

DATABASES

RUN TIME

ANALYSIS (MONGODB, NEO4J, MARIADB)

Logo, company name

Description automatically generated

BY NAVEEN TIWARI

MAT ID: 522364

Objective:

In this report I am going to do a benchmark analysis of the runtime of three different databases: MongoDB, Neo4j, MariaDB. The databases contain information of the lost credit cards for that each database is divided into four tables of sizes: 25000, 50000, 75000 and 100000 rows. All the data have been imported from <https://www.onlinedatagenerator.com/>.

In four different csv files with sizes mention above. To get the runtime analysis I have used four queres that gives same output for the three-database system with increasing complexity.

The runtime analysis has been performed using Python. For MongoDB and MariaDB I have used docker containers and a normal Neo4j application because it was preinstalled.

GENERAL OVERVIEW OF THE DATA MODEL USED:

Table

Description automatically generated

Logo, company name

Description automatically generated

***MongoDB****is a*[*source-available*](https://en.wikipedia.org/wiki/Source-available)[*cross-platform*](https://en.wikipedia.org/wiki/Cross-platform)[*document-oriented database*](https://en.wikipedia.org/wiki/Document-oriented_database)*program. Classified as a*[*NoSQL*](https://en.wikipedia.org/wiki/NoSQL)*database program, MongoDB uses*[*JSON*](https://en.wikipedia.org/wiki/JSON)*-like documents with optional*[*schemas*](https://en.wikipedia.org/wiki/Database_schema)*. MongoDB is developed by*[*MongoDB Inc.*](https://en.wikipedia.org/wiki/MongoDB_Inc.)*and licensed under the*[*Server Side Public License*](https://en.wikipedia.org/wiki/Server_Side_Public_License)*(SSPL) which is deemed non-free by several distributions. The latter type of document extends the Json model to provide data types additional, ordered fields and to be efficient in encoding and decoding with several programming languages. A MongoDB instance can have zero or more databases, each of which acts as a top-tier container for everything else. A database can have zero or more 'collections'. A collection has a lot in common with Traditional 'tables'. The collections are made up of zero or more 'documents', the latter can be compared to the 'rows' (records) of a table. A document is in turn composed of one or more 'fields', similar to the concept of 'Columns'.*



***Neo4j****is a*[*graph database*](https://en.wikipedia.org/wiki/Graph_database)[*management system*](https://en.wikipedia.org/wiki/Management_system)*developed by Neo4j, Inc. Described by its developers as an*[*ACID*](https://en.wikipedia.org/wiki/ACID)*-compliant transactional database with native graph storage and processing, Neo4j is available in a*[*GPL3*](https://en.wikipedia.org/wiki/GNU_General_Public_License)*-licensed*[*source-available*](https://en.wikipedia.org/wiki/Source-available)*"community edition", with*[*online backup*](https://en.wikipedia.org/wiki/Remote_backup_service)*and*[*high availability*](https://en.wikipedia.org/wiki/High_availability)*extensions licensed under a closed-source commercial license. Neo also licenses Neo4j with these extensions under closed-source commercial terms. Neo4j is implemented in*[*Java*](https://en.wikipedia.org/wiki/Java_(programming_language))*and accessible from software written in other languages using the*[*Cypher query language*](https://en.wikipedia.org/wiki/Cypher_(query_language))*through a transactional HTTP endpoint, or through the binary "*[*Bolt*](https://en.wikipedia.org/wiki/Bolt_(network_protocol))*" protocol The "4j" in Neo4j is a reference to its being built in Java, however is now largely viewed as an*[*anachronism*](https://en.wikipedia.org/wiki/Anachronism)*. It is a Nosql database that is based on graph dbms that implements a database management that is flexible in dealing with incredibly structured data. Neo4j graph DBMS exploit graphs for data storage, allowing the management of the management of closely related data structures, as in the case of social networks. Neo4j, not only allows us to store data, but also offers tools for the study of graphs , with the possibility of implementing very complex queries for the identification of nodes and arcs. More specifically, the graph that can be created is made up of nodes, arcs and property. Each node has properties, which represent our records and data from to memorize. The arcs, on the other hand, have a direction and represent the relationships between the nodes. Neo4j supports a declarative language called “Cypher”, a declarative language inspired by SQL. Cypher allows you to indicate what we want to select, insert, update or delete from our graphic data without a description of exactly how to do it.*



***MariaDB****is a community-developed, commercially supported*[*fork*](https://en.wikipedia.org/wiki/Fork_(software_development))*of the*[*MySQL*](https://en.wikipedia.org/wiki/MySQL)[*relational database management system*](https://en.wikipedia.org/wiki/Relational_database_management_system)*(RDBMS), intended to remain*[*free and open-source software*](https://en.wikipedia.org/wiki/Free_and_open-source_software)*under the*[*GNU General Public License*](https://en.wikipedia.org/wiki/GNU_General_Public_License)*. Development is led by some of the original developers of MySQL, who forked it due to concerns over its*[*acquisition*](https://en.wikipedia.org/wiki/Takeover)*by*[*Oracle Corporation*](https://en.wikipedia.org/wiki/Oracle_Corporation)*in 2009.*[*[6]*](https://en.wikipedia.org/wiki/MariaDB#cite_note-6)

*MariaDB is intended to maintain high compatibility with MySQL, with library binary parity and exact matching with MySQL*[*APIs*](https://en.wikipedia.org/wiki/Application_programming_interface)*and commands, allowing it in many cases to function as drop-in replacement for MySQL. However, new features are diverging.*[*[7]*](https://en.wikipedia.org/wiki/MariaDB#cite_note-7)*It includes new*[*storage engines*](https://en.wikipedia.org/wiki/Storage_engine)*like*[*Aria*](https://en.wikipedia.org/wiki/Aria_(storage_engine))*,*[*ColumnStore*](https://en.wikipedia.org/wiki/InfiniDB)*, and*[*MyRocks*](https://en.wikipedia.org/wiki/MyRocks)*.*

*Its lead developer/CTO is*[*Michael "Monty" Widenius*](https://en.wikipedia.org/wiki/Michael_Widenius)*, one of the founders of*[*MySQL AB*](https://en.wikipedia.org/wiki/MySQL_AB)*and the founder of Monty Program AB. On 16 January 2008, MySQL AB announced that it had agreed to be acquired by*[*Sun Microsystems*](https://en.wikipedia.org/wiki/Sun_Microsystems)*for approximately $1 billion. The acquisition completed on 26 February 2008. Sun was then bought the following year by*[*Oracle Corporation*](https://en.wikipedia.org/wiki/Oracle_Corporation)*. MariaDB is named after Widenius' younger daughter, Maria. (MySQL is named after his other daughter, My*

Implementation:

MongoDB:

For this database I have used docker. To make a mongodb container I have used the command:

docker run -d -p 27017:27017 --name dbmongo mongo:latest

As we have created a container so now we need to import the csv files to mongodb folder in order to use it for creating database for this we use the command:

docker cp /Users/naveen/Desktop/25.csv mongodb:/bin

now we will go inside the mongodb container using command:

docker exec -it dbmongo bash

Now we need to create four different collections from the csv files, for this we use the command:

mongoimport --db Lost\_credit\_card --collection db25 --type csv --file bin/25.csv –headerline

AS we have to create four collections inside the database Lost\_credit\_card , so we will change the name of collections and source csv file.

* We are done with MongoDB now we need to connect it with python.

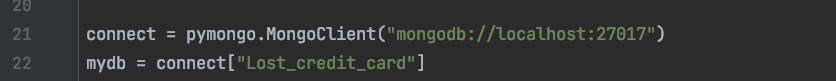
Libraries imported:

Graphical user interface, text

Description automatically generated with medium confidence

I have imported libraires pymongo, neo4j, MySQL for their respective databases. Along with it I have also used pandas and csv to read the csv data that we generated and to return the runtime of all the queries.

Now we need to make connection with the mongoDB database.



As we have already crated the collections, we just need to perform the queries. For this report we will run it for 31 times.

*Text

Description automatically generated*

Text

Description automatically generated

Now we take the run time of that query and append it in xk array which we can use later to export in csv file. Using this I have performed the query four times in four different processes.

NEO4J:

As I already had the application installed, so I just need to make node using the csv files data for that I have first created a database named “DBPT”.

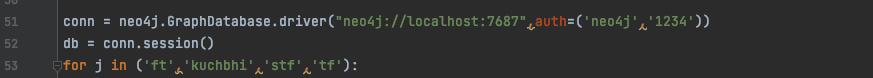
In order to load data from csv files we need to use command:

A picture containing text

Description automatically generated

As we need to make four labels so we will change the name of the labels which is just after CREATE line, E.g., “kuchbhi”.

Now we will initiate the connection procedure:



Once connected we will perform the queries for 31 times.

Text

Description automatically generated

Where xn stores the run time.

MARIADB:

First we need to make a container:

docker run --name mariadbproject -e MYSQL\_ROOT\_PASSWORD=1234 -p 3306:3306 -d docker.io/library/mariadb:10.3

Now we will import the csv file to mariadb folder:

In every step we change the name of csv file along with table name.

load data local infile 'bin/ExportCSV1.csv' into table db100 fields terminated by ',' ignore 1 lines;

Now we go inside the container using the credentials:

docker exec -it mariadbproject bash

mariadb -u root -p

enter password:

Now we will create the database “Lost\_credit\_card” using command :

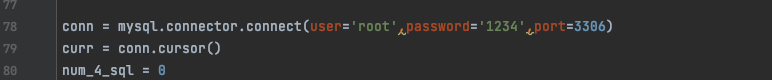
Create database Lost\_credit\_card;

Now we need to create four tables to import the data of the four csv files :

create table db100 (GUID int,First\_Name varchar(150),Last\_Name varchar(50),Gender varchar(100) , Email\_Address varchar(500), Phone\_Number int,Credit\_Card\_Number int,Credit\_Card\_Type varchar(100),IBAN int,City varchar(50),primary key(GUID));

We will just change the name of tables like db100, db75 etc, rest will remain same.

As we are finished with mariadb now we will connect to python:



Text

Description automatically generated

Now we perform the query 31 times:

Text

Description automatically generated

cm gives us the time it took for the query. This way we can run queries for different data set.

Now we have runned all the queries for different data sets, now we can export it to csv file to analyze. Where m is for mongodb, n for neo4j, ma for mariadb.

Text

Description automatically generated

DATABASE PERFORMANCE:

First Query:

MongoDB:

Text

Description automatically generated

Neo4j:

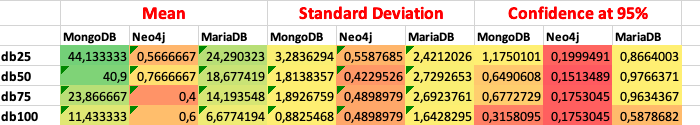


MariaDB:

Graphical user interface, text, application

Description automatically generated

Performance:



Second Query:

MongoDB:



Neo4j:



MariaDB:



Performance:



Third Query:

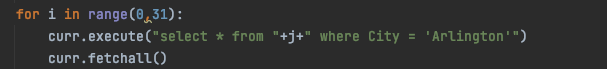
MongoDB:



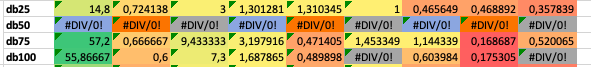
Neo4j:



MariaDB:



Performance:



Fourth Query:

MongoDB:



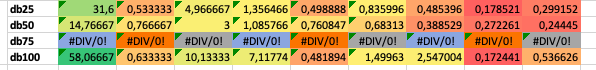
Neo4j:



MariaDB:



Performance:



RECAPICULATING:

If we will have a look at the graphs, the overall time consumption of MongoDB is the most among three and neo4j is the least, whereas MariaDB in middle of both.

There is a slight variation in the First Query where for db75 and db100 time consumption is maximum for MariaDB.

This happens because the complexity of the mariadb query in db75 and db100, is more than as of MongoDB so the time reduces.

So, we can make an analysis that even though we perform same queries at these three databases the performance of neo4j is the best in short as well as long run.

THE END