parte per i valori nulli). Sono state create due collezioni d'appoggio france2 e france3 per effettuarne poi il join con il comando "\$lookup".

```
db.france.aggregate([
    { $group: { _id: "$preusuel", occorrenze: { $sum: "$nombre" } } },
    { $project: { name: "$_id", occorrenze: "$occorrenze" } },
    { $match: { occorrenze: { $gt: 20000 } } },
    { $project: { name: "$name", occorrenze: "$occorrenze" } },
    { $out: { db: "progetto2", coll: "france2" } }]);
db.france.aggregate([
    { $group: { _id: "$preusuel", occorrenze: { $sum: "$nombre" } } },
    { $project: { name: "$_id", occorrenze: "$occorrenze" } },
    { $match: { occorrenze: { $gt: 20000 } } },
    { $project: { name: "$name", occorrenze: "$occorrenze" } },
```

```

    { $out: { db: "progetto2", coll: "france3" } }]);
db.france2.aggregate([
  { "$lookup": {
    "from": "france3",
    "let": { "name": "$name" },
    "pipeline": [{
      "$match": {
        $expr: { $regexMatch: { input: "$name", regex: { $cond: { if:
          {$eq:["$name", "$$name"]}, then: "null", else: "$$name" } } } } }
      },
      "as": "similar" } }],
    { $match: { similar: { $exists: true, $not: { $size: 0 } } } } ]]);

```

Dataset 3 - Crimini commessi ad Atlanta Anche in questo caso le query sono state create per la sola visualizzazione.

```

db.crimes.aggregate([
  { $group: { _id: "$neighborhood", numberOfCrimesSameNeighborhood: { $sum: 1 } } }]);
db.crimes.aggregate([
  { $group: { _id: { neighborhood: "$neighborhood", date: "$date",
    crime: "$crime" } } },
  { $set: { numberOfCrimes: 1 } },
  { $project: { neighborhood: "$_id.neighborhood", date: "$_id.date",
    crime: "$_id.crime", numberOfCrimes: "$numberOfCrimes" } },
  { $group: { _id: { neighborhood: "$neighborhood", date: "$date" }, numberOfCrimes:
    { $sum: "$numberOfCrimes" }, crimes: { $push: "$crime" } } },
  { $project: { neighborhood: "$_id.neighborhood", date: "$_id.date", numberOfCrimes:
    "$numberOfCrimes", crimes: "$crimes" } },
  { $sort: { "numberOfCrimes": -1 } },
  { $group: { _id: "$neighborhood", sameDay: { $first: "$numberOfCrimes" }, crimes:
    { $first: "$crimes" }, date: { $first: "$date" } } },
  { $sort: { "sameDay": -1 } } ],
  { "allowDiskUse" : true });

```

Dataset 4 - Album di foto stock Per le immagini ci sono stati non pochi problemi dato che nè Compass(come detto in precedenza) nè la shell stessa di MongoDB, tramite il comando "mongoimport" utilizzato per il caricamento dei tre datasets strutturati, prevedono l'inserimento.

Una possibile soluzione, come visto per PostgreSQL, è la conversione dell'immagine in formato base64 e il relativo import tramite file json o csv.

Visti i precedenti problemi gli sviluppatori di MongoDB hanno introdotto nel passato recente uno strumento molto potente di gestione dei file: mongofiles.

Dataset 5 - Recensioni IMDB Per quanto riguarda invece MongoDB, il tools management utilizzato ovvero MongoDB Compass permette solo l'inserimento di file in formato json o csv. È stato quindi molto vantaggioso per i file xml e txt effettuare una conversione in uno dei due formati csv/json. A quel punto l'inserimento è risultato intuitivo e davvero molto efficiente, ma comunque macchinoso vista la grande quantità di operazioni di manipolazione da compiere.

Dataset 6 - Riassunti di casi giudiziari Stesse considerazioni del dataset precedente.

Visti i precedenti problemi gli sviluppatori di MongoDB hanno introdotto nel passato recente uno strumento molto potente di gestione dei file: mongofiles.

Dopo aver installato mongo tools, per inserire un file(o una lista di files) di qualsiasi tipo all'interno del db scelto(in questo caso "images") basta utilizzare l'apposito comando put:

```
mongofiles -d=images put 000000000785.jpg
```

Sperimentazione I caricamenti sono all'ordine di pochi millisecondi. I files sono immagazzinati in una struttura speciale chiamata GridFS. Invece di archiviare un file in un singolo documento, GridFS divide il file in blocchi e memorizza ogni blocco come un documento separato. Quando si interroga GridFS per un file, il driver riassembla i blocchi secondo la necessità. Inoltre usa due collezioni per archiviare i file, una archivia i blocchi di file e l'altra i metadati. La prima immagine rappresenta il modo in cui MongoDB archivia i blocchi, la seconda i files interi.

4.3.2 Neo4j

Neo4j è un database a grafo open source. Ovviamente, tutte le operazioni sono massimizzate per strutture a grafi come file XML, filesystem e reti, che possono essere rappresentate come grafi. L'esplorazione di queste strutture risulta in genere più veloce rispetto a un database a tabelle perché la ricerca di nodi in

relazione con un certo nodo è un'operazione primitiva e non richiede più operazioni, su tabelle diverse.

Ogni nodo contiene l'indice delle relazioni entranti e uscenti da esso, quindi la velocità di attraversamento del grafo non risente delle dimensioni complessive ma solo della densità dei nodi attraversati.

Più il grafo è connesso, migliori sono le prestazioni. Ha un linguaggio di query tutto suo e consente le operazioni di tipo ACID.

Sperimentazione

Dataset 1 - Cambio Euro/Dollaro Caricamento del file csv efficiente. Il linguaggio CQL(Cypher Query Language) dopo iniziali difficoltà risulta facile da utilizzare. Di seguito una delle query usate, in dettaglio quella per estrarre la media del valore cambio euro-dollaro negli anni dispari.

```
LOAD CSV FROM
    'file:///cambio\_eur\_usd.csv' AS row
WITH
    split (row[0], '-') AS data, toFloat(row[1]) AS valore
WHERE
    toInteger(data[0])%2=1
RETURN
    (sum(valore))/(count(valore));
```

Dataset 2 - Nomi bambini francesi Caricamento abbastanza efficiente(supporta file TSV). Destabilizzante il fatto di non poter usare le viste, ma c'è da sottolineare che questo è un prodotto pensato principalmente per dataset a grafo.

```
LOAD TSV FROM 'file:///france_names.csv' AS row
WITH row[1] as name, row[2] as year,row[3] as num
RETURN distinct name, toInteger(num) as n
ORDER BY n desc;
```

Dataset 3 - Crimini commessi ad Atlanta Stesse considerazioni per il dataset precedente.

Dataset 4 - Album di foto stock Per questo dataset, anche cercando online, viene consigliato, come uno dei pochi 'anti-use-cases' di questo database di non aggiungere tanti nodi di contenuto multimediale, ma piuttosto di caricare le foto in formato binario in ambiente cloud(AWS S3 o GCP) e poi creare un nodo che avesse il riferimento(url, ma anche metadati come dimensioni, qualità, codifica). In questo ambito le relazioni possono essere usate per esprimere concetti come commenti, permessi, proprietari di immagine ed azioni di tagging.

Dataset 5 - Recensioni IMDB Neo4j ha un caricamento molto più veloce rispetto a MariaDB ma comunque è stato necessario convertire i vari file .txt in csv tramite uno script Python altrimenti non è possibile(da sottolineare che Neo4j è pensato per grafi). Di seguito lo script Python usato.

```
import os

with open('reviews.csv', mode='a', encoding="utf8") as file_primary:
    for file in os.listdir('C:/Users/Jomin/Desktop/ac1Imdb'):
        with open('C:/Users/Jomin/Desktop/ac1Imdb/' + file, encoding="utf8") as f:
            data = f.read()
            file_primary.write(data + ';\n')
```

Dataset 6 - Riassunti di casi giudiziari Per questo database, questo è il dataset più interessante proprio perchè un file XML, può esser interpretato come un grafo. I caricamenti sono stati semplici, anche grazie alla libreria apoc(<https://neo4j.com/developer/neo4j-apoc/>). Nell'esempio sotto riportato si può pensare di avere un nodo per ogni caso e un nodo per ogni riassunto e collegarli per id corrispondente, per ampliare il grafo si possono anche introdurre giudici o stati così da collegarli ai casi in cui sono coinvolti. C'è da sottolineare che più i nodi sono collegati, meglio funziona lo strumento, ma che esagerando con i nodi, almeno in ambiente di test(quindi con un pc non da produzione) si rischiano rallentamenti notevoli, principalmente sulla visualizzazione del grafo che appare confusionaria, anche se c'è da dire che le query funzionano molto bene dato che navigano sulle varie relazioni

4.4 NewSQL

4.4.1 TiDB

TiDB("Ti" sta per Titanium) è un database NewSQL open source e distribuito che supporta i carichi di lavoro Hybrid Transactional and Analytical Processing (HTAP). È compatibile con MySQL(supporta quindi il linguaggio SQL) e offre scalabilità orizzontale, forte consistenza e alta disponibilità. Progettato per il cloud, TiDB offre affidabilità e sicurezza sulla piattaforma cloud. Inoltre scala in modo elastico per soddisfare i requisiti dei carichi di lavoro in continua evoluzione. Anche l'integrazione con Kubernetes è un punto a favore, TiDB Operator aiuta a gestirla e automatizza le attività operative.

Sperimentazione

Dataset 1 - Cambio Euro/Dollaro Appoggiandosi su MySQL, è stata utilizzata la sintassi comune ai linguaggi SQL. Naturalmente si differenzia però dai normali SQL per la velocità d'esecuzione delle query, ma un pò meno per la velocità d'import dato che è stato effettuato direttamente da file locali al filesystem. Tempi di import e esecuzione query risultano comunque coerenti con quanto detto.

Dataset 2 - Nomi bambini francesi L'import del file in formato tsv non ha avuto particolari problemi. Ricalcando il punto espresso per il dataset precedente, si noti che il load del file x8 ha impiegato circa 54 secondi contro l'esecuzione delle query di circa 5 secondi

Dataset 3 - Crimini commessi ad Atlanta Stesse considerazioni fatte in precedenza.

Dataset 4 - Album di foto stock Prima dei caricamenti il cluster deve essere collegato aggiungendo al comando "--local-infile=1" per supportare il caricamento locale.

Supponendo sempre di avere il dataset in locale, dato che TiDB si poggia su mySQL, similmente a PostgreSQL si può creare una tabella che contiene un id e un campo di tipo BLOB. Quest'ultimo è infatti il tipo di dati standard nei

database. NB: l'utente usato per connettersi al cluster deve avere i diritti sui file.

```
CREATE TABLE images
(
    id int,
    image blob
);
```

```
INSERT INTO images(id, image) VALUES(0, '/home/omar/Scrivania/Big Data/Secondo Progetto/Datasets/stock_album/000000000285.jpg');
```

Si caricano poi le immagini che avranno formato binario. Ogni caricamento ha tempo di circa 0,20 secondi.

Dataset 5 - Recensioni IMDB Allo stesso modo l'inserimento dei file testuali segue i comandi mySQL come "LOAD DATA".

```
CREATE TABLE reviews
(
    id int,
    review varchar(500)
);
```

```
LOAD DATA LOCAL INFILE '/home/omar/Scrivania/Big Data/Secondo Progetto/Datasets/movie_reviews/0_2.txt'
INTO TABLE reviews;
```

Dataset 6 - Riassunti di casi giudiziari Come prima si crea la tabella che conterrà i dati e si popola con l'apposito comando mySQL "LOAD XML".

```
CREATE TABLE cases
(
    id varchar(10),
    testo text
);
```

```
LOAD XML LOCAL INFILE '/home/omar/Scrivania/Big Data/Secondo Progetto/Datasets/
```

```
case_summary/06_1.xml'  
INTO TABLE cases;  
ROWS IDENTIFIED BY '<sentences>';
```

In generale ha tempi simili a PostgreSQL dato che gli import vengono eseguiti da locale.

4.4.2 Singlestore

Singlestore è un database SQL relazionale e distribuito, usato soprattutto per la sua velocità a livello di dati, transazioni e query.

Permette lo storage di dati relazionali ma anche JSON, timeseries e grafi - inoltre tramite vari connettori permette anche l'uso di altri formati come ad esempio XML. A livello di cloud si appoggia su AWS, Azure o Google Cloud Platform, mentre per i dati sorgenti essi possono essere passati da un computer locale, da cloud o da pipeline ETL.

Ha un'interfaccia grafica intuitiva e di facile utilizzo. Nel nostro caso abbiamo usato delle macchine con le seguenti specifiche:

Sperimentazione

Dataset 1 - Cambio Euro/Dollaro Caricamenti ed esecuzioni di query quasi istantanee. L'elevata potenza di calcolo si nota facilmente. I dati possono essere importati da locale, da cloud(GCP, AWS, Azure), tramite pipeline ETL, file SQL ed altri.

Dataset 2 - Nomi bambini francesi Anche qui, prestazioni elevate. File TSV supportato.

Dataset 3 - Crimini commessi ad Atlanta Stesse considerazioni per il dataset precedente. Il supporto di SQL ha facilitato molto l'esecuzione delle varie query.

Dataset 4 - Album di foto stock Sono state caricate le foto su Azure e poi salvati i riferimenti. Prestazioni sempre altissime.

Dataset 5 - Recensioni IMDB Anche in questo caso, ottime performance.

Dataset 6 - Riassunti di casi giudiziari I file XML possono essere caricati semplicemente come testo, oppure tramite connettori appositi possono essere salvati con la struttura predefinita. Anche qui i tempi di caricamenti sono stati ideali.

4.5 Risultati Query

Per i tre dataset strutturati, abbiamo effettuato delle query, talvolta stressanti, per poterne misurare le prestazioni, come tempo impiegato.

Generalmente, i NewSQL si sono comportati benissimo in ogni caso, i NoSQL hanno dato ottimi risultati ma inferiori ai NewSQL. I database SQL hanno dato buone prestazioni per dimensioni piccole dei dataset ma al crescere dei dati, le prestazioni calano notevolmente.

Di seguito un esempio di query per singolo dataset e i risultati.

Dai tempi si può notare come gli SQL abbiano nella scalabilità il loro punto debole. I NoSQL si sono comportati molto bene, come i NewSQL che però si appoggiano a strutture cloud e avevano una potenza di calcolo superiore.

```
LOAD DATA LOCAL INFILE 'C:\\Users\\Jomin\\Desktop\\atlanta_crimes.csv'  
REPLACE INTO TABLE `mysql`.`ac`;
```

```
CREATE VIEW cnd AS  
SELECT ac.crime, ac.neighborhood, ac.date FROM ac;
```

```
CREATE VIEW number_crime_same_day as  
SELECT COUNT(distinct cnd.crime) AS same_day,  
GROUP_CONCAT(cnd.crime), cnd.neighborhood, cnd.date  
FROM cnd  
GROUP BY cnd.date, cnd.neighborhood  
ORDER BY same_day DESC;
```

```
CREATE VIEW neigh_crimes as  
SELECT count(ac.crime) as count_crime, ac.neighborhood  
FROM ac  
GROUP BY ac.neighborhood;
```

```
SELECT *  
FROM neigh_crimes a, number_crime_same_day b
```

```

WHERE a.neighborhood = b.neighborhood
AND a.neighborhood <> ' '
GROUP BY a.neighborhood
ORDER BY a.count_crime DESC, b.same_day DESC;

```

Database	Dataset 1(x4, x8)	Dataset 2(x4, x8)	Dataset 3(x4, x8)
MariaDB	0,13 s - 0,57 s - 0,77 s	10,3 s - 26,2 s - 64 s	13 s - 29 s - 69 s
PostgreSQL	0,632 s - 0,78 s - 1,39 s	3 s - 7,5 s - 14,9 s	6 s - 18 s - 30 s
Neo4j	0,041 s - 0,115 s - 0,2 s	22 s - 23 s - 24,5 s	15 s - 20 s - 21 s
MongoDB	0,74 s - 2,5 s - 4,5 s	5,7 s - 27,2 s - 65,2 s	5,1 s - 25,8 s - 50,1 s
Singlestore	0,13 s - 0,125 sec - 0,18 s	0,7 s - 2 s - 2,4 s	2 s - 3,1 s - 3,5 s
TiDB	2,1 s - 2,69 s - 3,52 s	18,2 s - 27,6 s - 59,3 s	26,9 s - 71,4 s - 155 s

- Tempi d'esecuzione delle query in secondi, per i 3 dataset strutturati, per ogni database, per file originale, x4 e x8.

Select File

/home/omar/Scrivania/06_1.csv

BROWSE

Select Input File Type

JSON

CSV

Options

Select delimiter

COMMA

☒ Ignore empty strings

☐ Stop on errors

Specify Fields and Types

	<input checked="" type="checkbox"/> "id" <div>String</div>	<input checked="" type="checkbox"/> #Text <div>String</div>
1	s0	\n Background to the current application \n \n1 The applicants s
2	s1	When referring to the applicants generally, I will do so as 'the
3	s2	Each of the Sharman applicants was one of ten respondents to inf
4	s3	Wilcox J made orders ancillary to the Mareva orders on 22 March
5	s4	2 Wilcox J delivered judgment on the complex issues of liability
6	s5	In the meantime, Ms Hemming had filed two disclosure affidavits
7	s6	Disclosure affidavits were eventually sworn on behalf of Sharmar
8	s7	Sharman License and Sharman Networks had also unsuccessfully sou
9	s8	3 On 24 May 2005, the Music companies filed a further amended nc
10	s9	The Sharman applicants acknowledge that the orders made by Moore

CANCEL

IMPORT

Figure 7: Conversione degli XML.


```
_id: ObjectId("60dedfaeae387d25ac1d53c5")  
files_id: ObjectId("60dedfaeae387d25ac1d53c4")  
n: 0  
data: Binary( '/9j/4AAQSkZJRgABAQEBALEsAAD/2wBDAAMC
```

```
_id: ObjectId("60dedfb6ae387d25baf1f0bd")  
files_id: ObjectId("60dedfb6ae387d25baf1f0bc")  
n: 0  
data: Binary( '/9j/4AAQSkZJRgABAQEASABIAAD/4gxYSUNC
```

```
_id: ObjectId("60dedfbaeae387d25c9480bfb")  
files_id: ObjectId("60dedfbaeae387d25c9480bfa")  
n: 0  
data: Binary( '/9j/4AAQSkZJRgABAQEASABIAAD//gAMQXBw
```

```
_id: ObjectId("60dedfaeae387d25ac1d53c4")  
chunkSize: 261120  
uploadDate: 2021-07-02T09:43:10.841+00:00  
length: 155667  
md5: "03f275eb54a74fbe39263aa0adafdd0a"  
filename: "0000000000632.jpg"
```

```
_id: ObjectId("60dedfb6ae387d25baf1f0bc")  
chunkSize: 261120  
uploadDate: 2021-07-02T09:43:18.485+00:00  
length: 176410  
md5: "cc92903689360505929b2e2af7972651"  
filename: "0000000000776.jpg"
```

```
_id: ObjectId("60dedfbaeae387d25c9480bfa")  
chunkSize: 261120  
uploadDate: 2021-07-02T09:43:26.095+00:00  
length: 133674  
md5: "499f3a154b1c7d7019e15db5373c7a48"  
filename: "0000000000785.jpg"
```

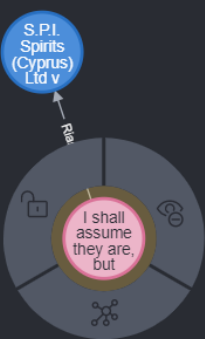
Figure 8: Caricamento immagini.

```
neo4j$ LOAD CSV FROM 'file:///reviews.csv' AS row RETURN row;
```

	row
1	["I admit", " the great majority of films released before say 1933 are just not for me. Of the dozen or so major silents I have viewed", " one I loved (The Crowd)", " and two were very good (The Last Command and City Lights", " that latter Chaplin circa 1931). So I was apprehensive about this one", " and humor is often difficult to appreciate (uh", " enjoy) decades later. I did like the lead actors", " but thought little of the film. One intriguing sequence. Early on", " the guys are supposed to get de-loused and for about three minutes", " fully dressed", " do some schtick. In the background", " perhaps three dozen men pass by", " all naked", " white and black (WWI ?)", " and for most", " their butts", " part or full backside", " are shown. Was this an early variation of beefcake courtesy of Howard Hughes?;"]
2	["I went and saw this movie last night after being coaxed to by a few friends of mine. Ill admit that I was reluctant to see it because from what I knew of Ashton Kutcher he was only able to do comedy. I was wrong. Kutcher played the character of Jake Fischer very well", " and Kevin Costner played Ben Randall with such professionalism. The sign of a good movie is that it can toy with our emotions. This one did exactly that. The entire theater (which was sold out) was overcome by laughter during the first half of the movie", " and were moved to tears during the second half. While exiting the theater I not only saw many women in tears", " but many full grown men as well", " trying desperately not to let anyone see them crying. This movie was great", " and I suggest that you go see it before you judge.;"]
3	["Once again Mr. Costner has dragged out a movie for far longer than necessary. Aside from the terrific sea rescue sequences", " of which there are very few I just did not care about any of the characters. Most of us have ghosts in the closet", " and Costners character are realized early on", " and then forgotten until much later", " by which time I did not care. The character we should really care about is a very cocky", " overconfident Ashton Kutcher. The problem is he comes off as kid who thinks hes better than anyone else around him and shows no signs of a cluttered closet. His only obstacle appears to be winning over Costner. Finally when we are well past the half way point of this stinker", " Costner tells us all about Kutchers ghosts. We are told why Kutcher is driven to be the best with no prior inking or foreshadowing. No magic here", " it was all I could do to keep from turning it off an hour in.;"]
4	["Story of a man who has unnatural feelings for a pig. Starts out with a opening scene that is a terrific example of absurd comedy. A formal orchestra audience is turned into an insane", " violent mob by the crazy chantings of its singers. Unfortunately it stays absurd the WHOLE time with no general narrative eventually making it just too off putting. Even those from the era should be turned off. The cryptic dialogue would make Shakespeare seem easy to a third grader. On a technical level its better than you might think with some good cinematography by future great Vilmos Zsigmond. Future stars Sally Kirkland and Frederic Forrest can be seen briefly.;"]
5	

Figure 9: Recensioni nel database.

```
neo4j$ Match (n)-[r]→(m) Return n,r,m limit 1
```



```
s1 <id>: 3 name: S.P.I. Spirits (Cyprus) Ltd v Diageo Australia Ltd [2006] FCA 14 sentence1: It is not clear whether these orders are sought in the alternative. sentence2: I shall assume they are, but make it clear that if I do not think it appropriate to make an order in terms of that sought pursuant to O 29 r 2(a), I would not b sentence3: 2 The applicants S.P.I. sentence4: Spirits (Cyprus) Limited and Spirits International N.V. (respectively "SPI Spirits" and "Spirits International") oppose
```

Figure 10: Esempio relationship caso-riassunto.

```
mysql> SELECT a.name name_a, b.name name_b, a.occorrenze occorrenze_a, b.occorrenze occorrenze_b
-> FROM nomi_occorrenze a, nomi_occorrenze b
-> WHERE a.name LIKE CONCAT('%', b.name, '%')
-> AND b.occorrenze > 20000
-> AND a.occorrenze > 20000
-> AND a.name <> b.name;
```

name_a	name_b	occorrenze_a	occorrenze_b
MARIE-NOËLLE	NOËL	21225	51758
MARIE-NOËLLE	MARIE	21225	2261978
MARIE-NOËLLE	NOËLLE	21225	28216
JONATHAN	NATHAN	104895	106745
JOSEPHINE	JOSEPH	90170	374753
LUCETTE	LUC	69951	65021
PASCALÉ	PASCAL	102649	307980
ANGÉLIQUE	ANGÉ	76589	20166
LILIANE	LILIAN	112069	25987
ÉMILIANNE	ÉMILIE	45388	192098
ÉMILIANNE	ÉMILIE	45388	23802
MARIE-JOSÉ	MARIE	34756	2261978
MARIE-JOSÉ	JOSÉ	34756	58733
NOËLLE	NOËL	28216	51758
JEAN-MARC	JEAN	90329	1919282
JEAN-MARC	MARC	90329	237339
YVONNE	YVON	257076	40554
ROLANDE	ROLAND	39953	132441
JEAN-CHRISTOPHE	JEAN	31884	1919282
JEAN-CHRISTOPHE	CHRISTOPHE	31884	376050
MARCEL	MARC	467535	237339
JEAN-CLAUDE	JEAN	172541	1919282
JEAN-CLAUDE	AUDE	172541	31457
JEAN-CLAUDE	CLAUDE	172541	468462
ANDRÉA	ANDRÉ	48592	712594
SUZANNE	ANNE	288539	365279
MARIE-THÉRÈSE	MARIE	88574	2261978
MARIE-THÉRÈSE	THÉRÈSE	88574	175916
MARIE-CLAUDE	MARIE	57345	2261978
MARIE-CLAUDE	AUDE	57345	31457
MARIE-CLAUDE	CLAUDE	57345	468462
LÉONIE	LÉON	56337	117972
LÉONIE	LÉO	56337	93526
CORALIE	ALI	55155	25293
MARTINE	MARTIN	320278	60085
PAULETTE	PAUL	213450	413525
PAULETTE	PAULE	213450	46300
MAGALI	ALI	74133	25293
ANDRÉE	ANDRÉ	215643	712594
VALENTINE	VALENTIN	46544	106677
EMMANUEL	MANUEL	135255	36385
EMMANUEL	EMMA	135255	150593
FRANÇOISE	FRANÇOIS	401561	398669
WILLIAM	LIAM	74774	27531
ANNE-MARIE	MARIE	99860	2261978
ANNE-MARIE	ANNE	99860	365279

Figure 11: Risultato.

	Jomin Neelikatt's Cluster PRODUCTION	S-1 8 vCPUs 64 GB Memory 1 TB Storage	6/19/21 at 2:06:38 PM
--	---	---	-----------------------

Figure 12: Specifiche cluster Singlestore.



TiDBCluster 	
Status	Normal 
TiDB Version	v4.0.11
Region	us-west-2
Cluster Tier	T3.standard
TiDB Nodes	2 8 vCPU
TiKV Nodes	3 8 vCPU 1024 GiB
Total Capacity	3072 GiB
Creator	oma.mohd@stud.uniroma3.it

Figure 13: Specifiche cluster TiDB.

5 Considerazioni sui Database

5.1 MariaDB

Il prodotto è validissimo, anche grazie ad un'interfaccia utente semplice, intuitiva e completa; stesso discorso vale per gestioni e configurazioni.

Come prestazioni, si conferma un buon prodotto, è ovvio che per grandi moli di dati, i tempi inizino a salire - gli SQL sono nati in un periodo in cui il concetto di Big Data era ancora impensabile.

Sono moltissimi le tipologie di dati supportate - anche a grazie a tantissimi connettori disponibili. Essendo un SQL non c'è la necessità di imparare nuovi linguaggi(l'SQL è universalmente conosciuto sia in ambito lavorativo che accademico). Essendo open-source ha una vastissima community(in parte anche eredità dal 'padre' MySQL) e di conseguenza, ogni problema che abbiamo riscontrato è stato facilmente risolto tramite qualche ricerca online.

5.2 PostgreSQL

PostgreSQL fa quello che ci si aspetta da uno dei database relazionali SQL più utilizzati da molti anni ed in tutto il mondo. Non per niente è uno dei pochi db insegnati in ambito accademico. Come per MariaDB essendo un SQL non c'è stata la necessità di imparare nuovi linguaggi(l'SQL è universalmente conosciuto sia in ambito lavorativo che accademico).

L'interfaccia che offre pgadmin è intuitiva ed efficace, permette di effettuare molte operazioni scrivendo poco codice. Il query tool fa il suo dovere egregiamente permettendo anche la visualizzazione in tempo reale dei risultati intermedi e finali e dei tempi di esecuzione. Qualche problema è stato riscontrato nell'inserimento multiplo di file diversi dai soliti csv, tsv. Problemi risolti comunque con manipolazione dei dati.

5.3 Neo4j

Il prodotto è suggerito maggiormente a chi deve usare dati che possono essere strutturati come un grafo. Ma anche nei nostri esperimenti su altri tipi di dati, si è comportato egregiamente.

La gestione dei dati è agevole, come la creazione di nodi e relazioni.

Possiede molti connettori e plugins per poter integrare moltissimi tipi di dati. Essendo un NoSQL è molto efficiente anche su grandi moli di dati, risolvendo il problema di scalabilità implicito negli SQL.

5.4 MongoDB

MongoDB rappresenta una valida alternativa a classici strumenti SQL. Presenta lo scoglio iniziale di capire e utilizzare un nuovo linguaggio(MQL) con tutti i suoi operatori e comandi, e di capire quando e in che modo utilizzarli e combinarli efficientemente. Una volta superato però si inizia a conoscere un database veramente molto valido che offre strumenti per risolvere ogni problema in questo tipo di contesto. Saltano subito all'occhio la sua velocità di esecuzione(utilizza l'indicizzazione) e la facilità di import e gestione di diverse tipologie di dati. Anche MongoCompass da una mano notevole sotto questo punto di vista.

5.5 Singlestore

Singlestore è un prodotto incredibile. Prestazioni elevate, solidità, intuitività nell'interfaccia, compatibilità con tantissimi ambienti sono solo alcune delle sue svariate potenzialità. Sono presenti varie tipologie di macchine in base alle esigenze(noι ne abbiamo scelta una altamente performante). Supporta il linguaggio SQL ed ha un'assistenza da parte del team molto attenta (appena l'abbiamo installato, ci hanno contattato dal team di sviluppo per chiederci informazioni e se avessimo necessità d'aiuto).

5.6 TiDB

Infine TiDB rappresenta forse proprio lo spirito di un database NewSQL, unire la facilità e la popolarità del linguaggio SQL con l'efficienza e la velocità di un database NoSQL. Grazie anche e soprattutto all'utilizzo di cluster su cloud o in locale. I tempi, a prima vista non eccellenti, sono dovuti all'import dei file dal filesystem locale. Infatti considerando solo esecuzione di query risulta essere uno dei migliori db testati. Come Singlestore ha un'assistenza da parte del team di sviluppo molto attenta.

6 Considerazioni Finali

Dopo aver letto la parte teorica delle tre filosofie(SQL, NoSQL, NewSQL) sembrava abbastanza evidente che i database NewSQL fossero decisamente migliori per prestazioni e scalabilità - proprietà che nell'era dei Big Data sono dei must. Anche visionando i tempi delle query nella tabella precedente, l'esito sembrava abbastanza scontato, invece secondo noi **non esiste una soluzione generalmente migliore**. E' sì vero che, ad esempio Singlestore ha avuto prestazioni nettamente migliori, ma c'è anche da dire che Maria DB e PostgreSQL sono stati eseguiti in locale mentre Singlestore in ambiente cloud con 1 terabyte di memoria, 8 CPU e 64G di RAM, con un costo di quasi 3 dollari all'ora.

Un altro dettaglio è che i NoSQL non hanno linguaggio standard anche se ci sono proposte([UnQL](#)), ciò implica una curva d'apprendimento iniziale più alta rispetto all'SQL che viene spesso insegnato anche durante gli studi.

La scelta su una tipologia di database **deve dipendere** dalle proprie esigenze di progetto. Dovrebbero essere considerati tanti aspetti, come ad esempio quali, quanti e che gestione dei dati è necessaria(OLTP, IoT, analisi temporali, BI, esigenze particolari), necessità d'uso SQL/proprietà ACID contrapposta a gestione di dati non strutturati(senza dimenticare consistenza, persistenza e disponibilità), architettura (single server, cluster, sharding, replicazione). Quindi, dopo le varie considerazioni si possono considerare i vari database - in questo senso, il mercato offre una vastissima scelta.

La scelta non risulta banale ma ed essenziale per un progetto proprio per non incorrere in problemi futuri a progetto iniziato.

In conclusione, ci riteniamo soddisfatti delle riflessioni maturate e degli obiettivi raggiunti, il progetto ha dato i suoi frutti sia in una consapevolezza maggiore sulle potenzialità di tipologia di database e singoli database ma anche per quanto riguarda una conoscenza generale in ambito big data.