

UNIVERSITÀ DEGLI STUDI DI MESSINA

Dipartimento di Scienze Matematiche e Informatiche Scienze
Fisiche e Scienze della Terra

Corso di Laurea in Informatica

RELAZIONE - BASI DI DATI II

**“ANALISI DELLA RETE
POLITICA DI DONALD TRUMP”**

Sonia Trifiletti
Federica Zuccarello

Prof. Antonio Celesti

Anno Accademico 2018-2019

Indice

Indice	2
Problematica affrontata	4
Introduzione	4
I DBMS NoSQL	4
Soluzioni DB utilizzate	5
Neo4j	5
MongoDB	5
Strumenti Utilizzati	6
Progettazione	7
Data Model	7
Neo4j: Modello di dati grafico	8
Tecnica di Modellazione di un Grafo	9
MongoDB: Modello di dati documentale	9
Tecnica di Modellazione di un Documento	10
Implementazione	12
Importazione in Neo4j	12
Importazione in MongoDB	13
Implementazione Query	14
Query 1	14
Query 2	15
Query 3	15
Query 4	16
Query 5	16
Query 6	17
Query 7	17
Generatore Dati	18

Esperimenti	19
Risultati Neo4j - Dataset di 1000 e 10000	19
Confronto dei risultati	20
Risultati MongoDB - Dataset di 1000 e 10000	22
Confronto Dataset da 1000 per i due DB	22

Problematica affrontata

Introduzione

Il 15 gennaio BuzzFeed ha rilasciato un ampio set di dati sulle connessioni di Donald Trump, incluse persone, organizzazioni e la natura delle loro relazioni. In questo progetto mostriamo la visualizzazione della rete politica di Donald Trump basata su quel set di dati. In questo tipo di contesto la soluzione migliore per l'analisi dei dati è sicuramente l'utilizzo di un DB NoSql in cui l'informazione più preziosa risiede nelle relazioni tra gli oggetti piuttosto che nei dati stessi, Neo4j (il più famoso Graph Database) esemplifica perfettamente questo modello di offerta. Per dimostrare la totale bontà del metodo scelto, si è deciso di implementare la stessa problematica utilizzando un altro DB Nosql, MongoDB (Document Database), che segue una filosofia opposta, cioè debolmente orientata alle relazioni.

I DBMS NoSQL

Negli ultimi anni, con l'avvento dei *Big Data* è incredibilmente aumentata la popolarità delle tecnologie di immagazzinamento di informazioni conosciute con il nome di **NoSQL** acronimo che sta per *Not only SQL*.

I NoSQL Database Management System sono sistemi software che consentono di immagazzinare e organizzare i dati senza fare affidamento sul modello relazionale. Ne esistono tantissimi e possono avere le caratteristiche più disparate, vediamo quali sono i più importanti:

- Key-Value store.
- Document-oriented.
- Column store.
- Graph DBMS.

Soluzioni DB utilizzate

Neo4j

Neo4j è un famoso database grafico, cioè un database utilizzato per modellare i dati sotto forma di grafo, qui i nodi di un grafo rappresentano le entità mentre le relazioni descrivono l'associazione di questi nodi.

È DBMS open source transazionale, prodotto dalla software house Neo Technology, sviluppato completamente in Java, robusto, scalabile e ad alte prestazioni. È dotato di:

- transazioni ACIDe,
- alta disponibilità,
- tecniche di memorizzazione per miliardi di nodi e relazioni,
- alta velocità di interrogazione tramite attraversamenti,
- linguaggio di interrogazione dichiarativo e grafico (**Cypher**).

È un DBMS schema-less, ciò sta a significare che i suoi dati non devono attenersi ad alcuna struttura di riferimento prefissata, inoltre non possiede una politica di accesso controllata.

MongoDB

MongoDB è un database NoSQL orientato ai documenti, che nasce nel 2007 in California come servizio da utilizzare nell'ambito di un progetto più ampio, ma presto è diventato un prodotto indipendente ed open-source. Esso memorizza i documenti in JSON, più precisamente in formato **BSON** (concettualmente simile a un oggetto JSON ma codificato in forma binaria). Il documento è fondamentalmente un albero che può contenere molti dati, anche annidati. I documenti sono raggruppati in **collezioni** che possono essere anche eterogenee, non esiste quindi uno schema fisso per i documenti, tra le collezioni non ci sono relazioni o legami garantiti.

Le caratteristiche chiave di MongoDB sono:

- alta disponibilità, dal momento che la replicazione di un database può avvenire in modo molto semplice,
- scalabilità orizzontale, ossia la possibilità di distribuire le collezioni in cluster di nodi, in modo da supportare grandi quantità di dati senza influire pesantemente sulle performance.
- alta velocità, a scapito delle proprietà ACIDe.

Strumenti Utilizzati

- *Sublime Text*: editor di testo
 - *Eclipse Java*: ambiente di sviluppo per programmi Java
 - *XAMPP*: una piattaforma software open source costituita dal server web Apache
 - *Mozilla Firefox*: browser usato per sviluppo e test
- inserire tutti gli strumenti....

Progettazione

Data Model

Nell'ambito delle basi di dati, il modello dei dati rappresenta un insieme di strumenti concettuali, detto “formalismo”, che consta di tre componenti essenziali:

- insieme di strutture dati con opportuni operatori
- notazione per specificare i dati tramite le strutture dati del modello
- insieme di operazioni per manipolare i dati.

Qualsiasi modello dei dati deve risolvere due principali quesiti:

- come rappresentare le entità e i loro attributi
- come rappresentare le associazioni.

Per descrivere bene questi aspetti sopra citati è stato costruito un modello concettuale, in particolare **E-R**, per rappresentare la gestione degli oggetti più importanti e le loro associazioni.(fig. 1)

Chiaramente lavorando con Database Nosql tale grafico avrà il solo obiettivo di mostrare in maniera generale il comportamento dei dati ma non risulterà totalmente vincolante ai fini della stesura del modello logico di ciascun DB utilizzato.

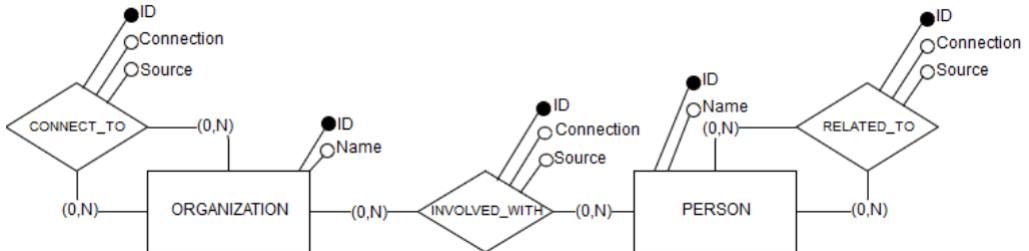


Figura 1: Modello Concettuale, E-R

Da un'attenta analisi del Dataset messo a disposizione dalla traccia ci si è resi conto che le entità coinvolte potevano essere suddivise direttamente in base al loro tipo perché presenti in solo due forme: *ORGANIZATION* e *PERSON*, per tale ragione questi due oggetti diventano le entità fondamentali dell'E-R aventi due attributi: **ID** (identificatore) e Name. Esistono tre tipi di associazioni descritte in base alle entità coinvolte:

- a) *CONNECT_TO*, descrive relazioni tra due organization (associazione ricorsiva molti a molti)
 - b) *INVOLVED_WITH*, descrive relazioni tra un organization e un person (associazione binaria molti a molti)
 - c) *RELATED_TO*, descrive relazioni tra due person (associazione ricorsiva molti a molti)

Tutte le relazioni hanno gli stessi attributi: **ID** (identificatore), Connection (natura delle loro relazioni), Source (Sito web con esemplificazione della relazione).

Neo4j: Modello di dati grafico

Neo4j Graph Database memorizza tutti i suoi dati sotto forma di grafo, perciò è naturale chiedersi cosa sia un grafo.

Un grafo è un insieme di nodi connessi da relazioni. I grafi rappresentano le entità con i nodi, e il modo nel quale queste entità si rapportano con il mondo, con le relazioni. I dati del nodo e delle relazioni sono rappresentate in termini di proprietà (coppie chiave-valore).

Dunque i principali elementi costitutivi di un modello di dati grafico sono:

- I nodi

- Le relazioni
- Le Proprietà

Tecnica di Modellazione di un Grafo

Partendo dal nostro schema E-R, costruito precedentemente, effettuiamo una vera e propria traduzione ad un modello a grafo che cerchi di descrivere al meglio il dominio del problema.

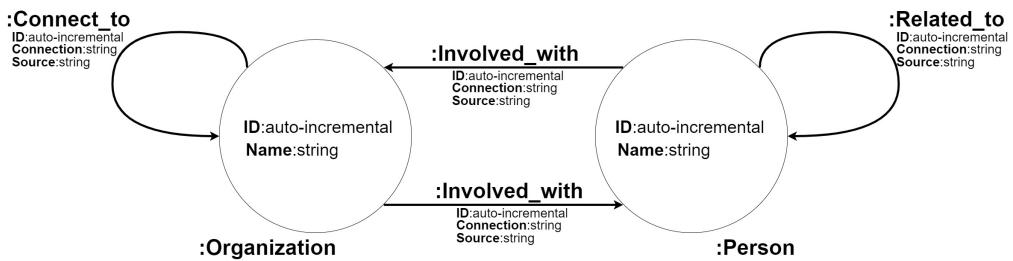


Figura 2: Modello Grafico generalizzato

Osservando questo modello si intuisce bene la natura del database. I cerchi rappresentano i nodi che, ovviamente in questo contesto, sono i due oggetti: *Organization* e *Person* e le frecce rappresentano le relazioni cioè: *Connect_to*, *Involved_with*, *Related_to*, le relazioni hanno direzioni unidirezionale e bidirezionale. Entrambi i nodi e le relazioni contengono proprietà: *ID* che è un valore incrementale univoco e *Name*, *Connection*, *Source* che sono valori di tipo stringa.

MongoDB: Modello di dati documentale

In MongoDB l'unità fondamentale di memorizzazione delle informazioni è rappresentata dal documento. Tale termine non ha una definizione precisa e univoca, in quanto un documento è un qualsiasi insieme di informazioni concettualmente coerenti raggruppate secondo una data codifica.

I documenti vengono raccolti in collezioni (*collection*) le quali rappresentano una suddivisione concettuale dei dati in modo analogo alle tabelle dei database relazionali; a differenza di quest'ultime, in MongoDB una collezione non ha vincoli strutturali e pertanto i documenti presenti all'interno possono avere campi diversi, per tipo e valore. A loro volta, le collection risiedono nel database che rappresenta l'universo del sistema di cui si vogliono gestire i dati.

Modello Relazione	Modello documentale
DataBase	Database
Tabella	Collezione
Riga	Documento
Colonna	Campo
Chiave Primaria	ObjectId fornito da mongodb stesso

Figura 3: Confronto della terminologia RDBMS con MongoDB

Tecnica di Modellazione di un Documento

MongoDB non permette alcune operazioni fondamentali tipiche dei database relazionali, quali join tra collection. La mancanza di controlli di integrità referenziale tra i dati obbliga il progettista a modellare lo schema logico con modalità molto diverse da quelle messe in atto durante lo sviluppo di soluzioni SQL-based.

Dataset:DataTrump

```
Collection:Dataset
  Document
    _id : ObjectId
    name : String
    type : String
    Connections : Array
      0 : Object
        id_con : String
        name_con : String
        type_con : String
        type_con : String
```

Figura 4: Modello documentale generalizzato

Il documento in esame permette di mostrare la soluzione implementativa utilizzata per gestire la mancanza di relazioni. Innanzitutto all'interno di ogni documento è presente:

- a) il campo speciale **ObjectID** (“_id”) per l'identificazione univoca del documento
- b) il campo “name” contenente il nome dell'entità

- c) il campo “*type*” che può assumere il valore di Organization o Person
- d) il campo “*Connections*” di tipo array contenente documenti annidati per descrivere le associazioni.
 - d.1) il sotto-campo ‘*id_con*’ contenente gli idObject delle entità collegate
 - d.2) il sotto-campo ‘*name_con*’ contenente i nomi delle entità collegate
 - d.3) il sotto-campo ‘*type_con*’ contenente il tipo di legame
 - d.4) il sotto-campo ‘*source_con*’ contenente i siti web delle entità collegate

Questo approccio mantiene tutti i dati relativi in ogni singolo documento, il che rende più facile il recupero e la manutenzione. Lo svantaggio è che se il documento incorporato continua a crescere troppo di dimensioni, può influire sulle prestazioni di lettura / scrittura.

Implementazione

Completato lo step progettuale si passa alla fase implementativa. Il primo aspetto fondamentale è effettuare una corretta importazione dei dati per entrambi i DB.

Importazione in Neo4j

Come già detto nell'introduzione, Neo4j supporta come linguaggio dichiarativo ufficiale **Cypher**, creato dal team di Neo4j allo scopo di interagire con il database in modo molto semplice, è possibile infatti scrivere le interrogazioni direttamente da riga di comando dell'interfaccia di Neo4j. Di seguito viene proposto il codice utilizzato per l'importazione dei dati dal file CSV a disposizione.

```
1 LOAD CSV WITH HEADERS FROM "file:///TrumpWorldData.csv" AS row
2 WITH row WHERE row.`Entity A Type` = 'Organization' AND row.`Entity B Type` = 'Organization'
3 MERGE (o1:Organization {name:row.`Entity A`})
4 MERGE (o2:Organization {name:row.`Entity B`})
5 CREATE (o1)-[con:CONNECTED_TO]->(o2)
6 SET con.connection=row.Connection, con.source=row.`Source(s)`
```

Figura 5: Importazione delle organizzazioni e le loro connessioni

```
1 LOAD CSV WITH HEADERS FROM "file:///TrumpWorldData.csv" AS row
2 WITH row WHERE row.`Entity A Type` = 'Person' AND row.`Entity B Type` = 'Organization'
3 MERGE (p:Person {name:row.`Entity A`})
4 MERGE (o:Organization {name:row.`Entity B`})
5 CREATE (p)-[con:INVOLVED_WITH]->(o)
6 SET con.connection=row.Connection, con.source=row.`Source(s)`
7
8 LOAD CSV WITH HEADERS FROM "file:///TrumpWorldData.csv" AS row
9 WITH row WHERE row.`Entity A Type` = 'Organization' AND row.`Entity B Type` = 'Person'
10 MERGE (o:Organization {name:row.`Entity A`})
11 MERGE (p:Person {name:row.`Entity B`})
12 CREATE (p)-[con:INVOLVED_WITH]->(o)
13 SET con.connection=row.Connection, con.source=row.`Source(s)`
```

Figura 6: Importazione delle organizzazioni e delle persone connesse

```

1 LOAD CSV WITH HEADERS FROM "file:///TrumpWorldData.csv" AS row
2 WITH row WHERE row.`Entity A Type` = 'Person' AND row.`Entity B Type` = 'Person'
3 MERGE (p1:Person {name:row.`Entity A`})
4 MERGE (p2:Person {name:row.`Entity B`})
5 CREATE (p2)-[con:RELATED_TO]-(p1)
6 SET con.connection=row.Connection, con.source=row.`Source(s)`

```

Figura 7: Importazione delle organizzazioni e delle persone connesse

Importazione in MongoDB

Sebbene MongoDB disponga di una propria Shell per l’invio di interrogazioni e comandi e per restituire i risultati, per i più importanti e utilizzati linguaggi di programmazione sono state realizzate delle librerie (**Driver**, come si dice nell’ambito della gestione di database) per l’interfacciamento con MongoDB in modo semplice e intuitivo. Per gli script di importazione è stato scelto l’*API PHP* mentre per la scrittura delle query l’*API Python*. Per cercare di ottimizzare al meglio i dati sono stati creati 4 script relativi ad ogni step:

1. “*manipulation_dataset.php*”, dal file CSV originale vengono creati 3 nuovi sotto-dataset (“*CONNECT_TO.php*”, “*INVOLVED_WITH.php*”, “*RELATED_TO.php*”) secondo il modello descritto nello schema concettuale.
2. “*import_data.php*”, permette di inserire i dati dei 3 dataset nella collection di MongoDB, per ogni documento viene generatore il proprio ObjectID.

```

56 //Inserisce il nome di ORGANIZATION
57     for ($i=0; $i<$conto1; $i++){
58         $doc1 = [
59             '_id' => new MongoDB\BSON\ObjectId(),
60             'name' => $output1[$i],
61             'type' => "Organization",
62         ];
63         $bulk->insert($doc1);
64     }
65 //Inserisce il nome di PERSON
66     for ($i=0; $i<$conto2; $i++){
67         $doc2 = [
68             '_id' => new MongoDB\BSON\ObjectId(),
69             'name' => $output2[$i],
70             'type' => "Person",
71         ];
72         $bulk->insert($doc2);
73     }

```

3. “*recuperoIDMONGO.php*”, recupera gli ObjectID da i documenti di MongoDB generati e li salva in un file CSV (*recuperaID*) con i nomi associati.

4. “*updateIDMONGO.php*”, permette di individuare le connessioni tra le entità a causa della mancanza del costrutto join in MongoDB.
5. *appendConnection.php*, per inserire nel campo Connections tutte le entità che sono associate

```

56 //Inserisce il nome di ORGANIZATION
57     for ($i=0; $i<$conto1; $i++){
58         $doc1 = [
59             '_id' => new MongoDB\BSON\ObjectId(),
60             'name' => $output1[$i],
61             'type' => "Organization",
62         ];
63         $bulk->insert($doc1);
64     }
65 //Inserisce il nome di PERSON
66     for ($i=0; $i<$conto2; $i++){
67         $doc2 = [
68             '_id' => new MongoDB\BSON\ObjectId(),
69             'name' => $output2[$i],
70             'type' => "Person",
71         ];
72         $bulk->insert($doc2);
73     }

```

Implementazione Query

Una volta inseriti i dati vengono eseguite per entrambi i DB 7 query. Per ogni interrogazione viene riportato l'intero codice scritto in **Cypher** (immagine su) e in **MongoDBShell** (immagine giù).

Query 1

Visualizzazione del nome di 10 nodi random.

```
MATCH (n) RETURN n.name LIMIT 10"
```

```
DataTrump.DataSet.find().limit(10)
```

Query 2

Ricerca le Organizzazioni che hanno interagito maggiormente e restituisce in ordine decrescente il conto totale delle connessioni stabilite con i relativi collegamenti:

```
MATCH (a:Organization)-[r]-()
RETURN a.name, COUNT(*) as b ORDER BY COUNT(*) DESC
```

```
DataTrump.DataSet.aggregate([
  {'$match': {'type': 'Organization'}},
  {'$unwind': '$Connections'},
  {'$group': {'_id': '$Connections.name_con', 'totali': {'$sum': 1}}},
  {'$sort': {'totali': -1}}
])
```

Query 3

Restituisce il numero di volte che una connessione si presenta nel dataset

```
"MATCH (a)-[r]-(b) RETURN r.connection AS connection,
count(*) as b ORDER BY count(*) DESC"
```

```
DataTrump.DataSet.aggregate([
  {'$unwind': '$Connections'},
  {'$group': {'_id': '$Connections.type_con',
    'totali': {'$sum': 1}}},
  {'$sort': {'totali': -1}}
])
```

Query 4

Rispetto alle organizzazioni Trump Tower Commercial LLC e 40 Wall Street LLC che sono quelle più popolari, quali organizzazioni sono comuni ad entrambe

```
"MATCH (o1:Organization {name:'TRUMP TOWER COMMERCIAL LLC'})  
MATCH (o2:Organization {name:'40 WALL STREET LLC'})  
MATCH path = (o1)-[*..3]-(o2) RETURN path"
```

```
DataTrump.DataSet.find({'$and':  
  [{Connections.name_con:'TRUMP TOWER COMMERCIAL LLC'},  
   {'Connections.name_con':'40 WALL STREET LLC'}]  
})
```

Query 5

Chi ha nominato Trump come capo di Gabinetto

```
"MATCH (p:Person)-[con:RELATED_TO]->()  
WHERE con.connection CONTAINS 'Labor Secretary'  
RETURN p.name, con.connection ORDER BY p.name ASC"
```

```
DataTrump.DataSet.find({'$and':  
  [{Connections.name_con:'DONALD J. TRUMP'},  
   {'Connections.type_con':  
     {'$regex':'Labor Secretary'}}]})
```

Query 6

Mostra le persone che stanno nelle cerchie di DONALD J. TRUMP scartando quei percorsi che passano per IVANKA TRUMP (figlia), prendendo le persone distinte, e restituisce la persona trovata e un valore booleano che indica se è anche amico di IVANKA TRUMP. Tutto in ordine alfabetico.

```
"MATCH path=(:Person {name:'DONALD J. TRUMP'}) '
  '-[:RELATED_TO*1..3]-(other:Person) WHERE ALL(x IN NODES(path)
    'WHERE x.name <> 'IVANKA TRUMP') WITH DISTINCT(other) as other
    RETURN other,EXISTS( (other)-[:RELATED_TO]-(:Person{name:'IVANKA TRUMP'}))
    as friendOfTrump ORDER BY other.name"
```

```
DataTrump.DataSet.find(
  {'$and':[{'Connections.name_con':'DONALD J. TRUMP'},
  {'type':'Person'},
  {'name':{'$ne':'IVANKA TRUMP'}},
  {'Connections.name_con':'IVANKA TRUMP'}
  ]})
```

Query 7

Ritorna tutte le organizzazioni dove i rispettivi nomi iniziano per TRUMP

```
"MATCH (a:Organization)
  WHERE a.name STARTS WITH 'TRUMP'
  RETURN a ORDER BY a.name"
```

```
DataTrump.DataSet.find({
  'name':{'$regex': '^TRUMP'}})
```

Generatore Dati

Nella fase successiva quella di testing, per verificare la maggiore efficienza di una delle due soluzioni di DataBase, verranno calcolati i tempi di risposta delle query su dimensioni di dataset differenti. Il file CSV è composto da circa 3000 record, per ottenere set di dati di grandezze maggiori è stato implementato un vero e proprio generatore di dati fittizzi. L'idea è stata quella di cercare online nomi e cognomi comuni americani da attribuire all'entità persona e aziende americane per le organization riportate come elenco in un semplice file testuale. Tramite delle funzioni casuali sono stati creati nuovi record che riprendono esattamente il modello del CSV originale.

Esperimenti

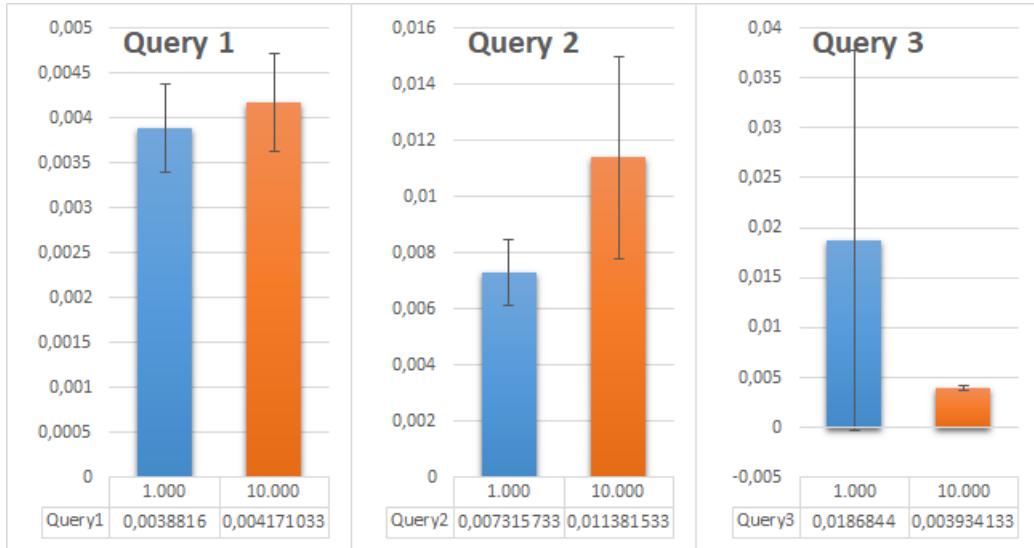
Una volta generati i diversi dataset e importanti nei DB non resta che lanciare le query e farsi restituire i tempi di esecuzione. A causa di meccanismi di chaching, il tempo di esecuzione della prima query sarà sempre maggiore dei tempi delle seguenti query se non viene variato l'input. Per questo motivo ogni interrogazione verrà ripetuta 31 volte e verrà considerato il valore medio con intervalli di confidenza al 95%.

Risultati Neo4j - Dataset di 1000 e 10000

N esecuzione	1 000 NEO4J							
	Query1	Query2	Query3	Query4	Query5	Query6	Query7	
1	1,242245000	1,768433000	4,329502000	2,268962000	2,161198000	1,224087000	1,460535000	
2	0,004737000	0,017369000	0,125132000	0,007500000	0,012632000	0,005526000	0,012237000	
3	0,003553000	0,006316000	0,011053000	0,004737000	0,012632000	0,004343000	0,004342000	
4	0,003158000	0,006316000	0,008684000	0,005132000	0,008684000	0,004342000	0,003948000	
5	0,004342000	0,006710000	0,017763000	0,005526000	0,006711000	0,004342000	0,003552000	
6	0,003553000	0,005132000	0,006710000	0,003947000	0,007500000	0,004342000	0,003948000	
7	0,003157000	0,007106000	0,003948000	0,004342000	0,008684000	0,003947000	0,003947000	
8	0,003158000	0,007500000	0,003553000	0,003552000	0,009868000	0,003947000	0,003947000	
9	0,004737000	0,004737000	0,003157000	0,003553000	0,126712000	0,003553000	0,005921000	
10	0,003553000	0,004737000	0,003158000	0,003948000	0,013026000	0,003948000	0,005922000	
11	0,003158000	0,004736000	0,003158000	0,003948000	0,004342000	0,003948000	0,003947000	
12	0,003157000	0,015395000	0,003947000	0,003947000	0,009869000	0,003948000	0,003552000	
13	0,003158000	0,015000000	0,004737000	0,003158000	0,011053000	0,003947000	0,003553000	
14	0,003158000	0,011842000	0,009079000	0,005131000	0,006315000	0,003948000	0,003552000	
15	0,003553000	0,007500000	0,005526000	0,010263000	0,010658000	0,003553000	0,003948000	
16	0,003158000	0,010263000	0,003947000	0,005922000	0,01447000	0,003553000	0,003553000	
17	0,003158000	0,007500000	0,005527000	0,004342000	0,007105000	0,003158000	0,003948000	
18	0,003157000	0,005131000	0,003948000	0,004737000	0,005921000	0,003158000	0,014605000	
19	0,003158000	0,005132000	0,006710000	0,004737000	0,005921000	0,003158000	0,005131000	
20	0,003158000	0,007894000	0,279870000	0,009474000	0,006315000	0,003553000	0,004342000	
21	0,003158000	0,005131000	0,009474000	0,004342000	0,007105000	0,003552000	0,012631000	
22	0,003947000	0,005131000	0,004343000	0,004343000	0,007500000	0,003158000	0,003948000	
23	0,003553000	0,006316000	0,003553000	0,004737000	0,003553000	0,003552000	0,003948000	
24	0,009474000	0,004737000	0,003553000	0,003947000	0,003157000	0,004342000	0,003553000	
25	0,003158000	0,007105000	0,003158000	0,006316000	0,003553000	0,003553000	0,003948000	
26	0,007500000	0,006316000	0,003553000	0,007500000	0,003158000	0,003158000	0,003553000	
27	0,003552000	0,005921000	0,003553000	0,005527000	0,003553000	0,003158000	0,004342000	
28	0,003158000	0,005526000	0,003552000	0,004343000	0,003158000	0,003158000	0,003948000	
29	0,004737000	0,005526000	0,005922000	0,009473000	0,003158000	0,003158000	0,005131000	
30	0,005132000	0,005526000	0,005527000	0,006315000	0,003158000	0,003158000	0,005131000	
31	0,003158000	0,005921000	0,004737000	0,003948000	0,003158000	0,003552000	0,005132000	
media delle 30	0,0038816	0,0073157	0,0186844	0,005289567	0,010986867	0,003723767	0,005105333	
Std.Dev	0,001382908	0,0032834	0,053121171	0,001819131	0,021726121	0,000536225	0,002783529	
95% Conf	0,000494858	0,0011749	0,019008817	0,000650956	0,00777445	0,000191882	0,000996055	

N esecuzione	10.000 NEO4J						
	Query1	Query2	Query3	Query4	Query5	Query6	Query7
1	1,31764	3,066335	1,364614	2,98265	1,456588	1,657906	1,233956
2	0,005132	0,057237	0,005526	0,043422	0,005921	0,011053	0,004737
3	0,003158	0,015789	0,003553	0,043816	0,005526	0,009474	0,003947
4	0,005131	0,019343	0,003418	0,023685	0,005527	0,005921	0,006316
5	0,004737	0,015395	0,003553	0,01579	0,005132	0,005526	0,005131
6	0,003158	0,022895	0,003553	0,011053	0,005526	0,003552	0,024079
7	0,003158	0,020131	0,005131	0,008684	0,005526	0,005921	0,004737
8	0,003158	0,010658	0,003947	0,008684	0,004737	0,004342	0,003947
9	0,003158	0,011052	0,003158	0,008289	0,004342	0,0075	0,005527
10	0,003158	0,009868	0,004342	0,007894	0,004737	0,009079	0,005921
11	0,003552	0,010658	0,004342	0,008289	0,004737	0,004737	0,005527
12	0,003158	0,008684	0,006315	0,007895	0,003553	0,003947	0,004342
13	0,003158	0,006316	0,003552	0,007895	0,003947	0,004342	0,005526
14	0,011448	0,009079	0,003553	0,007895	0,004342	0,004737	0,005922
15	0,004737	0,005526	0,005131	0,008684	0,004737	0,004737	0,005132
16	0,004737	0,005921	0,003553	0,008289	0,004737	0,004342	0,004342
17	0,004342	0,0075	0,003553	0,007895	0,004737	0,004737	0,004342
18	0,004342	0,005132	0,003948	0,00829	0,004737	0,004342	0,003947
19	0,004342	0,007895	0,003553	0,00829	0,003948	0,004737	0,004342
20	0,004342	0,005131	0,004737	0,007895	0,003947	0,004737	0,003158
21	0,004342	0,004737	0,003947	0,009868	0,003948	0,004737	0,003158
22	0,004737	0,004736	0,004342	0,010658	0,003948	0,005921	0,003158
23	0,004736	0,005526	0,005131	0,009474	0,003947	0,010658	0,003553
24	0,004342	0,005526	0,003947	0,007895	0,0075	0,004342	0,003553
25	0,003553	0,00474	0,003552	0,00829	0,0075	0,003553	0,003158
26	0,003947	0,004737	0,003158	0,008289	0,005526	0,003553	0,003158
27	0,003552	0,02408	0,003157	0,01579	0,005132	0,003552	0,003157
28	0,003552	0,00671	0,003158	0,008289	0,004737	0,003158	0,003158
29	0,003553	0,009868	0,003158	0,007895	0,004737	0,003158	0,003158
30	0,003553	0,005526	0,003158	0,011053	0,004342	0,003158	0,003158
31	0,003158	0,006316	0,003158	0,011052	0,004737	0,004343	0,003158
media delle 30	0,004171033	0,0113815	0,00934133	0,012039567	0,004881667	0,0052632	0,004881633
Std.Dev	0,001507281	0,0101191	0,000811868	0,00905206	0,000914301	0,002121707	0,003702262
95% Conf	0,000539364	0,003621	0,000290518	0,003239179	0,000327172	0,000759229	0,001324813

Confronto dei risultati





Risultati MongoDB - Dataset di 1000 e 10000

N esecuzione	1.000							
	MONGO							
	Query1	Query2	Query3	Query4	Query5	Query6	Query7	
1	1,998960000	126,629600000	48,045310000	3,221468000	1,867117000	2,058171000	4,069368000	
2	0,082895000	20,778690000	15,588260000	0,115264000	0,081711000	0,093159000	0,101448000	
3	0,072632000	18,515250000	16,393530000	0,122369000	0,041053000	0,087237000	0,094342000	
4	0,069869000	15,092860000	15,332070000	0,094737000	0,048947000	0,067501000	0,042632000	
5	0,042632000	22,334750000	11,119810000	0,081316000	0,065132000	0,187501000	0,178422000	
6	0,069080000	18,416570000	13,563640000	0,119211000	0,049343000	0,064737000	0,198949000	
7	0,061974000	12,178110000	15,619840000	0,112501000	0,047368000	0,139343000	0,723558000	
8	0,065132000	24,809370000	18,396040000	0,098290000	0,053685000	0,057632000	0,138553000	
9	0,063948000	15,425630000	19,236440000	0,069474000	0,045790000	0,034342000	0,049737000	
10	0,291318000	17,481030000	15,253120000	0,067500000	0,056843000	0,049737000	0,092369000	
11	0,054869000	18,647490000	14,885230000	0,089211000	0,075790000	0,082501000	0,041053000	
12	0,062763000	16,974190000	17,281690000	0,086448000	0,236844000	0,101053000	0,042237000	
13	0,079343000	17,600640000	21,008420000	0,066711000	0,090790000	0,069475000	0,062763000	
14	0,073816000	19,188680000	25,037920000	0,050922000	0,055263000	0,394739000	0,058816000	
15	0,043816000	18,730780000	14,375220000	0,175264000	0,044606000	0,049342000	0,046974000	
16	0,076974000	18,759990000	17,453400000	0,702241000	0,045000000	0,052500000	0,050527000	
17	0,095527000	21,225530000	20,304210000	0,083290000	0,065922000	0,053290000	0,055658000	
18	0,083684000	97,801030000	15,014700000	0,179606000	0,067895000	0,052106000	0,044606000	
19	0,056447000	13,182320000	15,563000000	0,186712000	0,077369000	0,054474000	0,172896000	
20	0,049342000	13,451930000	17,314850000	0,097895000	0,195790000	0,046974000	0,035131000	
21	0,052501000	17,977620000	13,915350000	0,080132000	0,049738000	0,048158000	0,099079000	
22	0,048553000	13,207190000	17,021160000	0,175265000	0,078553000	0,048158000	0,172501000	
23	0,057237000	17,720640000	16,050100000	0,081711000	0,082895000	0,052106000	0,168159000	
24	0,078158000	13,579820000	15,908000000	0,028027000	0,039474000	0,100659000	0,067106000	
25	0,074605000	14,155750000	16,646550000	0,049737000	0,048947000	0,044606000	0,104606000	
26	0,056053000	11,202310000	24,340030000	0,067106000	0,048553000	0,084474000	0,046184000	
27	0,049738000	32,560080000	17,878930000	0,080921000	0,193422000	0,089606000	0,048947000	
28	0,047764000	20,608160000	12,741790000	0,056447000	0,136185000	0,085264000	0,043816000	
29	0,068685000	19,200000000	20,126970000	0,058816000	0,062369000	0,081317000	0,046579000	
30	0,062369000	14,930000000	14,497590000	0,050921000	0,047763000	0,087238000	0,040263000	
31	0,050526000	11,310000000	17,524450000	0,072237000	27,016600000	0,087238000	0,078158000	
media delle 30	0,071408333	20,23513	16,84641	0,113343	0,974988	0,0848822	0,104869	
Std.Dev	0,042853336	15,03029	3,016091	0,116465	4,836049	0,0652588	0,124863	
95% Conf	0,015334588	5,378421	1,079274	0,041676	1,730526	0,0233521	0,044681	

Confronto Dataset da 1000 per i due DB

